

Business Intelligence Universidades

Generado por Doxygen 1.9.3

1 Índice de namespaces	1
1.1 Lista de 'namespaces'	1
2 Índice de clases	5
2.1 Lista de clases	5
3 Índice de archivos	7
3.1 Lista de archivos	7
4 Documentación de namespaces	13
4.1 Referencia del Namespace app	13
4.1.1 Documentación de las variables	13
4.1.1.1 _favicon	13
4.1.1.2 app	13
4.1.1.3 debug	14
4.1.1.4 layout	14
4.1.1.5 server	14
4.1.1.6 title	14
4.2 Referencia del Namespace callbacks	14
4.3 Referencia del Namespace callbacks.alumnado	14
4.4 Referencia del Namespace callbacks.alumnado.filters	15
4.5 Referencia del Namespace callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado	15
4.5.1 Documentación de las funciones	15
4.5.1.1 update_filter_asignaturas_matri_alumnado()	15
4.6 Referencia del Namespace callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado	16
4.6.1 Documentación de las funciones	16
4.6.1.1 update_filter_curso_academico_alumnado()	16
4.7 Referencia del Namespace callbacks.alumnado.filters.callback_filter_titulacion_alumnado	17
4.7.1 Documentación de las funciones	18
4.7.1.1 update_filter_titulacion_alumnado()	18
4.8 Referencia del Namespace callbacks.alumnado.graphs	19
4.9 Referencia del Namespace callbacks.alumnado.graphs.general	19
4.10 Referencia del Namespace callbacks.alumnado.graphs.general.callback_graph_asig_superadas_← media	19
4.10.1 Documentación de las funciones	19
4.10.1.1 update_graph_alumnado()	19
4.11 Referencia del Namespace callbacks.alumnado.graphs.general.callback_graph_calif_cual_← comparativa	21
4.11.1 Documentación de las funciones	21
4.11.1.1 update_graph_alumnado()	21
4.12 Referencia del Namespace callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_nota	23
4.12.1 Documentación de las funciones	23
4.12.1.1 update_graph_alumnado()	23
4.13 Referencia del Namespace callbacks.alumnado.graphs.personal	25

4.14 Referencia del Namespace callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_← alumnado	25
4.14.1 Documentación de las funciones	25
4.14.1.1 update_graph_alumnado()	26
4.15 Referencia del Namespace callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_← alumnado	27
4.15.1 Documentación de las funciones	27
4.15.1.1 update_graph_alumnado()	27
4.16 Referencia del Namespace callbacks.alumnado.graphs.personal.callback_graph_progreso_← academico_alumnado	29
4.16.1 Documentación de las funciones	29
4.16.1.1 update_graph_alumnado()	29
4.17 Referencia del Namespace callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_← _alumnado	30
4.17.1 Documentación de las funciones	30
4.17.1.1 update_graph_alumnado()	30
4.18 Referencia del Namespace callbacks.alumnado.utils	31
4.19 Referencia del Namespace callbacks.alumnado.utils.callback_resumen_alumnado	32
4.19.1 Documentación de las funciones	32
4.19.1.1 not_data()	32
4.19.1.2 update_resumen_alumnado()	32
4.20 Referencia del Namespace callbacks.alumnado.utils.callback_select_alumnado	34
4.20.1 Documentación de las funciones	34
4.20.1.1 store_selected_alumnado()	34
4.21 Referencia del Namespace callbacks.alumnado.utils.callback_tabs_alumnado	35
4.21.1 Documentación de las funciones	35
4.21.1.1 render_content()	35
4.22 Referencia del Namespace callbacks.common	36
4.23 Referencia del Namespace callbacks.common.callback_sidebarCollapse	36
4.23.1 Documentación de las funciones	37
4.23.1.1 sidebarCollapse()	37
4.24 Referencia del Namespace callbacks.common.callback_update_layout	37
4.24.1 Documentación de las funciones	37
4.24.1.1 update_layout()	38
4.25 Referencia del Namespace callbacks.common.callback_user_role	38
4.25.1 Documentación de las funciones	39
4.25.1.1 initialize_dropdown()	39
4.25.1.2 update_role()	39
4.25.2 Documentación de las variables	40
4.25.2.1 prevent_initial_call	40
4.26 Referencia del Namespace callbacks.docente	40
4.27 Referencia del Namespace callbacks.docente.filters	40
4.28 Referencia del Namespace callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente	40

4.28.1 Documentación de las funciones	40
4.28.1.1 update_filter_asignaturas_docente()	41
4.29 Referencia del Namespace callbacks.docente.filters.callback_filter_all_curso_academico	42
4.29.1 Documentación de las funciones	42
4.29.1.1 update_filter_all_cursos_academicos_docente()	42
4.30 Referencia del Namespace callbacks.docente.filters.callback_filter_asignaturas_docente	43
4.30.1 Documentación de las funciones	43
4.30.1.1 update_filter_asignaturas_docente()	43
4.31 Referencia del Namespace callbacks.docente.filters.callback_filter_curso_academico_docente	44
4.31.1 Documentación de las funciones	44
4.31.1.1 update_filter_curso_academico_docente()	44
4.32 Referencia del Namespace callbacks.docente.filters.callback_filter_titulacion_docente	45
4.32.1 Documentación de las funciones	45
4.32.1.1 update_filter_titulacion_docente()	45
4.33 Referencia del Namespace callbacks.docente.graphs	46
4.34 Referencia del Namespace callbacks.docente.graphs.general	46
4.35 Referencia del Namespace callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente	46
4.35.1 Documentación de las funciones	46
4.35.1.1 update_graph_docente()	47
4.36 Referencia del Namespace callbacks.docente.graphs.general.callback_graph_all_calif_media_docente	48
4.36.1 Documentación de las funciones	48
4.36.1.1 update_graph_docente()	48
4.37 Referencia del Namespace callbacks.docente.graphs.personal	49
4.38 Referencia del Namespace callbacks.docente.graphs.personal.callback_graph_alu_genero_docente	50
4.38.1 Documentación de las funciones	50
4.38.1.1 update_graph_docente()	50
4.39 Referencia del Namespace callbacks.docente.graphs.personal.callback_graph_alu_media_docente	51
4.39.1 Documentación de las funciones	51
4.39.1.1 update_graph_docente()	51
4.40 Referencia del Namespace callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente	52
4.40.1 Documentación de las funciones	53
4.40.1.1 update_graph_docente()	53
4.41 Referencia del Namespace callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente	54
4.41.1 Documentación de las funciones	54
4.41.1.1 update_graph_docente()	54
4.42 Referencia del Namespace callbacks.docente.utils	56
4.43 Referencia del Namespace callbacks.docente.utils.callback_resumen_docente	56
4.43.1 Documentación de las funciones	56
4.43.1.1 not_data()	56
4.43.1.2 update_resumen_docente()	57

4.44 Referencia del Namespace callbacks.docente.utils.callback_select_docente	57
4.44.1 Documentación de las funciones	58
4.44.1.1 store_selected_docente()	58
4.45 Referencia del Namespace callbacks.docente.utils.callback_tabs_docente	58
4.45.1 Documentación de las funciones	59
4.45.1.1 render_content()	59
4.46 Referencia del Namespace callbacks.gestor	60
4.47 Referencia del Namespace callbacks.gestor.filters	60
4.48 Referencia del Namespace callbacks.gestor.filters.callback_filter_all_curso_academico_gestor	61
4.48.1 Documentación de las funciones	61
4.48.1.1 update_filter_all_curso_academico_gestor()	61
4.49 Referencia del Namespace callbacks.gestor.filters.callback_filter_curso_academico_gestor	62
4.49.1 Documentación de las funciones	62
4.49.1.1 update_filter_curso_academico_gestor()	62
4.50 Referencia del Namespace callbacks.gestor.filters.callback_filter_titulaciones_gestor	63
4.50.1 Documentación de las funciones	63
4.50.1.1 update_filter_titulaciones_gestor()	63
4.51 Referencia del Namespace callbacks.gestor.graphs	65
4.52 Referencia del Namespace callbacks.gestor.graphs.indicadores	65
4.53 Referencia del Namespace callbacks.gestor.graphs.indicadores.callback_graph_egresados_genro_gestor	65
4.53.1 Documentación de las funciones	65
4.53.1.1 generate_csv()	65
4.53.1.2 get_data()	66
4.53.1.3 toggle_modal()	67
4.53.1.4 update_graph_gestor()	68
4.53.1.5 update_table()	70
4.54 Referencia del Namespace callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor	71
4.54.1 Documentación de las funciones	71
4.54.1.1 generate_csv()	71
4.54.1.2 get_data()	72
4.54.1.3 toggle_modal()	73
4.54.1.4 update_graph_gestor()	74
4.54.1.5 update_table()	75
4.55 Referencia del Namespace callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genro_gestor	76
4.55.1 Documentación de las funciones	76
4.55.1.1 generate_csv()	76
4.55.1.2 get_data()	77
4.55.1.3 toggle_modal()	79
4.55.1.4 update_graph_gestor()	79
4.55.1.5 update_table()	81

4.56 Referencia del Namespace callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_← nacionalidad_gestor	82
4.56.1 Documentación de las funciones	82
4.56.1.1 generate_csv()	82
4.56.1.2 get_data()	83
4.56.1.3 toggle_modal()	85
4.56.1.4 update_graph_gestor()	85
4.56.1.5 update_table()	86
4.57 Referencia del Namespace callbacks.gestor.graphs.resultados	87
4.58 Referencia del Namespace callbacks.gestor.graphs.resultados.callback_graph_duracion_media_← estudios_nota_gestor	88
4.58.1 Documentación de las funciones	88
4.58.1.1 generate_csv()	88
4.58.1.2 get_data()	89
4.58.1.3 toggle_modal()	90
4.58.1.4 update_graph_gestor()	91
4.58.1.5 update_table()	92
4.59 Referencia del Namespace callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_← titulacion_gestor	93
4.59.1 Documentación de las funciones	93
4.59.1.1 generate_csv()	93
4.59.1.2 get_data()	94
4.59.1.3 toggle_modal()	95
4.59.1.4 update_grph_gestor()	96
4.59.1.5 update_table()	97
4.60 Referencia del Namespace callbacks.gestor.graphs riesgo_abandono	98
4.61 Referencia del Namespace callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_← abandono_gestor	98
4.61.1 Documentación de las funciones	98
4.61.1.1 generate_csv()	99
4.61.1.2 get_data()	99
4.61.1.3 toggle_modal()	101
4.61.1.4 update_graph_gestor()	102
4.61.1.5 update_table()	103
4.62 Referencia del Namespace callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_← graduacion_gestor	104
4.62.1 Documentación de las funciones	104
4.62.1.1 generate_csv()	104
4.62.1.2 get_data()	105
4.62.1.3 toggle_modal()	106
4.62.1.4 update_graph_gestor()	107
4.62.1.5 update_table()	108
4.63 Referencia del Namespace callbacks.gestor.utils	109

4.64 Referencia del Namespace callbacks.gestor.utils.callback_resumen_gestor	109
4.64.1 Documentación de las funciones	109
4.64.1.1 not_data()	110
4.64.1.2 update_resumen_gestor()	110
4.65 Referencia del Namespace callbacks.gestor.utils.callback_select_gestor	111
4.65.1 Documentación de las funciones	111
4.65.1.1 store_selected_gestor()	111
4.66 Referencia del Namespace callbacks.gestor.utils.callback_tabs_gestor	112
4.66.1 Documentación de las funciones	112
4.66.1.1 render_content()	112
4.67 Referencia del Namespace components	114
4.68 Referencia del Namespace components.alumnado	114
4.69 Referencia del Namespace components.alumnado.filters	114
4.70 Referencia del Namespace components.alumnado.filters.filter_asignaturas_matri_alumnado	115
4.70.1 Documentación de las funciones	115
4.70.1.1 filter_asignaturas_matri_alumnado()	115
4.71 Referencia del Namespace components.alumnado.filters.filter_curso_academico_alumnado	116
4.71.1 Documentación de las funciones	116
4.71.1.1 filter_curso_academico_alumnado()	116
4.72 Referencia del Namespace components.alumnado.filters.filter_titulacion_alumnado	117
4.72.1 Documentación de las funciones	117
4.72.1.1 filter_titulacion_alumnado()	117
4.73 Referencia del Namespace components.alumnado.graphs	118
4.74 Referencia del Namespace components.alumnado.graphs.graphs_general_alumnado	118
4.74.1 Documentación de las funciones	118
4.74.1.1 graphs_general_alumnado()	118
4.75 Referencia del Namespace components.alumnado.graphs.graphs_personal_alumnado	119
4.75.1 Documentación de las funciones	119
4.75.1.1 graphs_personal_alumnado()	119
4.76 Referencia del Namespace components.alumnado.utils	120
4.77 Referencia del Namespace components.alumnado.utils.recomendador_alumnado	120
4.77.1 Documentación de las funciones	120
4.77.1.1 recomendador_alumnado()	121
4.78 Referencia del Namespace components.alumnado.utils.resumen_alumnado	122
4.78.1 Documentación de las funciones	122
4.78.1.1 resumen_alumnado()	122
4.79 Referencia del Namespace components.alumnado.utils.select_alumnado	122
4.79.1 Documentación de las funciones	122
4.79.1.1 select_alumnado()	123
4.80 Referencia del Namespace components.alumnado.utils.tabs_alumnado	123
4.80.1 Documentación de las funciones	123
4.80.1.1 tabs_alumnado()	124

4.81 Referencia del Namespace components.common	124
4.82 Referencia del Namespace components.common.create_graph	124
4.82.1 Documentación de las funciones	125
4.82.1.1 create_graph()	125
4.83 Referencia del Namespace components.common.create_graph_with_table	125
4.83.1 Documentación de las funciones	126
4.83.1.1 create_graph_with_table()	126
4.84 Referencia del Namespace components.common.filters	127
4.84.1 Documentación de las funciones	127
4.84.1.1 filters()	127
4.85 Referencia del Namespace components.common.footer	128
4.85.1 Documentación de las funciones	128
4.85.1.1 footer()	128
4.86 Referencia del Namespace components.common.header	128
4.86.1 Documentación de las funciones	129
4.86.1.1 header()	129
4.87 Referencia del Namespace components.common.modal_data	129
4.87.1 Documentación de las funciones	129
4.87.1.1 create_modal()	130
4.88 Referencia del Namespace components.common.sidebar	130
4.88.1 Documentación de las funciones	131
4.88.1.1 sidebar()	131
4.89 Referencia del Namespace components.docente	131
4.90 Referencia del Namespace components.docente.filters	132
4.91 Referencia del Namespace components.docente.filters.filter_all_asignaturas_titulacion_docente	132
4.91.1 Documentación de las funciones	132
4.91.1.1 filter_all_asignaturas_titulacion_docente()	132
4.92 Referencia del Namespace components.docente.filters.filter_all_curso_academico	133
4.92.1 Documentación de las funciones	133
4.92.1.1 filter_all_curso_academico()	133
4.93 Referencia del Namespace components.docente.filters.filter_asignaturas_docente	134
4.93.1 Documentación de las funciones	134
4.93.1.1 filter_asignaturas_docente()	134
4.94 Referencia del Namespace components.docente.filters.filter_curso_academico_docente	135
4.94.1 Documentación de las funciones	135
4.94.1.1 filter_curso_academico_docente()	135
4.95 Referencia del Namespace components.docente.filters.filter_titulacion_docente	136
4.95.1 Documentación de las funciones	136
4.95.1.1 filter_titulacion_docente()	136
4.96 Referencia del Namespace components.docente.graphs	137
4.97 Referencia del Namespace components.docente.graphs.graphs_general_docente	137
4.97.1 Documentación de las funciones	137

4.97.1.1 graphs_general_docente()	138
4.98 Referencia del Namespace components.docente.graphs.graphs_personal_docente	138
4.98.1 Documentación de las funciones	139
4.98.1.1 graphs_personal_docente()	139
4.99 Referencia del Namespace components.docente.utils	140
4.100 Referencia del Namespace components.docente.utils.recomendador_docente	140
4.100.1 Documentación de las funciones	140
4.100.1.1 recomendador_docente()	140
4.101 Referencia del Namespace components.docente.utils.resumen_docente	141
4.101.1 Documentación de las funciones	141
4.101.1.1 resumen_docente()	141
4.102 Referencia del Namespace components.docente.utils.select_docente	142
4.102.1 Documentación de las funciones	142
4.102.1.1 select_docente()	142
4.103 Referencia del Namespace components.docente.utils.tabs_docente	143
4.103.1 Documentación de las funciones	143
4.103.1.1 tabs_docente()	143
4.104 Referencia del Namespace components.gestor	143
4.105 Referencia del Namespace components.gestor.filters	144
4.106 Referencia del Namespace components.gestor.filters.filter_all_curso_academico_gestor	144
4.106.1 Documentación de las funciones	144
4.106.1.1 filter_all_curso_academico_gestor()	144
4.107 Referencia del Namespace components.gestor.filters.filter_curso_academico_gestor	145
4.107.1 Documentación de las funciones	145
4.107.1.1 filter_curso_academico_gestor()	145
4.108 Referencia del Namespace components.gestor.filters.filter_titulaciones_gestor	146
4.108.1 Documentación de las funciones	146
4.108.1.1 filter_titulaciones_gestor()	146
4.109 Referencia del Namespace components.gestor.graphs	147
4.110 Referencia del Namespace components.gestor.graphs.graphs_indicadores_gestor	147
4.110.1 Documentación de las funciones	147
4.110.1.1 graphs_indicadores_gestor()	148
4.111 Referencia del Namespace components.gestor.graphs.graphs_resultados_gestor	149
4.111.1 Documentación de las funciones	149
4.111.1.1 graphs_resultados_gestor()	149
4.112 Referencia del Namespace components.gestor.graphs.graphs_riesgo_abandono_gestor	150
4.112.1 Documentación de las funciones	150
4.112.1.1 graphs_riesgo_abandono_gestor()	151
4.113 Referencia del Namespace components.gestor.utils	152
4.114 Referencia del Namespace components.gestor.utils.recomendador_gestor	152
4.114.1 Documentación de las funciones	152
4.114.1.1 recomendador_gestor()	152

4.115 Referencia del Namespace components.gestor.utils.resumen_gestor	153
4.115.1 Documentación de las funciones	153
4.115.1.1 resumen_gestor()	154
4.116 Referencia del Namespace components.gestor.utils.select_gestor	154
4.116.1 Documentación de las funciones	154
4.116.1.1 select_gestor()	154
4.117 Referencia del Namespace components.gestor.utils.tabs_gestor	155
4.117.1 Documentación de las funciones	155
4.117.1.1 tabs_gestor()	155
4.118 Referencia del Namespace data	156
4.119 Referencia del Namespace data.db_connector	156
4.119.1 Documentación de las variables	156
4.119.1.1 config	157
4.119.1.2 db	157
4.120 Referencia del Namespace data.generate_synthetic_data	157
4.120.1 Documentación de las funciones	158
4.120.1.1 generate_alumnos()	158
4.120.1.2 generate_asignaturas()	158
4.120.1.3 generate_asignaturas_matriculadas()	159
4.120.1.4 generate_curso_aca()	160
4.120.1.5 generate_docentes()	160
4.120.1.6 generate_ebau_prueba()	161
4.120.1.7 generate_egresados()	162
4.120.1.8 generate_gestores()	162
4.120.1.9 generate_lineas_actas()	163
4.120.1.10 generate_matricula()	164
4.120.1.11 generate_unique_cod_asignaturas()	165
4.120.2 Documentación de las variables	165
4.120.2.1 alumnos_df	165
4.120.2.2 asignaturas_dict	165
4.120.2.3 asignaturas_matriculadas_df	165
4.120.2.4 cod_asignaturas_dict	166
4.120.2.5 cod_plan_dict	166
4.120.2.6 cursos_academicos	166
4.120.2.7 docentes_df	166
4.120.2.8 ebau_prueba_df	166
4.120.2.9 educacion	166
4.120.2.10 egresados_df	167
4.120.2.11 fake	167
4.120.2.12 gestores_df	167
4.120.2.13 index	167
4.120.2.14 lineas_actas_df	167

4.120.2.15 matricula_df	167
4.120.2.16 num_alumnos	168
4.120.2.17 num_docentes	168
4.120.2.18 num_gestores	168
4.120.2.19 universidades_dict	168
4.121 Referencia del Namespace data.queries	168
4.121.1 Documentación de las funciones	169
4.121.1.1 alumnos_all()	170
4.121.1.2 alumnos_egresados_genero_titulacion()	170
4.121.1.3 alumnos_egresados_nacionalidad_titulacion()	171
4.121.1.4 alumnos_genero_docente()	172
4.121.1.5 alumnos_nota_cualitativa_docente()	173
4.121.1.6 alumnos_nota_media_docente()	173
4.121.1.7 alumnos_nuevo_ingreso_genero_titulacion()	174
4.121.1.8 alumnos_nuevo_ingreso_nacionalidad_titulacion()	175
4.121.1.9 alumnos_repetidores_nuevos()	176
4.121.1.10 asignaturas_actas_titulacion()	176
4.121.1.11 asignaturas_docente()	177
4.121.1.12 asignaturas_matriculadas()	178
4.121.1.13 asignaturas_matriculadas_y_superadas()	179
4.121.1.14 asignaturas_superadas()	179
4.121.1.15 asignaturas_superadas_media_abandono()	180
4.121.1.16 cache_query()	181
4.121.1.17 calif_all_cualitativa_asignaturas()	182
4.121.1.18 calif_cualitativa_alumno_asignaturas()	182
4.121.1.19 calif_cualitativa_asignatura()	183
4.121.1.20 calif_cualitativa_comparativa()	184
4.121.1.21 calif_media_asignaturas()	185
4.121.1.22 calif_numerica_asignatura()	186
4.121.1.23 check_data()	186
4.121.1.24 curso_academico_actas_titulacion()	188
4.121.1.25 curso_academico_alumnado()	189
4.121.1.26 curso_academico_docente()	190
4.121.1.27 curso_academico_universidad()	191
4.121.1.28 cursos_academicos_egresados()	191
4.121.1.29 data_for_model()	192
4.121.1.30 docentes_all()	193
4.121.1.31 duracion_media_estudios_nota_gestor()	194
4.121.1.32 gestores_all()	194
4.121.1.33 nota_media_acceso_titulacion()	195
4.121.1.34 nota_media_alumno_titulacion()	196
4.121.1.35 nota_media_general_mi_nota()	197

4.121.1.36 numero_alumnos_matriculados_universidad()	197
4.121.1.37 resumen_alumno()	198
4.121.1.38 resumen_docente()	199
4.121.1.39 resumen_gestor()	199
4.121.1.40 tasa_abandono_titulacion_gestor()	200
4.121.1.41 tasa_graduacion_titulacion_gestor()	201
4.121.1.42 titulacion_alumnado()	201
4.121.1.43 titulacion_docente()	202
4.121.1.44 titulaciones_universidad_gestor()	203
4.121.1.45 universidad_alumno()	204
4.121.1.46 universidades_docente()	204
4.121.1.47 universidades_gestor()	205
4.122 Referencia del Namespace data.queries_dictionary	206
4.122.1 Documentación de las variables	207
4.122.1.1 queries	207
4.123 Referencia del Namespace layouts	207
4.124 Referencia del Namespace layouts.alumno_layout	207
4.124.1 Documentación de las funciones	207
4.124.1.1 alumno_layout()	207
4.125 Referencia del Namespace layouts.docente_layout	208
4.125.1 Documentación de las funciones	208
4.125.1.1 docente_layout()	208
4.126 Referencia del Namespace layouts.gestor_layout	208
4.126.1 Documentación de las funciones	208
4.126.1.1 gestor_layout()	209
4.127 Referencia del Namespace model	209
4.127.1 Documentación de las variables	210
4.127.1.1 best_estimators	210
4.127.1.2 columnas_categoricas	210
4.127.1.3 columnas_numericas	210
4.127.1.4 cross_val_scores_voting	210
4.127.1.5 current_year	211
4.127.1.6 data	211
4.127.1.7 desc	211
4.127.1.8 grid_search	211
4.127.1.9 id_test	211
4.127.1.10 id_train	211
4.127.1.11 ids	212
4.127.1.12 joblib_file	212
4.127.1.13 mejores_estimadores	212
4.127.1.14 mejores_modelos	212
4.127.1.15 metrics	212

4.127.1.16 modelos	212
4.127.1.17 param_grids	213
4.127.1.18 poly	213
4.127.1.19 preprocessor	213
4.127.1.20 random_state	214
4.127.1.21 smote	214
4.127.1.22 stratify	214
4.127.1.23 test_size	214
4.127.1.24 total	214
4.127.1.25 voting_clf	214
4.127.1.26 X	215
4.127.1.27 X_test	215
4.127.1.28 X_train	215
4.127.1.29 y	215
4.127.1.30 y_pred	215
4.127.1.31 y_pred_voting	215
4.127.1.32 y_test	216
4.127.1.33 y_train	216
4.128 Referencia del Namespace util	216
4.128.1 Documentación de las funciones	216
4.128.1.1 list_to_tuple()	216
4.128.1.2 load_data_for_model()	218
4.128.1.3 load_model()	219
4.128.1.4 random_color()	221
4.128.2 Documentación de las variables	221
4.128.2.1 config_mode_bar_buttons_gestor	221
5 Documentación de las clases	223
5.1 Referencia de la Clase data.db_connector.DatabaseConnector	223
5.1.1 Descripción detallada	223
5.1.2 Documentación del constructor y destructor	223
5.1.2.1 __init__()	224
5.1.3 Documentación de las funciones miembro	224
5.1.3.1 close()	224
5.1.3.2 execute_query()	224
5.1.4 Documentación de los datos miembro	225
5.1.4.1 db_url	225
5.1.4.2 engine	225
6 Documentación de archivos	227
6.1 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/app.py	227
6.2 app.py	227

6.3 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/_init__.py	228
6.4 __init__.py	228
6.5 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/_init__.py	228
6.6 __init__.py	228
6.7 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/filters/_init__.py	228
6.8 __init__.py	228
6.9 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/graphs/_init__.py	229
6.10 __init__.py	229
6.11 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/graphs/general/_init__.py	229
6.12 __init__.py	229
6.13 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/graphs/personal/_init__.py	229
6.14 __init__.py	229
6.15 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/utils/_init__.py	229
6.16 __init__.py	229
6.17 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/common/_init__.py	229
6.18 __init__.py	229
6.19 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/docente/_init__.py	230
6.20 __init__.py	230
6.21 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/docente/filters/_init__.py	230
6.22 __init__.py	230
6.23 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/docente/graphs/_init__.py	230
6.24 __init__.py	230
6.25 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/docente/graphs/general/_init__.py	230
6.26 __init__.py	230
6.27 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/docente/graphs/personal/_init__.py	230
6.28 __init__.py	230
6.29 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/docente/utils/_init__.py	231
6.30 __init__.py	231
6.31 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/_init__.py	231
6.32 __init__.py	231

6.33 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/filters/__init__.py	231
6.34 __init__.py	231
6.35 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/__init__.py	231
6.36 __init__.py	231
6.37 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/indicadores/__init__.py	231
6.38 __init__.py	231
6.39 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/resultados/__init__.py	232
6.40 __init__.py	232
6.41 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/riesgo_abandono/__init__.py	232
6.42 __init__.py	232
6.43 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/utils/__init__.py	232
6.44 __init__.py	232
6.45 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/__init__.py	232
6.46 __init__.py	232
6.47 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/alumnado/__init__.py	232
6.48 __init__.py	232
6.49 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/alumnado/filters/__init__.py	233
6.50 __init__.py	233
6.51 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/alumnado/graphs/__init__.py	233
6.52 __init__.py	233
6.53 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/alumnado/utils/__init__.py	233
6.54 __init__.py	233
6.55 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/common/__init__.py	233
6.56 __init__.py	233
6.57 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/docente/__init__.py	233
6.58 __init__.py	233
6.59 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/docente/filters/__init__.py	234
6.60 __init__.py	234
6.61 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↔ TFG/src/components/docente/graphs/__init__.py	234
6.62 __init__.py	234

6.63 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/components/docente/utils/__init__.py	234
6.64 __init__.py	234
6.65 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/components/gestor/__init__.py	234
6.66 __init__.py	234
6.67 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/components/gestor/filters/__init__.py	234
6.68 __init__.py	234
6.69 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/components/gestor/graphs/__init__.py	235
6.70 __init__.py	235
6.71 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/components/gestor/utils/__init__.py	235
6.72 __init__.py	235
6.73 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/data/__init__.py	235
6.74 __init__.py	235
6.75 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/layouts/__init__.py	235
6.76 __init__.py	235
6.77 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/filters/callback_filter_asignaturas_matri_alumnado.py	235
6.78 callback_filter_asignaturas_matri_alumnado.py	236
6.79 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/filters/callback_filter_curso_cademico_alumnado.py	237
6.80 callback_filter_curso_cademico_alumnado.py	237
6.81 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/filters/callback_filter_titulacion_alumnado.py	238
6.82 callback_filter_titulacion_alumnado.py	238
6.83 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/graphs/general/callback_graph_asig_superadas_media.py	239
6.84 callback_graph_asig_superadas_media.py	239
6.85 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/graphs/general/callback_graph_calif_cual_comparativa.py	241
6.86 callback_graph_calif_cual_comparativa.py	241
6.87 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/graphs/general/callback_graph_calif_media_mi_nota.py	243
6.88 callback_graph_calif_media_mi_nota.py	243
6.89 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/graphs/personal/callback_graph_calif_cualitativa_alumnado.py	245
6.90 callback_graph_calif_cualitativa_alumnado.py	245
6.91 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/graphs/personal/callback_graph_calif_numerica_alumnado.py	246
6.92 callback_graph_calif_numerica_alumnado.py	247

6.93 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/alumnado/graphs/personal/callback_graph_progreso_academico_alumnado.py .	248
6.94 callback_graph_progreso_academico_alumnado.py	248
6.95 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/alumnado/graphs/personal/callback_graph_tasa_rendimiento_alumnado.py . . .	249
6.96 callback_graph_tasa_rendimiento_alumnado.py	250
6.97 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/alumnado/utils/callback_resumen_alumnado.py	251
6.98 callback_resumen_alumnado.py	251
6.99 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/alumnado/utils/callback_select_alumnado.py	252
6.100 callback_select_alumnado.py	252
6.101 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/alumnado/utils/callback_tabs_alumnado.py	253
6.102 callback_tabs_alumnado.py	253
6.103 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/common/callback_sidebarCollapse.py	255
6.104 callback_sidebarCollapse.py	255
6.105 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/common/callback_update_layout.py	255
6.106 callback_update_layout.py	256
6.107 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/common/callback_user_role.py	256
6.108 callback_user_role.py	257
6.109 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/docente/filters/callback_filter_all_asignaturas_titulacion_docente.py . . .	257
6.110 callback_filter_all_asignaturas_titulacion_docente.py	258
6.111 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/docente/filters/callback_filter_all_curso_academico.py	259
6.112 callback_filter_all_curso_academico.py	259
6.113 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/docente/filters/callback_filter_asignaturas_docente.py	260
6.114 callback_filter_asignaturas_docente.py	260
6.115 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/docente/filters/callback_filter_curso_academico_docente.py	261
6.116 callback_filter_curso_academico_docente.py	261
6.117 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/docente/filters/callback_filter_titulacion_docente.py	262
6.118 callback_filter_titulacion_docente.py	262
6.119 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/docente/graphs/general/callback_graph_all_calif_cualitativa_docente.py . . .	263
6.120 callback_graph_all_calif_cualitativa_docente.py	263
6.121 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← _TFG/src/callbacks/docente/graphs/general/callback_graph_all_calif_media_docente.py	265
6.122 callback_graph_all_calif_media_docente.py	265

6.123 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_genero_docente.py	266
6.124 callback_graph_alu_genero_docente.py	266
6.125 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_media_docente.py	267
6.126 callback_graph_alu_media_docente.py	268
6.127 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_nota_cualitativa_docente.py	269
6.128 callback_graph_alu_nota_cualitativa_docente.py	269
6.129 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_repetidores_nuevos_docente.py	270
6.130 callback_graph_alu_repetidores_nuevos_docente.py	271
6.131 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/docente/utils/callback_resumen_docente.py	272
6.132 callback_resumen_docente.py	272
6.133 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/docente/utils/callback_select_docente.py	273
6.134 callback_select_docente.py	273
6.135 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/docente/utils/callback_tabs_docente.py	274
6.136 callback_tabs_docente.py	274
6.137 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/filters/callback_filter_all_curso_academico_gestor.py	275
6.138 callback_filter_all_curso_academico_gestor.py	276
6.139 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/filters/callback_filter_curso_academico_gestor.py	277
6.140 callback_filter_curso_academico_gestor.py	277
6.141 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/filters/callback_filter_titulaciones_gestor.py	278
6.142 callback_filter_titulaciones_gestor.py	278
6.143 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_egresados_genero_gestor.py	279
6.144 callback_graph_egresados_genero_gestor.py	279
6.145 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_egresados_nacionalidad_gestor.py	282
6.146 callback_graph_egresados_nacionalidad_gestor.py	283
6.147 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_nuevo_ingreso_genero_gestor.py	285
6.148 callback_graph_nuevo_ingreso_genero_gestor.py	286
6.149 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_nuevo_ingreso_nacionalidad_gestor.py	288
6.150 callback_graph_nuevo_ingreso_nacionalidad_gestor.py	289
6.151 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/resultados/callback_graph_duracion_media_estudios_nota_gestor.py	291
6.152 callback_graph_duracion_media_estudios_nota_gestor.py	292

6.153 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/resultados/callback_graph_nota_acceso_titulacion_gestor.py . . .	294
6.154 callback_graph_nota_acceso_titulacion_gestor.py	295
6.155 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/riesgo_abandono/callback_graph_tasa_abandono_gestor.py . . .	297
6.156 callback_graph_tasa_abandono_gestor.py	298
6.157 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/graphs/riesgo_abandono/callback_graph_tasa_graduacion_gestor.py . . .	300
6.158 callback_graph_tasa_graduacion_gestor.py	301
6.159 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/utils/callback_resumen_gestor.py	303
6.160 callback_resumen_gestor.py	303
6.161 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/utils/callback_select_gestor.py	304
6.162 callback_select_gestor.py	305
6.163 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/callbacks/gestor/utils/callback_tabs_gestor.py	305
6.164 callback_tabs_gestor.py	306
6.165 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/filters/filter_asignaturas_matri_alumnado.py	307
6.166 filter_asignaturas_matri_alumnado.py	307
6.167 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/filters/filter_curso_academico_alumnado.py	308
6.168 filter_curso_academico_alumnado.py	308
6.169 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/filters/filter_titulacion_alumnado.py	309
6.170 filter_titulacion_alumnado.py	309
6.171 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/graphs/graphs_general_alumnado.py	310
6.172 graphs_general_alumnado.py	310
6.173 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/graphs/graphs_personal_alumnado.py	311
6.174 graphs_personal_alumnado.py	311
6.175 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/utils/recomendador_alumnado.py	312
6.176 recomendador_alumnado.py	312
6.177 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/utils/resumen_alumnado.py	313
6.178 resumen_alumnado.py	313
6.179 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/utils/select_alumnado.py	314
6.180 select_alumnado.py	314
6.181 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/alumnado/utils/tabs_alumnado.py	314
6.182 tabs_alumnado.py	315

6.183 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/common/create_graph.py	315
6.184 create_graph.py	316
6.185 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/common/create_graph_with_table.py	316
6.186 create_graph_with_table.py	316
6.187 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/common/filters.py	317
6.188 filters.py	317
6.189 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/common/footer.py	318
6.190 footer.py	318
6.191 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/common/header.py	318
6.192 header.py	319
6.193 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/common/modal_data.py	319
6.194 modal_data.py	320
6.195 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/common/sidebar.py	320
6.196 sidebar.py	320
6.197 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/filters/filter_all_asignaturas_titulacion_docente.py	321
6.198 filter_all_asignaturas_titulacion_docente.py	321
6.199 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/filters/filter_all_curso_academico.py	322
6.200 filter_all_curso_academico.py	322
6.201 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/filters/filter_asignaturas_docente.py	323
6.202 filter_asignaturas_docente.py	323
6.203 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/filters/filter_curso_academico_docente.py	324
6.204 filter_curso_academico_docente.py	324
6.205 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/filters/filter_titulacion_docente.py	325
6.206 filter_titulacion_docente.py	325
6.207 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/graphs/graphs_general_docente.py	326
6.208 graphs_general_docente.py	326
6.209 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/graphs/graphs_personal_docente.py	327
6.210 graphs_personal_docente.py	327
6.211 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/utils/recomendador_docente.py	328
6.212 recomendador_docente.py	328

6.213 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/utils/resumen_docente.py	329
6.214 resumen_docente.py	329
6.215 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/utils/select_docente.py	329
6.216 select_docente.py	330
6.217 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/docente/utils/tabs_docente.py	330
6.218 tabs_docente.py	330
6.219 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/filters/filter_all_curso_academico_gestor.py	331
6.220 filter_all_curso_academico_gestor.py	331
6.221 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/filters/filter_curso_academico_gestor.py	332
6.222 filter_curso_academico_gestor.py	332
6.223 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/filters/filter_titulaciones_gestor.py	333
6.224 filter_titulaciones_gestor.py	333
6.225 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/graphs/graphs_indicadores_gestor.py	334
6.226 graphs_indicadores_gestor.py	334
6.227 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/graphs/graphs_resultados_gestor.py	336
6.228 graphs_resultados_gestor.py	336
6.229 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/graphs/graphs_riesgo_abandono_gestor.py	337
6.230 graphs_riesgo_abandono_gestor.py	337
6.231 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/utils/recomendador_gestor.py	338
6.232 recomendador_gestor.py	338
6.233 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/utils/resumen_gestor.py	339
6.234 resumen_gestor.py	339
6.235 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/utils/select_gestor.py	340
6.236 select_gestor.py	340
6.237 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/components/gestor/utils/tabs_gestor.py	340
6.238 tabs_gestor.py	341
6.239 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/data/db_connector.py	341
6.240 db_connector.py	342
6.241 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/data/generate_synthetic_data.py	343
6.242 generate_synthetic_data.py	344

6.243 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/data/queries.py	349
6.244 queries.py	350
6.245 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/data/queries_dictionary.py	357
6.246 queries_dictionary.py	357
6.247 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/layouts/alumno_layout.py	366
6.248 alumno_layout.py	366
6.249 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/layouts/docente_layout.py	366
6.250 docente_layout.py	367
6.251 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/layouts/gestor_layout.py	367
6.252 gestor_layout.py	367
6.253 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/model.py	368
6.254 model.py	369
6.255 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities←_TFG/src/util.py	371
6.256 util.py	371
Índice alfabético	375

Capítulo 1

Indice de namespaces

1.1. Lista de 'namespaces'

Lista de los 'namespaces', con una breve descripción:

app	13
callbacks	14
callbacks.alumnado	14
callbacks.alumnado.filters	15
callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado	15
callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado	16
callbacks.alumnado.filters.callback_filter_titulacion_alumnado	17
callbacks.alumnado.graphs	19
callbacks.alumnado.graphs.general	19
callbacks.alumnado.graphs.general.callback_graph_asig_superadas_media	19
callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa	21
callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_nota	23
callbacks.alumnado.graphs.personal	25
callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado	25
callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_alumnado	27
callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado	29
callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado	30
callbacks.alumnado.utils	31
callbacks.alumnado.utils.callback_resumen_alumnado	32
callbacks.alumnado.utils.callback_select_alumnado	34
callbacks.alumnado.utils.callback_tabs_alumnado	35
callbacks.common	36
callbacks.common.callback_sidebarCollapse	36
callbacks.common.callback_update_layout	37
callbacks.common.callback_user_role	38
callbacks.docente	40
callbacks.docente.filters	40
callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente	40
callbacks.docente.filters.callback_filter_all_curso_academico	42
callbacks.docente.filters.callback_filter_asignaturas_docente	43
callbacks.docente.filters.callback_filter_curso_academico_docente	44
callbacks.docente.filters.callback_filter_titulacion_docente	45
callbacks.docente.graphs	46
callbacks.docente.graphs.general	46
callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente	46

callbacks.docente.graphs.general.callback_graph_all_calif_media_docente	48
callbacks.docente.graphs.personal	49
callbacks.docente.graphs.personal.callback_graph_alu_genero_docente	50
callbacks.docente.graphs.personal.callback_graph_alu_media_docente	51
callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente	52
callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente	54
callbacks.docente.utils	56
callbacks.docente.utils.callback_resumen_docente	56
callbacks.docente.utils.callback_select_docente	57
callbacks.docente.utils.callback_tabs_docente	58
callbacks.gestor	60
callbacks.gestor.filters	60
callbacks.gestor.filters.callback_filter_all_curso_academico_gestor	61
callbacks.gestor.filters.callback_filter_curso_academico_gestor	62
callbacks.gestor.filters.callback_filter_titulaciones_gestor	63
callbacks.gestor.graphs	65
callbacks.gestor.graphs.indicadores	65
callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor	65
callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor	71
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor	76
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor	82
callbacks.gestor.graphs.resultados	87
callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor	88
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor	93
callbacks.gestor.graphs.riesgo_abandono	98
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor	98
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor	104
callbacks.gestor.utils	109
callbacks.gestor.utils.callback_resumen_gestor	109
callbacks.gestor.utils.callback_select_gestor	111
callbacks.gestor.utils.callback_tabs_gestor	112
components	114
components.alumnado	114
components.alumnado.filters	114
components.alumnado.filters.filter_asignaturas_matri_alumnado	115
components.alumnado.filters.filter_curso_academico_alumnado	116
components.alumnado.filters.filter_titulacion_alumnado	117
components.alumnado.graphs	118
components.alumnado.graphs.graphs_general_alumnado	118
components.alumnado.graphs.graphs_personal_alumnado	119
components.alumnado.utils	120
components.alumnado.utils.recomendador_alumnado	120
components.alumnado.utils.resumen_alumnado	122
components.alumnado.utils.select_alumnado	122
components.alumnado.utils.tabs_alumnado	123
components.common	124
components.common.create_graph	124
components.common.create_graph_with_table	125
components.common.filters	127
components.common.footer	128
components.common.header	128
components.common.modal_data	129
components.common.sidebar	130
components.docente	131
components.docente.filters	132
components.docente.filters.filter_all_asignaturas_titulacion_docente	132
components.docente.filters.filter_all_curso_academico	133
components.docente.filters.filter_asignaturas_docente	134

components.docente.filters.filter_curso_academico_docente	135
components.docente.filters.filter_titulacion_docente	136
components.docente.graphs	137
components.docente.graphs.graphs_general_docente	137
components.docente.graphs.graphs_personal_docente	138
components.docente.utils	140
components.docente.utils.recomendador_docente	140
components.docente.utils.resumen_docente	141
components.docente.utils.select_docente	142
components.docente.utils.tabs_docente	143
components.gestor	143
components.gestor.filters	144
components.gestor.filters.filter_all_curso_academico_gestor	144
components.gestor.filters.filter_curso_academico_gestor	145
components.gestor.filters.filter_titulaciones_gestor	146
components.gestor.graphs	147
components.gestor.graphs.graphs_indicadores_gestor	147
components.gestor.graphs.graphs_resultados_gestor	149
components.gestor.graphs.graphs_riesgo_abandono_gestor	150
components.gestor.utils	152
components.gestor.utils.recomendador_gestor	152
components.gestor.utils.resumen_gestor	153
components.gestor.utils.select_gestor	154
components.gestor.utils.tabs_gestor	155
data	156
data.db_connector	156
data.generate_synthetic_data	157
data.queries	168
data.queries_dictionary	206
layouts	207
layouts.alumno_layout	207
layouts.docente_layout	208
layouts.gestor_layout	208
model	209
util	216

Capítulo 2

Índice de clases

2.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

data.db_connector.DatabaseConnector	223
---	-----

Capítulo 3

Índice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

```
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/app.py . . . . . 227
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/model.py . . . . . 368
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/util.py . . . . . 371
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/__init__.py 228
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/__init__.py
                                              228
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/__init__.py
                                              228
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/callback_filter_asign
                                              235
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/callback_filter_curs
                                              237
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/callback_filter_titula
                                              238
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/__init__.py
                                              229
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/__init__.py
                                              229
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/callback_c
                                              239
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/callback_c
                                              241
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/callback_c
                                              243
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/__init__.py
                                              229
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_
                                              245
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_
                                              246
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_
                                              248
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_
                                              249
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/__init__.py
                                              229
```

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/[callback_resumen_alumnado](#)
251
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/[callback_select_alumno](#)
252
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/[callback_tabs_alumno](#)
253
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/[__init__.py](#)
229
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/[callback_sidebarCollapse](#)
255
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/[callback_update_layout.py](#)
255
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/[callback_userRole.py](#)
256
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/[__init__.py](#)
230
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/[__init__.py](#)
230
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/[callback_filter_all_asignacion](#)
257
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/[callback_filter_all_curso](#)
259
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/[callback_filter_asignacion](#)
260
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/[callback_filter_curso](#)
261
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/[callback_filter_titulacion](#)
262
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/[__init__.py](#)
230
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/general/[__init__.py](#)
230
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/general/[callback_grado](#)
263
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/general/[callback_gradoAlumno](#)
265
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/[__init__.py](#)
230
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/[callback_gradoAlumno](#)
266
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/[callback_gradoAlumno](#)
267
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/[callback_gradoAlumno](#)
269
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/[callback_gradoAlumno](#)
270
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/[__init__.py](#)
231
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/[callback_resumen_docente](#)
272
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/[callback_select_docente](#)
273
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/[callback_tabs_docente](#)
274
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/[__init__.py](#)
231
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/[__init__.py](#)
231

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/[callback_filter_all_cursos](#) .py
275
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/[callback_filter_curso_academico](#) .py
277
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/[callback_filter_titulacion](#) .py
278
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/[__init__](#).py
231
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/[__init__](#).py
231
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/[callback_tasa_abandono](#) .py
279
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/[callback_tasa_graduacion](#) .py
282
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/[callback_titulacion](#) .py
285
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/[callback_titulacion](#) .py
288
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/[__init__](#).py
232
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/[callback_graficas](#) .py
291
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/[callback_graficas](#) .py
294
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo←_abandono/[__init__](#).py
232
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo←_abandono/[callback_graph_tasa_abandono_gestor](#).py
297
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo←_abandono/[callback_graph_tasa_graduacion_gestor](#).py
300
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/[__init__](#).py
232
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/[callback_resumen_gestor](#) .py
303
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/[callback_select_gestor](#).py
304
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/[callback_tabs_gestor](#).py
305
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/[__init__](#).py
232
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/[__init__](#).py
232
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/[__init__](#).py
233
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/[filter_asignaturas](#) .py
307
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/[filter_curso_academico](#) .py
308
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/[filter_titulacion_academica](#) .py
309
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/graphs/[__init__](#).py
233
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/graphs/[graphs_generales](#) .py
310
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/graphs/[graphs_personas](#) .py
311
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/[__init__](#).py
233

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/recomendador_alumnado.py
312
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/resumen_alumnado.py
313
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/select_alumnado.py
314
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/tabs_alumnado.py
314
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/__init__.py
233
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/create_graph.py
315
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/create_graph_with_table.py
316
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/filters.py
317
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/footer.py
318
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/header.py
318
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/modal_data.py
319
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/common/sidebar.py
320
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/__init__.py
233
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/__init__.py
234
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_all_asignaturas.py
321
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_all_curso_academico.py
322
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_asignaturas.py
323
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_curso_academico.py
324
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_titulacion_docente.py
325
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/graphs/__init__.py
234
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/graphs/graphs_general.py
326
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/graphs/graphs_personal.py
327
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/__init__.py
234
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/recomendador_docente.py
328
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/resumen_docente.py
329
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/select_docente.py
329
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/tabs_docente.py
330
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/__init__.py
234
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/__init__.py
234

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_all_curso_academico.py . . . 331
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_curso_academico.py . . . 332
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_titulaciones_gerenciales.py . . . 333
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/__init__.py . . . 235
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_indicadores.py . . . 334
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_resultados.py . . . 336
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_riesgo_abierto.py . . . 337
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/__init__.py . . . 235
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/recomendador_gestor.py . . . 338
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/resumen_gestor.py . . . 339
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/select_gestor.py . . . 340
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/tabs_gestor.py . . . 340
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/__init__.py . . . 235
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/db_connector.py . . . 341
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/generate_synthetic_data.py . . . 343
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/queries.py . . . 349
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/queries_dictionary.py . . . 357
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/layouts/__init__.py . . . 235
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/layouts/alumno_layout.py . . . 366
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/layouts/docente_layout.py . . . 366
C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/layouts/gestor_layout.py . . . 367

Capítulo 4

Documentación de namespaces

4.1. Referencia del Namespace app

Variables

- `app`
- `title`
- `_favicon`
- `server = app.server`
- `layout`
- `debug`

4.1.1. Documentación de las variables

4.1.1.1. `_favicon`

```
app._favicon [private]
```

Definición en la línea [34](#) del archivo [app.py](#).

4.1.1.2. `app`

```
app.app
```

Valor inicial:

```
00001 = dash.Dash(  
00002     __name__,  
00003     external_stylesheets=[  
00004         'src/assets/css/styles.css',  
00005         dbc.themes.BOOTSTRAP  
00006     ],  
00007     suppress_callback_exceptions=True  
00008 )
```

Definición en la línea [24](#) del archivo [app.py](#).

4.1.1.3. debug

app.debug

Definición en la línea 48 del archivo [app.py](#).

4.1.1.4. layout

app.layout

Definición en la línea 37 del archivo [app.py](#).

4.1.1.5. server

app.server = app.server

Definición en la línea 35 del archivo [app.py](#).

4.1.1.6. title

app.title

Definición en la línea 33 del archivo [app.py](#).

4.2. Referencia del Namespace callbacks

Namespaces

- namespace [alumnado](#)
- namespace [common](#)
- namespace [docente](#)
- namespace [gestor](#)

4.3. Referencia del Namespace callbacks.alumnado

Namespaces

- namespace [filters](#)
- namespace [graphs](#)
- namespace [utils](#)

4.4. Referencia del Namespace callbacks.alumnado.filters

Namespaces

- namespace [callback_filter_asignaturas_matri_alumnado](#)
- namespace [callback_filter_curso_academico_alumnado](#)
- namespace [callback_filter_titulacion_alumnado](#)

4.5. Referencia del Namespace callbacks.alumnado.filters.callback_← filter_asignaturas_matri_alumnado

Funciones

- def [update_filter_asignaturas_matri_alumnado](#) (alumno_id, curso_academico, titulacion, n_clicks, existing_options)

4.5.1. Documentación de las funciones

4.5.1.1. update_filter_asignaturas_matri_alumnado()

```
def callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado.update_filter_←
asignaturas_matri_alumnado (
    alumno_id,
    curso_academico,
    titulacion,
    n_clicks,
    existing_options )
```

Actualiza las opciones del dropdown de asignaturas matriculadas por el alumno y gestiona el evento del botón "Seleccionar todo".

Args:

alumno_id (str): Identificador del alumno.
 curso_academico (list): Lista con los cursos académicos
 titulacion (str): Titulación seleccionada
 n_clicks (int): Número de clicks en el botón "Seleccionar todo"
 existing_options (list): Opciones actuales del dropdown

Returns:

list: Opciones del dropdown
 list: Valor seleccionado

Definición en la línea 24 del archivo [callback_filter_asignaturas_matri_alumnado.py](#).

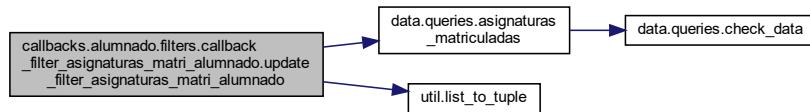
```
00024 def update_filter_asignaturas_matri_alumnado(alumno_id, curso_academico, titulacion, n_clicks,
00025     existing_options):
00026     """
00027     Actualiza las opciones del dropdown de asignaturas matriculadas por el alumno y
00028     gestiona el evento del botón "Seleccionar todo".
00029     Args:
00030         alumno_id (str): Identificador del alumno.
00031         curso_academico (list): Lista con los cursos académicos
00032         titulacion (str): Titulación seleccionada
```

```

00033     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00034     existing_options (list): Opciones actuales del dropdown
00035
00036     Returns:
00037         list: Opciones del dropdown
00038         list: Valor seleccionado
00039     """
00040
00041 ctx = callback_context
00042 trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None
00043
00044     # Evento de selección de todas las asignaturas matriculadas
00045     if trigger_id == 'select-all-button' and n_clicks > 0:
00046         return existing_options, [option['value'] for option in existing_options]
00047
00048     if not (alumno_id and curso_academico and titulacion):
00049         return [], None
00050
00051     try:
00052         curso_academico = list_to_tuple(curso_academico)
00053     except Exception as e:
00054         return [], None
00055
00056     data = asignaturas_matriculadas(alumno_id, curso_academico, titulacion)
00057
00058     if not data:
00059         return [], None
00060
00061     opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
00062     value = [option['value'] for option in opciones_dropdown]
00063
00064     return opciones_dropdown, value

```

Gráfico de llamadas para esta función:



4.6. Referencia del Namespace callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado

Funciones

- def `update_filter_curso_academico_alumnado` (alumno_id, titulacion, n_clicks, existing_options)

4.6.1. Documentación de las funciones

4.6.1.1. update_filter_curso_academico_alumnado()

```

def callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado.update_filter_curso_academico_alumnado (
    alumno_id,
    titulacion,
    n_clicks,
    existing_options )

```

Actualiza las opciones del dropdown de cursos académicos y gestiona el evento del botón "Seleccionar todo".

Args:

- alumno_id (str): Identificador del alumno.
- titulacion (str): Titulación seleccionada
- n_clicks (int): Número de clicks en el botón "Seleccionar todo"
- existing_options (list): Opciones actuales del dropdown

Returns:

- list: Opciones del dropdown
- list: Valor seleccionado

Definición en la línea 23 del archivo [callback_filter_curso_cademico_alumnado.py](#).

```
00023 def update_filter_curso_academico_alumnado(alumno_id, titulacion, n_clicks, existing_options):
00024     """
00025 Actualiza las opciones del dropdown de cursos académicos y
00026 gestiona el evento del botón "Seleccionar todo".
00027
00028     Args:
00029         alumno_id (str): Identificador del alumno.
00030         titulacion (str): Titulación seleccionada
00031         n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00032         existing_options (list): Opciones actuales del dropdown
00033
00034     Returns:
00035         list: Opciones del dropdown
00036         list: Valor seleccionado
00037     """
00038
00039 ctx = callback_context
00040 trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None
00041
00042     # Evento de selección de todos los cursos académicos
00043 if trigger_id == 'select-all-cursos-academicos' and n_clicks > 0:
00044     return existing_options, [option['value'] for option in existing_options]
00045
00046 if not (alumno_id and titulacion):
00047     return [], None
00048
00049     # Obtener los cursos académicos desde la base de datos
00050 data = curso_academico_alumnado(alumno_id, titulacion)
00051
00052 if not data:
00053     return [], None
00054
00055 opciones_dropdown = [{'label': curso[0], 'value': curso[0]} for curso in data]
00056 value = [option['value'] for option in opciones_dropdown]
00057
00058 return opciones_dropdown, value
00059
```

Gráfico de llamadas para esta función:



4.7. Referencia del Namespace callbacks.alumnado.filters.callback_filter_titulacion_alumnado

Funciones

- [def update_filter_titulacion_alumnado \(alumno_id, selected_value, stored_titulacion\)](#)

4.7.1. Documentación de las funciones

4.7.1.1. update_filter_titulacion_alumnado()

```
def callbacks.alumnado.filters.callback_filter_titulacion_alumnado.update_filter_titulacion_←
alumnado (
    alumno_id,
    selected_value,
    stored_titulacion )
```

Actualiza las opciones del dropdown de titulaciones del perfil "Alumno".

Args:

- alumno_id (str): Identificador del alumno.
- selected_value (str): Valor seleccionado en el dropdown
- stored_titulacion (str): Titulación almacenada

Returns:

- list: Opciones del dropdown
- str: Valor seleccionado

Definición en la línea 22 del archivo [callback_filter_titulacion_alumnado.py](#).

```
00022 def update_filter_titulacion_alumnado(alumno_id, selected_value, stored_titulacion):
00023     """
00024     Actualiza las opciones del dropdown de titulaciones del perfil "Alumno".
00025
00026     Args:
00027         alumno_id (str): Identificador del alumno.
00028         selected_value (str): Valor seleccionado en el dropdown
00029         stored_titulacion (str): Titulación almacenada
00030
00031     Returns:
00032         list: Opciones del dropdown
00033         str: Valor seleccionado
00034     """
00035 if not alumno_id:
00036     return [], None, None
00037
00038     data = titulacion_alumnado(alumno_id)
00039
00040     if not data:
00041         return [], None, None
00042
00043     opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
00044
00045     # Si no hay valor seleccionado pero hay un valor almacenado, lo usamos si está en las opciones
00046     if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
00047         return opciones_dropdown, stored_titulacion, stored_titulacion
00048
00049     return opciones_dropdown, selected_value, selected_value
```

Gráfico de llamadas para esta función:



4.8. Referencia del Namespace callbacks.alumnado.graphs

Namespaces

- namespace [general](#)
- namespace [personal](#)

4.9. Referencia del Namespace callbacks.alumnado.graphs.general

Namespaces

- namespace [callback_graph_asig_superadas_media](#)
- namespace [callback_graph_calif_cual_comparativa](#)
- namespace [callback_graph_calif_media_mi_nota](#)

4.10. Referencia del Namespace callbacks.alumnado.graphs.general. callback_graph_asig_superadas_media

Funciones

- def [update_graph_alumnado](#) (curso_academico, alumno_id, asignaturas_matriculadas, titulacion)

4.10.1. Documentación de las funciones

4.10.1.1. update_graph_alumnado()

```
def callbacks.alumnado.graphs.general.callback_graph_asig_superadas_media.update_graph_←  
alumnado (←  
    curso_academico,  
    alumno_id,  
    asignaturas_matriculadas,  
    titulacion )
```

Actualiza el gráfico de relación entre la nota media y el número de asignaturas superadas por alumno del perfil "Alumno" de la pestaña "Rendimiento académico general".

Args:

 curso_academico (list): Lista con los cursos académicos
 alumno_id (str): Identificador del alumno
 asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
 titulacion (str): Titulación seleccionada

Returns:

 go.Figure: Figura con el gráfico

Definición en la línea 27 del archivo [callback_graph_asig_superadas_media.py](#).

```

00027 def update_graph_alumnado(curso_academico, alumno_id, asignaturas_matriculadas, titulacion):
00028 """
00029 Actualiza el gráfico de relación entre la nota media y el número de asignaturas superadas
00030 por alumno del perfil "Alumno" de la pestaña "Rendimiento académico general".
00031
00032     Args:
00033         curso_academico (list): Lista con los cursos académicos
00034         alumno_id (str): Identificador del alumno
00035         asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00036         titulacion (str): Titulación seleccionada
00037
00038     Returns:
00039         go.Figure: Figura con el gráfico
00040
00041 """
00042 fig = go.Figure()
00043
00044 fig.update_layout(
00045     title={
00046         "text": "Relación nota media y número de asignaturas superadas por alumno",
00047             "x": 0.5,
00048     },
00049     xaxis_title="Nota Media",
00050     yaxis_title="Nº Asignaturas superadas",
00051     legend_title="Estado del alumno",
00052     showlegend=True,
00053     xaxis=dict(range=[0, 10]),
00054     yaxis=dict(range=[0, 40]),
00055 )
00056
00057 if not (curso_academico and alumno_id and asignaturas_matriculadas and titulacion):
00058     return fig
00059
00060 try:
00061     curso_academico = list_to_tuple(curso_academico)
00062     asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00063 except Exception as e:
00064     print("Error:", e)
00065     return fig
00066
00067 data_universidad = universidad_alumno(alumno_id)
00068
00069 if not data_universidad:
00070     return fig
00071
00072 data = asignaturas_superadas_media_abandono(
00073     curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
00074 )
00075
00076 if not data:
00077     return fig
00078
00079 df = pd.DataFrame(
00080     data, columns=["Alumno_id", "Abandono", "Nota_Media", "Asignaturas_Superadas"]
00081 )
00082
00083 df["Abandono"] = (
00084     df["Abandono"]
00085     .str.strip()
00086     .str.lower()
00087     .replace({"si": "Abandona", "no": "No abandona"})
00088 )
00089 df["Personal"] = df["Alumno_id"].apply(lambda x: " (Yo)" if x == alumno_id else "")
00090 df["Key"] = df["Abandono"] + df["Personal"]
00091
00092 colors = {
00093     "Abandona": "red",
00094     "No abandona": "blue",
00095     "Abandona (Yo)": "yellow",
00096     "No abandona (Yo)": "yellow",
00097 }
00098
00099 for key, group in df.groupby("Key"):
00100     fig.add_trace(
00101         go.Scatter(
00102             x=group["Nota_Media"],
00103             y=group["Asignaturas_Superadas"],
00104             mode="markers",
00105             name=key,
00106             marker=dict(
00107                 size=12,
00108                 line=dict(width=2 if " (Yo)" in key else 1),
00109                 color=colors[key],
00110             ),
00111             opacity=1.0 if " (Yo)" in key else 0.8,
00112         )

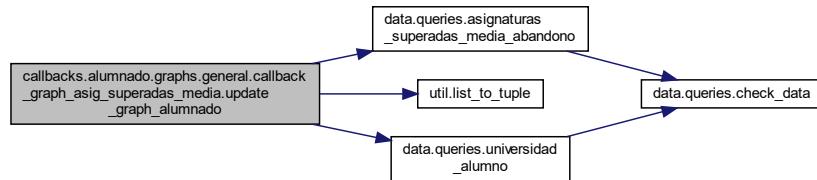
```

```

00113      )
00114
00115     return fig

```

Gráfico de llamadas para esta función:



4.11. Referencia del Namespace callbacks.alumnado.graphs.general.← callback_graph_calif_cual_comparativa

Funciones

- def [update_graph_alumnado](#) (curso_academico, asignaturas_matriculadas, alumno_id, titulacion)

4.11.1. Documentación de las funciones

4.11.1.1. update_graph_alumnado()

```
def callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa.update_graph_←
alumnado (
    curso_academico,
    asignaturas_matriculadas,
    alumno_id,
    titulacion )
```

Actualiza el gráfico de calificaciones cualitativas general por curso académico del perfil "Alumno" de la pestaña "Rendimiento académico general".

Args:

curso_academico (list): Lista con los cursos académicos
asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
alumno_id (str): Identificador del alumno
titulacion (str): Titulación seleccionada

Returns:

go.Figure: Figura con el gráfico

Definición en la línea 31 del archivo [callback_graph_calif_cual_comparativa.py](#).

```

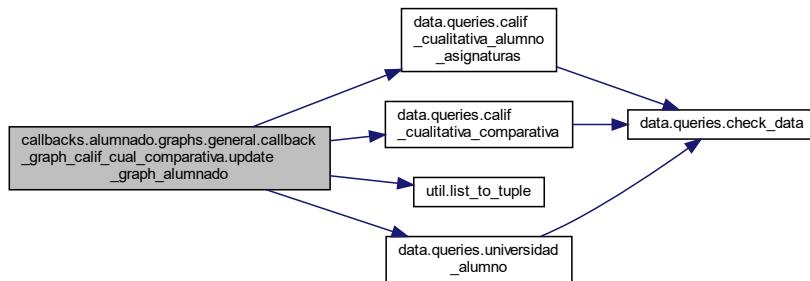
00031 def update_graph_alumnado(curso_academico, asignaturas_matriculadas, alumno_id, titulacion):
00032 """
00033 Actualiza el gráfico de calificaciones cualitativas general por curso académico
00034 del perfil "Alumno" de la pestaña "Rendimiento académico general".
00035
00036     Args:
00037         curso_academico (list): Lista con los cursos académicos
00038         asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00039         alumno_id (str): Identificador del alumno
00040         titulacion (str): Titulación seleccionada
00041
00042     Returns:
00043         go.Figure: Figura con el gráfico
00044 """
00045 fig = go.Figure()
00046
00047 fig.update_layout(
00048     title={
00049         "text": "Calificaciones cualitativas general por curso académico",
00050             "x": 0.5,
00051         },
00052         barmode="stack",
00053         xaxis={"title": "Asignatura", "tickangle": 45},
00054         yaxis={"title": "Nº Alumnos matriculados"},
00055         showlegend=True,
00056         legend={"title": "Calificación"},
00057         height=600,
00058     )
00059
00060     if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
00061         return fig
00062
00063     try:
00064         curso_academico = list_to_tuple(curso_academico)
00065         asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00066     except Exception as e:
00067         print("Error:", e)
00068         return fig
00069
00070     data_universidad = universidad_alumno(alumno_id)
00071
00072     if not data_universidad:
00073         return fig
00074
00075     general_data = calif_cualitativa_comparativa(
00076         curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
00077     )
00078     student_data = calif_cualitativa_alumno_asignaturas(
00079         alumno_id, curso_academico, asignaturas_matriculadas, titulacion
00080     )
00081
00082     if not general_data:
00083         return fig
00084
00085     general_df = pd.DataFrame(
00086         general_data, columns=["Asignatura", "Calificacion", "Numero"]
00087     )
00088     student_df = pd.DataFrame(
00089         student_data, columns=["Asignatura", "Calificacion", "Numero"]
00090     )
00091
00092     categories = ["Suspensos", "No presentado", "Aprobado", "Notable", "Sobresaliente"]
00093
00094     color_mapping = {
00095         "Sobresaliente": "blue",
00096         "Notable": "green",
00097         "Aprobado": "orange",
00098         "Suspensos": "red",
00099         "No presentado": "gray",
00100     }
00101
00102     general_pivot = general_df.pivot_table(
00103         index="Asignatura", columns="Calificacion", values="Numero", fill_value=0
00104     )
00105     general_pivot = general_pivot.reindex(columns=categories, fill_value=0)
00106
00107     for category in categories:
00108         fig.add_trace(
00109             go.Bar(
00110                 x=general_pivot.index,
00111                 y=general_pivot[category],
00112                 name=category,
00113                 marker=dict(color=color_mapping[category]),
00114                 opacity=0.7,
00115             )
00116         )

```

4.12 Referencia del Namespace callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_notas

```
00117     for category in categories:
00118         y = [
00119             1 if (
00120                 (subject in student_df['Asignatura'].values) and
00121                 (student_df[student_df['Asignatura'] == subject]['Calificacion'].values[0] ==
00122                  category)
00123             ) else 0 for subject in general_pivot.index
00124         ]
00125         fig.add_trace(
00126             go.Bar(
00127                 x=general_pivot.index,
00128                 y=y,
00129                 name=f'{category} (Yo)',
00130                 marker=dict(color=color_mapping[category], line=dict(color='black', width=2)),
00131                 opacity=1
00132             )
00133         )
00134     return fig
```

Gráfico de llamadas para esta función:



4.12. Referencia del Namespace callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_notas

Funciones

- def [update_graph_alumnado](#) (curso_academico, asignaturas_matriculadas, alumno_id, titulacion)

4.12.1. Documentación de las funciones

4.12.1.1. update_graph_alumnado()

```
def callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_notas.update_graph_alumnado(
(
    curso_academico,
    asignaturas_matriculadas,
    alumno_id,
    titulacion )
```

Actualiza el gráfico de relación calificaciones del alumno con nota media general del perfil "Alumno" de la pestaña "Rendimiento académico general".

Args:

curso_academico (list): Lista con los cursos académicos
 asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
 alumno_id (str): Identificador del alumno
 titulacion (str): Titulación seleccionada

Returns:

go.Figure: Figura con el gráfico

Definición en la línea 27 del archivo [callback_graph_calif_media_mi_not.py](#).

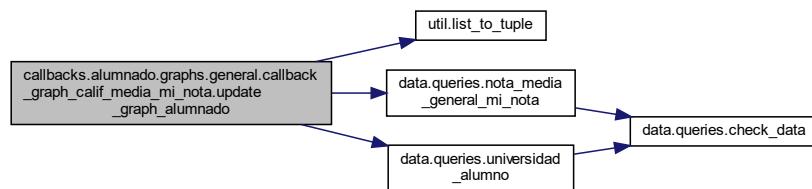
```
00027 def update_graph_alumnado(curso_academico, asignaturas_matriculadas, alumno_id, titulacion):
00028     """
00029     Actualiza el gráfico de relación calificaciones del alumno con nota media general
00030     del perfil "Alumno" de la pestaña "Rendimiento académico general".
00031
00032     Args:
00033         curso_academico (list): Lista con los cursos académicos
00034         asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00035         alumno_id (str): Identificador del alumno
00036         titulacion (str): Titulación seleccionada
00037
00038     Returns:
00039         go.Figure: Figura con el gráfico
00040
00041     """
00042 fig = go.Figure()
00043
00044 fig.update_layout(
00045     title={
00046         "text": "Relación calificaciones del alumno con nota media general",
00047         "x": 0.5,
00048     },
00049     xaxis={"title": "Asignatura", "tickangle": 45},
00050     yaxis={"title": "Nota"},
00051     barmode="group",
00052     height=600,
00053 )
00054
00055     if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
00056         return fig
00057
00058     try:
00059         curso_academico = list_to_tuple(curso_academico)
00060         asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00061     except Exception as e:
00062         print("Error:", e)
00063     return fig
00064
00065     data_universidad = universidad_alumno(alumno_id)
00066
00067     if not data_universidad:
00068         return fig
00069
00070     data = nota_media_general_mi_nota(
00071         curso_academico,
00072         asignaturas_matriculadas,
00073         alumno_id,
00074         titulacion,
00075         data_universidad[0][0],
00076     )
00077
00078     if not data:
00079         return fig
00080
00081     df = pd.DataFrame(data, columns=["Asignatura", "NotaMediaGeneral", "MiNota"])
00082
00083     fig.add_trace(
00084         go.Bar(
00085             x=df["Asignatura"],
00086             y=df["MiNota"],
00087             name="Mi nota",
00088             marker_color="blue",
00089             opacity=0.7,
00090         )
00091     )
00092
00093     fig.add_trace(
00094         go.Bar(
00095             x=df["Asignatura"],
00096             y=df["NotaMediaGeneral"],
```

```

00097         name="Nota media general",
00098         marker_color="grey",
00099         opacity=0.7,
00100     )
00101 )
00102
00103 return fig

```

Gráfico de llamadas para esta función:



4.13. Referencia del Namespace callbacks.alumnado.graphs.personal

Namespaces

- namespace [callback_graph_calif_cualitativa_alumnado](#)
- namespace [callback_graph_calif_numerica_alumnado](#)
- namespace [callback_graph_progreso_academico_alumnado](#)
- namespace [callback_graph_tasa_rendimiento_alumnado](#)

4.14. Referencia del Namespace callbacks.alumnado.graphs.personal.← callback_graph_calif_cualitativa_alumnado

Funciones

- def [update_graph_alumnado](#) (alumno_id, curso_academico, titulacion)

4.14.1. Documentación de las funciones

4.14.1.1. update_graph_alumnado()

```
def callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado.update_←
graph_alumnado (
    alumno_id,
    curso_academico,
    titulacion )
```

Actualiza el gráfico de calificaciones cualitativas de las asignaturas matriculadas del perfil "Alumno" de la pestaña "Expediente académico personal".

Args:

```
    alumno_id (str): Identificador del alumno.
    curso_academico (list): Lista con los cursos académicos
    titulacion (str): Titulación seleccionada
```

Returns:

```
    go.Figure: Figura con el gráfico
```

Definición en la línea 26 del archivo [callback_graph_calif_cualitativa_alumnado.py](#).

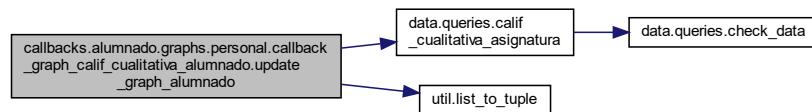
```
00026 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00027     """
00028 Actualiza el gráfico de calificaciones cualitativas de las asignaturas matriculadas
00029 del perfil "Alumno" de la pestaña "Expediente académico personal".
00030
00031     Args:
00032         alumno_id (str): Identificador del alumno.
00033         curso_academico (list): Lista con los cursos académicos
00034         titulacion (str): Titulación seleccionada
00035
00036     Returns:
00037         go.Figure: Figura con el gráfico
00038     """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     title={"text": "Calificaciones cualitativas", "x": 0.5},
00044     barmode="stack",
00045     xaxis={"title": "Curso académico"},
00046     yaxis={"title": "Nº Asignaturas matriculadas"},
00047     showlegend=True,
00048     legend={"title": "Calificación"},
00049 )
00050
00051 if not (alumno_id and curso_academico and titulacion):
00052     return fig
00053
00054 try:
00055     curso_academico = list_to_tuple(curso_academico)
00056 except Exception as e:
00057     print("Error:", e)
00058     return fig
00059
00060 data = calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion)
00061
00062 if not data:
00063     return fig
00064
00065 df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "cantidad"])
00066 df_pivot = df.pivot_table(
00067     index="curso_academico", columns="calificacion", values="cantidad", fill_value=0
00068 )
00069
00070 categories = ["No presentado", "Suspensó", "Aprobado", "Notable", "Sobresaliente"]
00071 for category in categories:
00072     if category not in df_pivot.columns:
00073         df_pivot[category] = 0
00074
00075 df_pivot = df_pivot[categories]
00076 df_pivot = df_pivot.sort_index()
00077
00078 color_mapping = {
00079     "Sobresaliente": "blue",
00080     "Notable": "green",
00081     "Aprobado": "orange",
00082     "Suspensó": "red",
00083     "No presentado": "gray",
```

```

00084     }
00085
00086     for category in categories:
00087         fig.add_trace(
00088             go.Bar(
00089                 x=df_pivot.index,
00090                 y=df_pivot[category],
00091                 name=category,
00092                 marker_color=color_mapping[category],
00093                 opacity=0.7,
00094             )
00095         )
00096
00097     return fig

```

Gráfico de llamadas para esta función:



4.15. Referencia del Namespace callbacks.alumnado.graphs.personal.← callback_graph_calif_numerica_alumnado

Funciones

- def [update_graph_alumnado](#) (alumno_id, curso_academico, titulacion)

4.15.1. Documentación de las funciones

4.15.1.1. update_graph_alumnado()

```

def callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_alumnado.update_graph_←
alumnado (
    alumno_id,
    curso_academico,
    titulacion )

```

Actualiza el gráfico de calificaciones cuantitativas de las asignaturas matriculadas del perfil "Alumno" de la pestaña "Expediente académico personal".

Args:

 alumno_id (str): Identificador del alumno.
 curso_academico (list): Lista con los cursos académicos
 titulacion (str): Titulación seleccionada

Returns:

 go.Figure: Figura con el gráfico

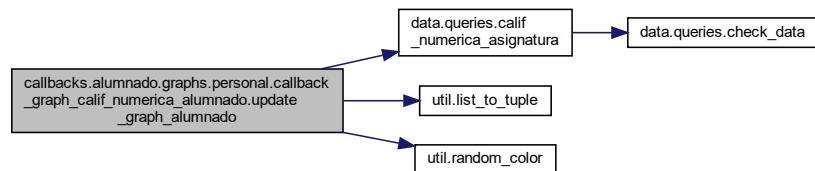
Definición en la línea 26 del archivo [callback_graph_calif_numerica_alumnado.py](#).

```

00026 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00027     """
00028 Actualiza el gráfico de calificaciones cuantitativas de las asignaturas matriculadas
00029 del perfil "Alumno" de la pestaña "Expediente académico personal".
00030
00031     Args:
00032         alumno_id (str): Identificador del alumno.
00033         curso_academico (list): Lista con los cursos académicos
00034         titulacion (str): Titulación seleccionada
00035
00036     Returns:
00037         go.Figure: Figura con el gráfico
00038     """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     title={"text": "Calificaciones cuantitativas", "x": 0.5},
00044     xaxis={"title": "Asignatura", "tickangle": 45},
00045     yaxis={"title": "Calificación"},
00046     height=600,
00047     showlegend=True,
00048     legend=dict(
00049         x=1,
00050         y=1,
00051         traceorder="normal",
00052         font=dict(
00053             size=10,
00054         ),
00055         title="Asignaturas",
00056     ),
00057 )
00058
00059 if not (alumno_id and curso_academico and titulacion):
00060     return fig
00061
00062 try:
00063     curso_academico = list_to_tuple(curso_academico)
00064 except Exception as e:
00065     print("Error:", e)
00066     return fig
00067
00068 data = calif_numerica_asignatura(alumno_id, curso_academico, titulacion)
00069
00070 if not data:
00071     return fig
00072
00073 df = pd.DataFrame(data, columns=["Asignatura", "Calificacion"])
00074 colors = random_color(len(df))
00075
00076 for index, row in df.iterrows():
00077     fig.add_trace(
00078         go.Bar(
00079             x=[row["Asignatura"]],
00080             y=[row["Calificacion"]],
00081             name=row["Asignatura"],
00082             marker_color=colors[index],
00083             opacity=0.7,
00084         )
00085     )
00086
00087 return fig

```

Gráfico de llamadas para esta función:



4.16. Referencia del Namespace callbacks.alumnado.graphs.personal. \leftarrow callback_graph_progreso_academico_alumnado

Funciones

- def update_graph_alumnado (alumno_id, curso_academico, titulacion)

4.16.1. Documentación de las funciones

4.16.1.1. update_graph_alumnado()

```
def callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado.update_←
graph_alumnado (
    alumno_id,
    curso_academico,
    titulacion )
```

Actualiza el gráfico de evolución del progreso académico del perfil "Alumno" de la pestaña "Expediente académico personal".

Args:

 alumno_id (str): Identificador del alumno.
 curso_academico (list): Lista con los cursos académicos
 titulacion (str): Titulación seleccionada

Returns:

go.Figure: Figura con el gráfico

Definición en la línea 26 del archivo [callback_graph_progreso_academico_alumnado.py](#).

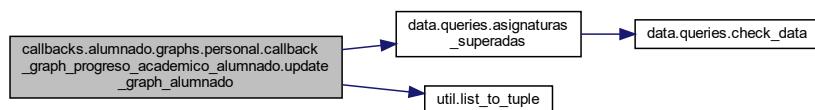
```
00026 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00027     """
00028     Actualiza el gráfico de evolución del progreso académico del perfil "Alumno"
00029     de la pestaña "Expediente académico personal".
00030
00031     Args:
00032         alumno_id (str): Identificador del alumno.
00033         curso_academico (list): Lista con los cursos académicos
00034         titulacion (str): Titulación seleccionada
00035
00036     Returns:
00037     go.Figure: Figura con el gráfico
00038
00039     """
00040
00041 fig = go.Figure()
00042
00043 fig.update_layout(
00044     title={"text": "Evolución del progreso académico", "x": 0.5},
00045     xaxis={"title": "Curso académico"},
00046     yaxis={"title": "Nº Asignaturas de superadas"},
00047     showlegend=False,
00048 )
00049
00050 if not (alumno_id and curso_academico and titulacion):
00051     return fig
00052
00053 try:
00054     curso_academico = list_to_tuple(curso_academico)
00055 except Exception as e:
00056     print("Error:", e)
00057     return fig
00058
00059 data = asignaturas_superadas(alumno_id, curso_academico, titulacion)
```

```

00060
00061     if not data:
00062         return fig
00063
00064     data = pd.DataFrame(data)
00065     academic_years = data["curso_academico"]
00066     subjects_passed = data["n_asig_superadas"]
00067
00068     cumulative_passed = []
00069     cumulative_total = 0
00070     for count in subjects_passed:
00071         cumulative_total += count
00072         cumulative_passed.append(cumulative_total)
00073
00074     fig.add_trace(
00075         go.Bar(x=academic_years, y=cumulative_passed, marker_color="blue", opacity=0.7)
00076     )
00077
00078     return fig

```

Gráfico de llamadas para esta función:



4.17. Referencia del Namespace callbacks.alumnado.graphs.personal.← callback_graph_tasa_rendimiento_alumnado

Funciones

- def [update_graph_alumnado](#) (alumno_id, curso_academico, titulacion)

4.17.1. Documentación de las funciones

4.17.1.1. [update_graph_alumnado\(\)](#)

```

def callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado.update_graph←
_alumnado (
    alumno_id,
    curso_academico,
    titulacion )

```

Actualiza el gráfico de tasa de rendimiento del perfil "Alumno" de la pestaña "Expediente académico personal".

Args:

alumno_id (str): Identificador del alumno.
curso_academico (list): Lista con los cursos académicos
titulacion (str): Titulación seleccionada

Returns:

go.Figure: Figura con el gráfico

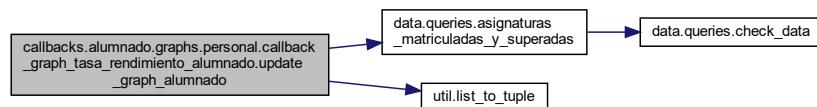
Definición en la línea 25 del archivo [callback_graph_tasa_rendimiento_alumnado.py](#).

```

00025 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00026 """
00027 Actualiza el gráfico de tasa de rendimiento del perfil "Alumno"
00028 de la pestaña "Expediente académico personal".
00029
00030     Args:
00031         alumno_id (str): Identificador del alumno.
00032         curso_academico (list): Lista con los cursos académicos
00033         titulacion (str): Titulación seleccionada
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037
00038 """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     title={"text": "Tasa de rendimiento por curso académico", "x": 0.5},
00044     xaxis={"title": "Tasa de rendimiento (%)"},
00045     yaxis={"title": "Curso académico"},
00046     showlegend=False,
00047 )
00048
00049 if not (alumno_id and curso_academico and titulacion):
00050     return fig
00051
00052 try:
00053     curso_academico = list_to_tuple(curso_academico)
00054 except Exception as e:
00055     print("Error:", e)
00056     return fig
00057
00058 data = asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion)
00059
00060 if not data:
00061     return fig
00062
00063 df = pd.DataFrame(data, columns=["curso_academico", "matriculadas", "superadas"])
00064 df["tasa_rendimiento"] = (df["superadas"] / df["matriculadas"]) * 100
00065
00066 fig.add_trace(
00067     go.Bar(
00068         x=df["tasa_rendimiento"],
00069         y=df["curso_academico"],
00070         orientation="h",
00071         marker_color="blue",
00072         opacity=0.7,
00073         width=0.7,
00074     )
00075 )
00076
00077 return fig

```

Gráfico de llamadas para esta función:



4.18. Referencia del Namespace callbacks.alumnado.utils

Namespaces

- namespace [callback_resumen_alumnado](#)
- namespace [callback_select_alumnado](#)
- namespace [callback_tabs_alumnado](#)

4.19. Referencia del Namespace

callbacks.alumnado.utils.callback_resumen_alumnado

Funciones

- def `update_resumen_alumnado` (`alumno_id, titulacion`)
- def `not_data` ()

4.19.1. Documentación de las funciones

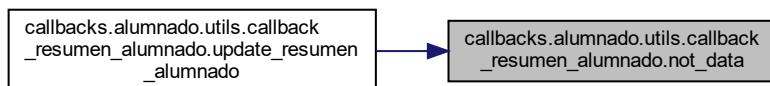
4.19.1.1. not_data()

```
def callbacks.alumnado.utils.callback_resumen_alumnado.not_data ( )
```

Definición en la línea 73 del archivo `callback_resumen_alumnado.py`.

```
00073 def not_data():
00074     return html.Div(
00075         [
00076             html.H2("Resumen"),
00077             html.P("Universidad:", className="resumen-label"),
00078             html.P("No disponible"),
00079             html.P("Titulación:", className="resumen-label"),
00080             html.P("No disponible"),
00081             html.P("Alumno:", className="resumen-label"),
00082             html.P("No disponible"),
00083             html.P("Nota Media:", className="resumen-label"),
00084             html.P("No disponible"),
00085             html.Hr(),
00086         ]
00087     )
```

Gráfico de llamadas a esta función:



4.19.1.2. update_resumen_alumnado()

```
def callbacks.alumnado.utils.callback_resumen_alumnado.update_resumen_alumnado (
    alumno_id,
    titulacion )
```

Actualiza el resumen del perfil "Alumno".

Args:

alumno_id (str): Identificador del alumno.
titulacion (str): Titulación seleccionada

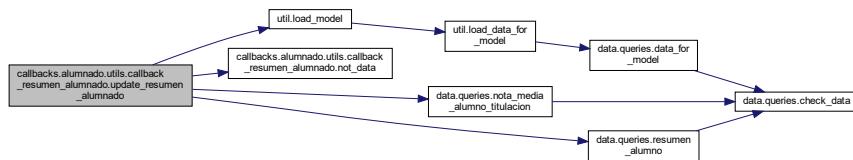
Returns:

list: Componentes con el resumen del perfil "Alumno"

Definición en la línea 21 del archivo [callback_resumen_alumnado.py](#).

```
00021 def update_resumen_alumnado(alumno_id, titulacion):
00022     """
00023     Actualiza el resumen del perfil "Alumno".
00024
00025     Args:
00026         alumno_id (str): Identificador del alumno.
00027         titulacion (str): Titulación seleccionada
00028
00029     Returns:
00030         list: Componentes con el resumen del perfil "Alumno"
00031     """
00032 if not (alumno_id and titulacion):
00033     return not_data()
00034
00035 data = resumen_alumno(alumno_id, titulacion)
00036
00037 data = pd.DataFrame(data, columns=["id_alumno", "universidad", "titulacion", "abandona"])
00038
00039 if data.empty:
00040     return not_data()
00041
00042 componentes = [
00043     html.H2("Resumen"),
00044     html.P("Universidad:", className="resumen-label"),
00045     html.P(data["universidad"].iloc[0]),
00046     html.P("Titulación:", className="resumen-label"),
00047     html.P(data["titulacion"].iloc[0]),
00048     html.P("Alumno:", className="resumen-label"),
00049     html.P(alumno_id),
00050     html.P("Nota Media:", className="resumen-label"),
00051     html.P(not_a_media_alumno_titulacion(alumno_id, titulacion))
00052 ]
00053
00054 if data["abandona"].iloc[0] == 'no':
00055     componentes.append(html.P("Estado:", className="resumen-label"))
00056     componentes.append(html.P("Activo"))
00057     componentes.append(html.P("Probabilidad de abandono:", className="resumen-label"))
00058     data_model = load_model()
00059
00060     probabilidad_abandono = data_model[data_model['id'] ==
00061     alumno_id]['probabilidad_abandono'].values[0]
00062     componentes.append(html.P(f"{probabilidad_abandono:.2%}"))
00063 else:
00064     componentes.append(html.P("Estado:", className="resumen-label"))
00065     componentes.append(html.P("Abandona"))
00066
00067 componentes.append(html.Hr())
00068
00069 return html.Div(componentes)
00070
00071
00072
```

Gráfico de llamadas para esta función:



4.20. Referencia del Namespace

callbacks.alumnado.utils.callback_select_alumnado

Funciones

- def `store_selected_alumnado` (`selected_value`, `stored_value`)

4.20.1. Documentación de las funciones

4.20.1.1. `store_selected_alumnado()`

```
def callbacks.alumnado.utils.callback_select_alumnado.store_selected_alumnado (
    selected_value,
    stored_value )
```

Almacena el valor seleccionado en el dropdown de alumnos.

Args:

`selected_value` (str): Valor seleccionado en el dropdown
`stored_value` (str): Valor almacenado

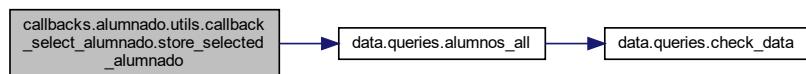
Returns:

`str`: Valor seleccionado
`list`: Opciones del dropdown
`str`: Valor almacenado

Definición en la línea 23 del archivo [callback_select_alumnado.py](#).

```
00023 def store_selected_alumnado(selected_value, stored_value):
00024     """
00025 Almacena el valor seleccionado en el dropdown de alumnos.
00026
00027 Args:
00028     selected_value (str): Valor seleccionado en el dropdown
00029     stored_value (str): Valor almacenado
00030
00031 Returns:
00032     str: Valor seleccionado
00033     list: Opciones del dropdown
00034     str: Valor almacenado
00035 """
00036 data = alumnos_all()
00037
00038 if not data:
00039     return None, [], None
00040
00041     opciones_dropdown = [{"label": alumno[0], "value": alumno[0]} for alumno in data]
00042
00043     if selected_value is None and stored_value in [
00044         op["value"] for op in opciones_dropdown
00045     ]:
00046         return stored_value, opciones_dropdown, stored_value
00047
00048     return selected_value, opciones_dropdown, selected_value
```

Gráfico de llamadas para esta función:



4.21. Referencia del Namespace callbacks.alumnado.utils.callback_tabs_alumnado

Funciones

- def `render_content` (tab, selected_alumnado)

4.21.1. Documentación de las funciones

4.21.1.1. `render_content()`

```
def callbacks.alumnado.utils.callback_tabs_alumnado.render_content (
    tab,
    selected_alumnado )
```

Renderiza el contenido de las pestañas del dashboard del alumnado.

Args:

```
    tab (str): Pestaña seleccionada
    selected_alumnado (str): Alumno seleccionado
```

Returns:

```
    list: Componentes de la pestaña seleccionada
```

Definición en la línea 30 del archivo `callback_tabs_alumnado.py`.

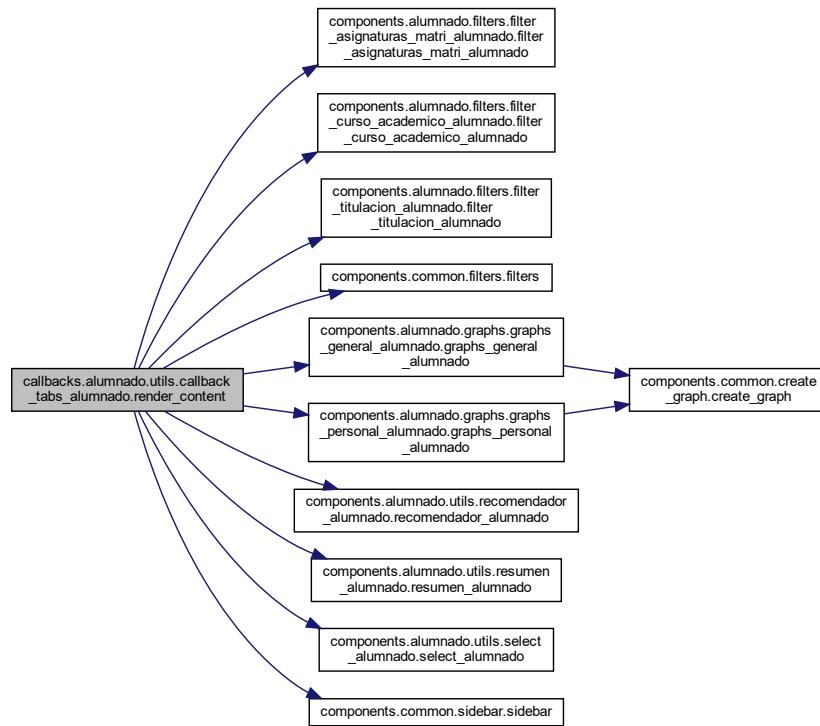
```
00030 def render_content(tab, selected_alumnado):
00031     """
00032     Renderiza el contenido de las pestañas del dashboard del alumnado.
00033
00034     Args:
00035         tab (str): Pestaña seleccionada
00036         selected_alumnado (str): Alumno seleccionado
00037
00038     Returns:
00039         list: Componentes de la pestaña seleccionada
00040
00041     """
00042     if tab == 'expediente-personal-tab':
00043         return html.Div([
00044             select_alumnado(),
00045             html.H2("Dashboard Alumnado", style={'textAlign': 'center'}),
00046             html.Div([
00047                 sidebar([
00048                     resumen_alumnado(),
00049                     filters([
00050                         filter_titulacion_alumnado(),
00051                         filter_curso_academico_alumnado()
00052                     ])
00053                 ]),
00054                 graphs_personal_alumnado()
00055             ], className='content-layout-dashboard')
00056         ]
00057     elif tab == 'rendimiento-academico-tab':
00058         return html.Div([
00059             html.H2("Dashboard Alumnado", style={'textAlign': 'center'}),
00060             html.Div([
00061                 sidebar([
00062                     resumen_alumnado(),
00063                     filters([
00064                         filter_titulacion_alumnado(),
00065                         filter_curso_academico_alumnado(),
00066                         filter_asignaturas_matri_alumnado()
00067                     ])
00068                 ]),
00069                 graphs_general_alumnado()
```

```

00070         ], className='content-layout-dashboard')
00071     )
00072     elif tab == 'recomendador-tab':
00073         return html.Div([
00074             recomendador_alumnado()
00075         ])
00076

```

Gráfico de llamadas para esta función:



4.22. Referencia del Namespace callbacks.common

Namespaces

- namespace `callback_sidebar_collapse`
- namespace `callback_update_layout`
- namespace `callback_user_role`

4.23. Referencia del Namespace callbacks.common.callback_sidebar_collapse

Funciones

- def `sidebarCollapse (n, is_open)`

4.23.1. Documentación de las funciones

4.23.1.1. sidebar_collapse()

```
def callbacks.common.callback_sidebarCollapse.sidebarCollapse ( n,  
                                                               is_open )
```

Callback que permite colapsar y descolapsar el sidebar.

Args:
n (int): Número de clicks
is_open (bool): Estado del sidebar

Returns:
bool: Estado del sidebar

Definición en la línea 19 del archivo [callback_sidebar_collapse.py](#).

```
00019 def sidebarCollapse(n, is_open):  
00020     """  
00021     Callback que permite colapsar y descolapsar el sidebar.  
00022  
00023     Args:  
00024     n (int): Número de clicks  
00025     is_open (bool): Estado del sidebar  
00026  
00027     Returns:  
00028     bool: Estado del sidebar  
00029     """  
00030     if n:  
00031         return not is_open  
00032     return is_open
```

4.24. Referencia del Namespace callbacks.common.callback_update_layout

Funciones

- def [update_layout](#) (role)

4.24.1. Documentación de las funciones

4.24.1.1. update_layout()

```
def callbacks.common.callback_update_layout.update_layout (
    role )
```

Actualiza el layout de la aplicación según el rol del usuario.

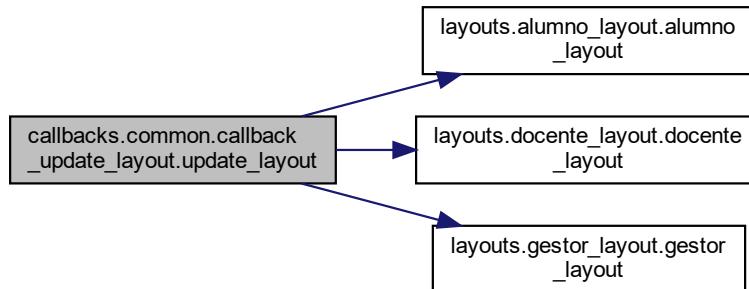
Args:
 role (str): Rol del usuario

Returns:
 str: Layout correspondiente al rol del usuario

Definición en la línea 21 del archivo [callback_update_layout.py](#).

```
00021 def update_layout(role):
00022     """
00023 Actualiza el layout de la aplicación según el rol del usuario.
00024
00025 Args:
00026     role (str): Rol del usuario
00027
00028 Returns:
00029     str: Layout correspondiente al rol del usuario
00030
00031 """
00032 if role == 'Alumno':
00033     return alumno_layout()
00034 elif role == 'Docente':
00035     return docente_layout()
00036 elif role == 'Gestor':
00037     return gestor_layout()
00038
00039 return ""
```

Gráfico de llamadas para esta función:



4.25. Referencia del Namespace `callbacks.common.callback_user_role`

Funciones

- def `initialize_dropdown` (ts, stored_role)
- def `update_role` (new_role, current_role)

Variables

- bool `prevent_initial_call` = True,

4.25.1. Documentación de las funciones

4.25.1.1. initialize_dropdown()

```
def callbacks.common.callback_user_role.initialize_dropdown (
    ts,
    stored_role )
```

Inicializa el dropdown con el rol almacenado.

Args:

ts (float): Timestamp de la última modificación del store
stored_role (str): Rol almacenado

Definición en la línea 20 del archivo [callback_user_role.py](#).

```
00020 def initialize_dropdown(ts, stored_role):
00021     """
00022     Inicializa el dropdown con el rol almacenado.
00023
00024     Args:
00025         ts (float): Timestamp de la última modificación del store
00026         stored_role (str): Rol almacenado
00027     """
00028     return stored_role if ts is not None else "Alumno"
00029
00030
00031 @callback(
00032     Output("store-role", "data"),
00033     Input("dropdown_role", "value"),
00034     State("store-role", "data"),
00035     prevent_initial_call=True,
00036 )
```

4.25.1.2. update_role()

```
def callbacks.common.callback_user_role.update_role (
    new_role,
    current_role )
```

Actualiza el rol almacenado en el store.

Args:

new_role (str): Nuevo rol seleccionado
current_role (str): Rol actual almacenado

Returns:

str: Rol almacenado

Definición en la línea 37 del archivo [callback_user_role.py](#).

```
00037 def update_role(new_role, current_role):
00038     """
00039     Actualiza el rol almacenado en el store.
00040
00041     Args:
00042         new_role (str): Nuevo rol seleccionado
00043         current_role (str): Rol actual almacenado
00044
00045     Returns:
00046         str: Rol almacenado
00047     """
00048     if new_role != current_role:
00049         return new_role
00050     return current_role
```

4.25.2. Documentación de las variables

4.25.2.1. prevent_initial_call

```
bool callbacks.common.callback_user_role.prevent_initial_call = True,
```

Definición en la línea 18 del archivo [callback_user_role.py](#).

4.26. Referencia del Namespace callbacks.docente

Namespaces

- namespace [filters](#)
- namespace [graphs](#)
- namespace [utils](#)

4.27. Referencia del Namespace callbacks.docente.filters

Namespaces

- namespace [callback_filter_all_asignaturas_titulacion_docente](#)
- namespace [callback_filter_all_curso_academico](#)
- namespace [callback_filter_asignaturas_docente](#)
- namespace [callback_filter_curso_academico_docente](#)
- namespace [callback_filter_titulacion_docente](#)

4.28. Referencia del Namespace callbacks.docente.filters.callback_← filter_all_asignaturas_titulacion_docente

Funciones

- def [update_filter_asignaturas_docente](#) (titulacion, curso_academico, n_clicks, existing_options)

4.28.1. Documentación de las funciones

4.28.1.1. update_filter_asignaturas_docente()

```
def callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente.update_←
filter_asignaturas_docente (
    titulacion,
    curso_academico,
    n_clicks,
    existing_options )
```

Actualiza las opciones del dropdown con todas las asignaturas de la titulación seleccionada y gestiona el evento del botón que selecciona todas las asignaturas.

Args:

```
titulacion (str): Titulación seleccionada
curso_academico (list): Cursos académicos seleccionados
n_clicks (int): Número de clicks en el botón "Seleccionar todo"
existing_options (list): Opciones actuales del dropdown
```

Returns:

```
list: Opciones del dropdown
list: Valor seleccionado
```

Definición en la línea 25 del archivo [callback_filter_all_asignaturas_titulacion_docente.py](#).

```
00025 def update_filter_asignaturas_docente(titulacion, curso_academico, n_clicks, existing_options):
00026     """
00027     Actualiza las opciones del dropdown con todas las asignaturas de la titulación seleccionada y
00028     gestiona el evento del botón que selecciona todas las asignaturas.
00029
00030     Args:
00031         titulacion (str): Titulación seleccionada
00032         curso_academico (list): Cursos académicos seleccionados
00033         n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00034         existing_options (list): Opciones actuales del dropdown
00035
00036     Returns:
00037         list: Opciones del dropdown
00038         list: Valor seleccionado
00039     """
00040     ctx = callback_context
00041     trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00042
00043     if trigger_id == "select-all-asignaturas-titulacion-docente" and n_clicks > 0:
00044         return existing_options, [option["value"] for option in existing_options]
00045
00046     if not (titulacion and curso_academico):
00047         return [], None
00048
00049     data = asignaturas_actas_titulacion(titulacion, curso_academico)
00050
00051     if not data:
00052         return [], None
00053
00054     opciones_dropdown = [
00055         {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
00056     ]
00057     value = (
00058         [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
00059     )
00060
00061     return opciones_dropdown, value
```

Gráfico de llamadas para esta función:



4.29. Referencia del Namespace

callbacks.docente.filters.callback_filter_all_curso_academico

Funciones

- def update_filter_all_cursos_academicos_docente (titulacion)

4.29.1. Documentación de las funciones

4.29.1.1. update_filter_all_cursos_academicos_docente()

```
def callbacks.docente.filters.callback_filter_all_curso_academico.update_filter_all_cursos_academicos_docente (
    titulacion )
```

Actualiza las opciones del dropdown con todos los cursos académicos de la titulación seleccionada.

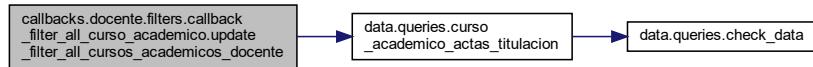
Args:
 titulacion (str): Titulación seleccionada

Returns:
 list: Opciones del dropdown
 list: Valor seleccionado

Definición en la línea 21 del archivo [callback_filter_all_curso_academico.py](#).

```
00021 def update_filter_all_cursos_academicos_docente(titulacion):
00022     """
00023 Actualiza las opciones del dropdown con todos los cursos académicos de la titulación seleccionada.
00024
00025 Args:
00026     titulacion (str): Titulación seleccionada
00027
00028 Returns:
00029     list: Opciones del dropdown
00030     list: Valor seleccionado
00031 """
00032
00033 if not titulacion:
00034     return [], None
00035
00036     data = curso_academico_actas_titulacion(titulacion)
00037
00038     if not data:
00039         return [], None
00040
00041     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00042     value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00043
00044     return opciones_dropdown, value
```

Gráfico de llamadas para esta función:



4.30. Referencia del Namespace callbacks.docente.filters.callback_filter_asignaturas_docente

Funciones

- def update_filter_asignaturas_docente (docente_id, titulacion)

4.30.1. Documentación de las funciones

4.30.1.1. update_filter_asignaturas_docente()

```
def callbacks.docente.filters.callback_filter_asignaturas_docente.update_filter_asignaturas_docente (
    docente_id,
    titulacion )
```

Actualiza las opciones del dropdown con las asignaturas del docente seleccionado.

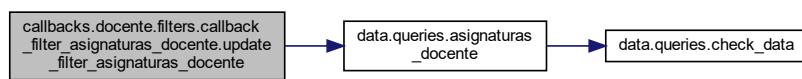
Args:
 docente_id (str): ID del docente
 titulacion (str): Titulación seleccionada

Returns:
 list: Opciones del dropdown
 str: Valor seleccionado

Definición en la línea 22 del archivo [callback_filter_asignaturas_docente.py](#).

```
00022 def update_filter_asignaturas_docente(docente_id, titulacion):
00023     """
00024     Actualiza las opciones del dropdown con las asignaturas del docente seleccionado.
00025
00026     Args:
00027         docente_id (str): ID del docente
00028         titulacion (str): Titulación seleccionada
00029
00030     Returns:
00031         list: Opciones del dropdown
00032         str: Valor seleccionado
00033
00034     """
00035     if not (docente_id and titulacion):
00036         return [], None
00037
00038     data = asignaturas_docente(docente_id, titulacion)
00039
00040     if not data:
00041         return [], None
00042
00043     opciones_dropdown = [
00044         {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
00045     ]
00046     value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00047
00048     return opciones_dropdown, value
```

Gráfico de llamadas para esta función:



4.31. Referencia del Namespace callbacks.docente.filters.callback_← filter_curso_academico_docente

Funciones

- def [update_filter_curso_academico_docente](#) (docente_id, asignatura, n_clicks, existing_options)

4.31.1. Documentación de las funciones

4.31.1.1. update_filter_curso_academico_docente()

```
def callbacks.docente.filters.callback_filter_curso_academico_docente.update_filter_curso_←
academico_docente (
    docente_id,
    asignatura,
    n_clicks,
    existing_options )
```

Actualiza las opciones del dropdown con los cursos académicos del docente seleccionado y gestiona el evento del botón que selecciona todos los cursos académicos.

Args:

```
    docente_id (str): ID del docente
    asignatura (str): Asignatura seleccionada
    n_clicks (int): Número de clicks en el botón "Seleccionar todo"
    existing_options (list): Opciones actuales del dropdown
```

Returns:

```
    list: Opciones del dropdown
    list: Valor seleccionado
```

Definición en la línea 26 del archivo [callback_filter_curso_academico_docente.py](#).

```
00026 def update_filter_curso_academico_docente(docente_id, asignatura, n_clicks, existing_options):
00027     """
00028 Actualiza las opciones del dropdown con los cursos académicos del docente seleccionado y
00029 gestiona el evento del botón que selecciona todos los cursos académicos.
00030
00031 Args:
00032     docente_id (str): ID del docente
00033     asignatura (str): Asignatura seleccionada
00034     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00035     existing_options (list): Opciones actuales del dropdown
00036
00037     Returns:
00038         list: Opciones del dropdown
00039         list: Valor seleccionado
00040
00041     """
00042     ctx = callback_context
00043     trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00044
00045     if trigger_id == "select-all-cursos-academicos" and n_clicks > 0:
00046         return existing_options, [option["value"] for option in existing_options]
00047
00048     if not (docente_id and asignatura):
00049         return [], None
00050
00051     data = curso_academico_docente(docente_id, asignatura)
00052
00053     if not data:
00054         return [], None
00055
```

```

00056     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00057     value = [option["value"] for option in opciones_dropdown]
00058
00059     return opciones_dropdown, value

```

Gráfico de llamadas para esta función:



4.32. Referencia del Namespace callbacks.docente.filters.callback_filter_titulacion_docente

Funciones

- def [update_filter_titulacion_docente](#)(docente_id, selected_value, stored_titulacion)

4.32.1. Documentación de las funciones

4.32.1.1. update_filter_titulacion_docente()

```

def callbacks.docente.filters.callback_filter_titulacion_docente.update_filter_titulacion_docente (
    docente_id,
    selected_value,
    stored_titulacion )

```

Actualiza las opciones del dropdown con las titulaciones del docente seleccionado.

Args:

docente_id (str): ID del docente
selected_value (str): Valor seleccionado
stored_titulacion (str): Titulación seleccionada

Returns:

list: Opciones del dropdown
str: Valor seleccionado

Definición en la línea 23 del archivo [callback_filter_titulacion_docente.py](#).

```

00023 def update_filter_titulacion_docente(docente_id, selected_value, stored_titulacion):
00024 """
00025 Actualiza las opciones del dropdown con las titulaciones del docente seleccionado.
00026
00027 Args:
00028 docente_id (str): ID del docente
00029 selected_value (str): Valor seleccionado
00030 stored_titulacion (str): Titulación seleccionada
00031
00032 Returns:

```

```

00033 list: Opciones del dropdown
00034 str: Valor seleccionado
00035
00036 """
00037 if not docente_id:
00038     return [], None, None
00039
00040 data = titulacion_docente(docente_id)
00041
00042 if not data:
00043     return [], None, None
00044
00045 opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
00046
00047 if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
00048     return opciones_dropdown, stored_titulacion, stored_titulacion
00049
00050 return opciones_dropdown, selected_value, selected_value
00051
00052

```

Gráfico de llamadas para esta función:



4.33. Referencia del Namespace callbacks.docente.graphs

Namespaces

- namespace [general](#)
- namespace [personal](#)

4.34. Referencia del Namespace callbacks.docente.graphs.general

Namespaces

- namespace [callback_graph_all_calif_cualitativa_docente](#)
- namespace [callback_graph_all_calif_media_docente](#)

4.35. Referencia del Namespace callbacks.docente.graphs.general.[←](#) callback_graph_all_calif_cualitativa_docente

Funciones

- def [update_graph_docente](#) (titulacion, curso_academico, asignatura)

4.35.1. Documentación de las funciones

4.35.1.1. update_graph_docente()

```
def callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente.update_←
graph_docente (
    titulacion,
    curso_academico,
    asignatura )
```

Actualiza el gráfico de calificaciones cualitativas de la titulación seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".

Args:

```
    titulacion (str): Titulación seleccionada
    curso_academico (list): Lista con los cursos académicos
    asignatura (list): Lista con las asignaturas seleccionadas
```

Returns:

```
    go Figure: Figura con el gráfico
```

Definición en la línea 25 del archivo [callback_graph_all_calif_cualitativa_docente.py](#).

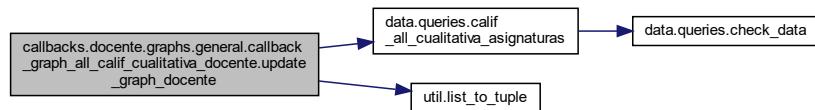
```
00025 def update_graph_docente(titulacion, curso_academico, asignatura):
00026     """
00027     Actualiza el gráfico de calificaciones cualitativas de la titulación
00028     seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00029
00030     Args:
00031         titulacion (str): Titulación seleccionada
00032         curso_academico (list): Lista con los cursos académicos
00033         asignatura (list): Lista con las asignaturas seleccionadas
00034
00035     Returns:
00036         go Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     barmode="stack",
00043         title={"text": "Calificaciones cualitativas de la titulación", "x": 0.5},
00044         xaxis={"title": "Asignaturas", "tickangle": 45},
00045         yaxis={"title": "Nº Alumnos matriculados"},
00046         legend=dict(
00047             x=1,
00048             y=1,
00049             traceorder="normal",
00050             font=dict(
00051                 size=10
00052             ),
00053             title="Calificaciones"
00054         ),
00055     )
00056
00057 if not (curso_academico and asignatura):
00058     return fig
00059
00060 try:
00061     asignatura = list_to_tuple(asignatura)
00062 except Exception as e:
00063     return fig
00064
00065 data = calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignatura)
00066
00067 if not data:
00068     return fig
00069
00070 df = pd.DataFrame(
00071     data, columns=["titulacion", "asignatura", "calificación", "n_alumnos"])
00072
00073
00074 colors_mapping = {
00075     "Sobresaliente": "blue",
00076     "Notable": "green",
00077     "Aprobado": "orange",
00078     "Suspensos": "red",
00079     "No presentado": "gray",
00080 }
00081 for calif, color in colors_mapping.items():
```

```

00083     df_calif = df[df["calificación"] == calif]
00084     fig.add_trace(
00085         go.Bar(
00086             x=df_calif["asignatura"],
00087             y=df_calif["n_alumnos"],
00088             name=calif,
00089             marker_color=color,
00090             opacity=0.7,
00091         )
00092     )
00093
00094     fig.update_xaxes(categoryorder="array", categoryarray=asignatura)
00095
00096     return fig

```

Gráfico de llamadas para esta función:



4.36. Referencia del Namespace callbacks.docente.graphs.general.← callback_graph_all_calif_media_docente

Funciones

- def [update_graph_docente](#) (titulacion, curso_academico, asignatura)

4.36.1. Documentación de las funciones

4.36.1.1. update_graph_docente()

```

def callbacks.docente.graphs.general.callback_graph_all_calif_media_docente.update_graph_←
docente (
    titulacion,
    curso_academico,
    asignatura )

```

Actualiza el gráfico de nota media por asignatura de la titulación seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".

Args:

titulacion (str): Titulación seleccionada
curso_academico (list): Lista con los cursos académicos
asignatura (list): Lista con las asignaturas seleccionadas

Returns:

go.Figure: Figura con el gráfico

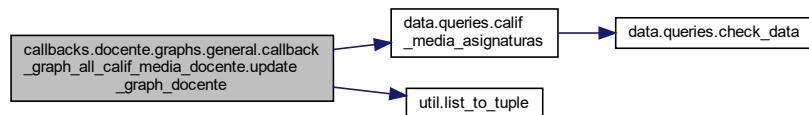
Definición en la línea 25 del archivo [callback_graph_all_calif_media_docente.py](#).

```

00025 def update_graph_docente(titulacion, curso_academico, asignatura):
00026 """
00027 Actualiza el gráfico de nota media por asignatura de la titulación
00028 seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00029
00030 Args:
00031     titulacion (str): Titulación seleccionada
00032     curso_academico (list): Lista con los cursos académicos
00033     asignatura (list): Lista con las asignaturas seleccionadas
00034
00035 Returns:
00036     go.Figure: Figura con el gráfico
00037 """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     title={"text": "Nota media por asignatura de la titulación", "x": 0.5},
00043     xaxis_title="Asignaturas",
00044     yaxis_title="Nota media",
00045 )
00046
00047 if not (curso_academico and asignatura):
00048     return fig
00049
00050 try:
00051     asignatura = list_to_tuple(asignatura)
00052 except Exception as e:
00053     return fig
00054
00055 data = calif_media_asignaturas(titulacion, curso_academico, asignatura)
00056
00057 if not data:
00058     return fig
00059
00060 df = pd.DataFrame(data, columns=["asignatura", "media_calif"])
00061
00062 fig.add_trace(
00063     go.Bar(
00064         x=df["asignatura"], y=df["media_calif"], marker_color="blue", opacity=0.7
00065     )
00066 )
00067
00068 return fig

```

Gráfico de llamadas para esta función:



4.37. Referencia del Namespace callbacks.docente.graphs.personal

Namespaces

- namespace [callback_graph_alu_genero_docente](#)
- namespace [callback_graph_alu_media_docente](#)
- namespace [callback_graph_alu_nota_cualitativa_docente](#)
- namespace [callback_graph_alu_repetidores_nuevos_docente](#)

4.38. Referencia del Namespace callbacks.docente.graphs.personal. callback_graph_alu_genero_docente

Funciones

- def update_graph_docente(asignaturas, curso_academico, docente_id)

4.38.1. Documentación de las funciones

4.38.1.1. update_graph_docente()

```
def callbacks.docente.graphs.personal.callback_graph_alu_genero_docente.update_graph_docente (
    asignaturas,
    curso_academico,
    docente_id )
```

Actualiza el gráfico de evolución de alumnos matriculados por género del perfil "Docente" de la pestaña "Rendimiento académico personal".

Args:

asignaturas (list): Lista con las asignaturas seleccionadas
 curso_academico (list): Lista con los cursos académicos
 docente_id (str): Identificador del docente

Returns:

go.Figure: Figura con el gráfico

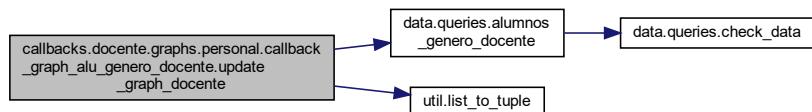
Definición en la línea 25 del archivo [callback_graph_alu_genero_docente.py](#).

```
00025 def update_graph_docente(asignaturas, curso_academico, docente_id):
00026     """
00027 Actualiza el gráfico de evolución de alumnos matriculados por género
00028 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00029
00030     Args:
00031         asignaturas (list): Lista con las asignaturas seleccionadas
00032         curso_academico (list): Lista con los cursos académicos
00033         docente_id (str): Identificador del docente
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     barmode="stack",
00043     title={"text": "Evolución alumnos matriculados por género", "x": 0.5},
00044     xaxis={"title": "Curso académico"},
00045     yaxis={"title": "Nº Alumnos matriculados"},
00046     legend={"title": "Género"},
00047 )
00048
00049 if not (asignaturas and curso_academico and docente_id):
00050     return fig
00051
00052 try:
00053     curso_academico = list_to_tuple(curso_academico)
00054 except Exception as e:
00055     return fig
00056
00057 data = alumnos_genero_docente(docente_id, asignaturas, curso_academico)
00058
00059 if not data:
00060     return fig
```

4.39 Referencia del Namespace callbacks.docente.graphs.personal.callback_graph_alu_media_docente 51

```
00061 df = pd.DataFrame(data, columns=["curso_academico", "genero", "cantidad"])
00062
00063 df_pivot = df.pivot(
00064     index="curso_academico", columns="genero", values="cantidad"
00065 ).fillna(0)
00066
00067 colores = {"Femenino": "red", "Masculino": "blue"}
00068
00069 for genero in df_pivot.columns:
00070     fig.add_trace(
00071         go.Bar(
00072             x=df_pivot.index,
00073             y=df_pivot[genero],
00074             name="Mujeres" if genero == "Femenino" else "Hombres",
00075             marker_color=colores[genero],
00076             opacity=0.7,
00077         )
00078     )
00079
00080
00081 return fig
```

Gráfico de llamadas para esta función:



4.39. Referencia del Namespace callbacks.docente.graphs.personal.← callback_graph_alu_media_docente

Funciones

- def [update_graph_docente](#) (asignaturas, curso_academico, docente_id)

4.39.1. Documentación de las funciones

4.39.1.1. update_graph_docente()

```
def callbacks.docente.graphs.personal.callback_graph_alu_media_docente.update_graph_docente (
    asignaturas,
    curso_academico,
    docente_id )
```

Actualiza el gráfico de evolución de la nota media por asignatura del perfil "Docente" de la pestaña "Rendimiento académico personal".

Args:

asignaturas (list): Lista con las asignaturas seleccionadas
curso_academico (list): Lista con los cursos académicos
docente_id (str): Identificador del docente

Returns:

go.Figure: Figura con el gráfico

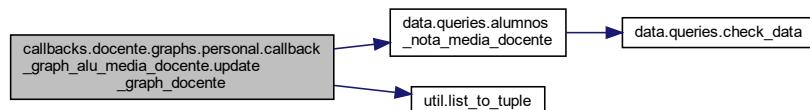
Definición en la línea 25 del archivo [callback_graph_alu_media_docente.py](#).

```

00025 def update_graph_docente(asignaturas, curso_academico, docente_id):
00026 """
00027 Actualiza el gráfico de evolución de la nota media por asignatura
00028 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00029
00030     Args:
00031         asignaturas (list): Lista con las asignaturas seleccionadas
00032         curso_academico (list): Lista con los cursos académicos
00033         docente_id (str): Identificador del docente
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     title={"text": "Evolución nota media por asignatura", "x": 0.5},
00043     xaxis_title="Curso académico",
00044     yaxis_title="Nota media",
00045     xaxis=dict(type="category"),
00046 )
00047
00048 if not (asignaturas and curso_academico and docente_id):
00049     return fig
00050
00051 try:
00052     curso_academico = list_to_tuple(curso_academico)
00053     asignaturas = list_to_tuple(asignaturas)
00054 except Exception as e:
00055     return fig
00056
00057 data = alumnos_nota_media_docente(asignaturas, curso_academico)
00058
00059 if not data:
00060     return fig
00061
00062 df = pd.DataFrame(data, columns=["curso_academico", "asignatura", "nota_media"])
00063
00064 fig.add_trace(
00065     go.Bar(
00066         x=df["curso_academico"],
00067         y=df["nota_media"],
00068         name="Nota media",
00069         marker=dict(color="blue"),
00070         opacity=0.7,
00071     )
00072 )
00073
00074 return fig

```

Gráfico de llamadas para esta función:



4.40. Referencia del Namespace [callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente](#)

Funciones

- [def update_graph_docente \(asignaturas, curso_academico, docente_id\)](#)

4.40.1. Documentación de las funciones

4.40.1.1. update_graph_docente()

```
def callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente.update_←
graph_docente (
    asignaturas,
    curso_academico,
    docente_id )
```

Actualiza el gráfico de evolución de calificaciones cualitativas del perfil "Docente" de la pestaña "Rendimiento académico personal".

Args:

- asignaturas (list): Lista con las asignaturas seleccionadas
- curso_academico (list): Lista con los cursos académicos
- docente_id (str): Identificador del docente

Returns:

- go.Figure: Figura con el gráfico

Definición en la línea 25 del archivo [callback_graph_alu_nota_cualitativa_docente.py](#).

```
00025 def update_graph_docente(asignaturas, curso_academico, docente_id):
00026     """
00027     Actualiza el gráfico de evolución de calificaciones cualitativas
00028     del perfil "Docente" de la pestaña "Rendimiento académico personal".
00029
00030     Args:
00031         asignaturas (list): Lista con las asignaturas seleccionadas
00032             curso_academico (list): Lista con los cursos académicos
00033             docente_id (str): Identificador del docente
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     barmode="stack",
00043         title={"text": "Evolución calificaciones cualitativas", "x": 0.5},
00044         xaxis={"title": "Curso académico"},
00045         yaxis={"title": "Nº Alumnos matriculados"},
00046         legend_title_text="Calificación",
00047     )
00048
00049 if not (asignaturas and curso_academico and docente_id):
00050     return fig
00051
00052 try:
00053     curso_academico = list_to_tuple(curso_academico)
00054     asignaturas = list_to_tuple(asignaturas)
00055 except Exception as e:
00056     return fig
00057
00058 data = alumnos_nota_cualitativa_docente(asignaturas, curso_academico)
00059 if not data:
00060     return fig
00061
00062 df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "n_alumnos"])
00063
00064 # Pivotear el DataFrame para obtener las cantidades por calificación en columnas separadas
00065 df_pivot = df.pivot(
00066     index="curso_academico", columns="calificacion", values="n_alumnos"
00067 ).fillna(0)
00068 catagories = ["No presentado", "Suspensos", "Aprobado", "Notable", "Sobresaliente"]
00069
00070 for calif in catagories:
00071     if calif not in df_pivot:
00072         df_pivot[calif] = 0
00073
```

```

00074     df_pivot = df_pivot[catalogos]
00075
00076     colors = {
00077         "Sobresaliente": "blue",
00078         "Notable": "green",
00079         "Aprobado": "orange",
00080         "Suspensos": "red",
00081         "No presentado": "gray",
00082     }
00083
00084     for calif in catalogos:
00085         fig.add_trace(
00086             go.Bar(
00087                 x=df_pivot.index,
00088                 y=df_pivot[calif],
00089                 name=calif,
00090                 marker=dict(color=colors[calif]),
00091                 opacity=0.7,
00092             )
00093         )
00094
00095     return fig

```

Gráfico de llamadas para esta función:



4.41. Referencia del Namespace callbacks.docente.graphs.personal.← callback_graph_alu_repetidores_nuevos_docente

Funciones

- def [update_graph_docente](#) (asignaturas, curso_academico, docente_id)

4.41.1. Documentación de las funciones

4.41.1.1. update_graph_docente()

```

def callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente.update_←
graph_docente (
    asignaturas,
    curso_academico,
    docente_id )

```

Actualiza el gráfico de evolución de alumnos de nuevo ingreso y repetidores del perfil "Docente" de la pestaña "Rendimiento académico personal".

Args:

asignaturas (list): Lista con las asignaturas seleccionadas
curso_academico (list): Lista con los cursos académicos
docente_id (str): Identificador del docente

Returns:

go.Figure: Figura con el gráfico

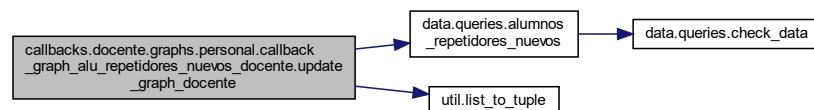
Definición en la línea 26 del archivo [callback_graph_alu_repetidores_nuevos_docente.py](#).

```

00026 def update_graph_docente(asignaturas, curso_academico, docente_id):
00027     """
00028     Actualiza el gráfico de evolución de alumnos de nuevo ingreso y repetidores
00029     del perfil "Docente" de la pestaña "Rendimiento académico personal".
00030
00031     Args:
00032         asignaturas (list): Lista con las asignaturas seleccionadas
00033         curso_academico (list): Lista con los cursos académicos
00034         docente_id (str): Identificador del docente
00035
00036     Returns:
00037         go.Figure: Figura con el gráfico
00038     """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     barmode="stack",
00044     title={"text": "Evolución alumnos de nuevo ingreso y repetidores", "x": 0.5},
00045     xaxis={"title": "Curso académico"},
00046     yaxis={"title": "Nº Alumnos matriculados"},
00047     showlegend=True,
00048     legend={"orientation": "h", "x": 0, "y": 1.1},
00049 )
00050
00051 if not (asignaturas and curso_academico and docente_id):
00052     return fig
00053
00054 try:
00055     curso_academico = list_to_tuple(curso_academico)
00056 except Exception as e:
00057     return fig
00058
00059 data = alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas)
00060
00061 if not data:
00062     return fig
00063
00064 df = pd.DataFrame(
00065     data,
00066     columns=["curso_academico", "alumnos_repetidores", "alumnos_nuevo_ingreso"],
00067 )
00068
00069 fig.add_trace(
00070     go.Bar(
00071         x=df["curso_academico"],
00072         y=df["alumnos_repetidores"],
00073         name="Alumnos repetidores",
00074         marker_color="red",
00075         opacity=0.7,
00076     )
00077 )
00078
00079 fig.add_trace(
00080     go.Bar(
00081         x=df["curso_academico"],
00082         y=df["alumnos_nuevo_ingreso"],
00083         name="Alumnos de primera matrícula",
00084         marker_color="green",
00085         opacity=0.7,
00086     )
00087 )
00088
00089 return fig

```

Gráfico de llamadas para esta función:



4.42. Referencia del Namespace callbacks.docente.utils

Namespaces

- namespace [callback_resumen_docente](#)
- namespace [callback_select_docente](#)
- namespace [callback_tabs_docente](#)

4.43. Referencia del Namespace callbacks.docente.utils.callback_resumen_docente

Funciones

- def [update_resumen_docente](#) (docente_id, titulacion)
- def [not_data](#) ()

4.43.1. Documentación de las funciones

4.43.1.1. not_data()

```
def callbacks.docente.utils.callback_resumen_docente.not_data ( )
```

Definición en la línea 53 del archivo [callback_resumen_docente.py](#).

```
00053 def not_data():
00054     return html.Div(
00055         [
00056             html.H2("Resumen"),
00057             html.P("Universidad:", className="resumen-label"),
00058             html.P("No disponible"),
00059             html.P("Titulación:", className="resumen-label"),
00060             html.P("No disponible"),
00061             html.P("Docente:", className="resumen-label"),
00062             html.P("No disponible"),
00063             html.Hr(),
00064         ]
00065     )
```

Gráfico de llamadas a esta función:



4.43.1.2. update_resumen_docente()

```
def callbacks.docente.utils.callback_resumen_docente.update_resumen_docente (
    docente_id,
    titulacion )
```

Actualiza el resumen del docente

Args:

docente_id (str): ID del docente
titulacion (str): Titulación seleccionada

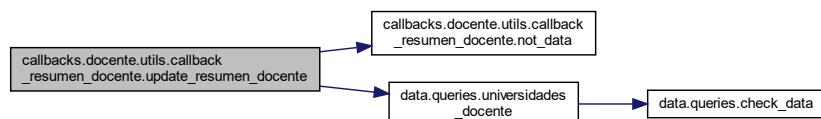
Returns:

html.Div: Layout del resumen del docente

Definición en la línea 20 del archivo **callback_resumen_docente.py**.

```
00020 def update_resumen_docente(docente_id, titulacion):
00021     """
00022     Actualiza el resumen del docente
00023
00024     Args:
00025         docente_id (str): ID del docente
00026         titulacion (str): Titulación seleccionada
00027
00028     Returns:
00029         html.Div: Layout del resumen del docente
00030     """
00031     if not (docente_id and titulacion):
00032         return not_data()
00033
00034     data = universidades_docente(docente_id)
00035
00036     if not data:
00037         return not_data()
00038
00039     return html.Div(
00040         [
00041             html.H2("Resumen"),
00042             html.P("Universidad:", className="resumen-label"),
00043             html.P(data[0][1]),
00044             html.P("Titulación:", className="resumen-label"),
00045             html.P(titulacion),
00046             html.P("Docente:", className="resumen-label"),
00047             html.P(docente_id),
00048             html.Hr(),
00049         ]
00050     )
00051
00052
```

Gráfico de llamadas para esta función:



4.44. Referencia del Namespace callbacks.docente.utils.callback_select_docente

Funciones

- def **store_selected_docente** (selected_value, stored_value)

4.44.1. Documentación de las funciones

4.44.1.1. store_selected_docente()

```
def callbacks.docente.utils.callback_select_docente.store_selected_docente (
    selected_value,
    stored_value )
```

Almacena el docente seleccionado en el store.

Args:

 selected_value (str): Valor seleccionado
 stored_value (str): Valor almacenado

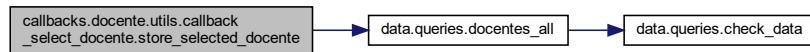
Returns:

 str: Valor seleccionado
 list: Opciones del dropdown
 str: Valor almacenado

Definición en la línea 22 del archivo [callback_select_docente.py](#).

```
00022 def store_selected_docente(selected_value, stored_value):
00023     """
00024 Almacena el docente seleccionado en el store.
00025
00026 Args:
00027     selected_value (str): Valor seleccionado
00028     stored_value (str): Valor almacenado
00029
00030 Returns:
00031     str: Valor seleccionado
00032     list: Opciones del dropdown
00033     str: Valor almacenado
00034 """
00035
00036 data = docentes_all()
00037
00038 if not data:
00039     return None, [], None
00040
00041     opciones_dropdown = [{"label": docente[0], "value": docente[0]} for docente in data]
00042
00043     if selected_value is None and stored_value in [
00044         op["value"] for op in opciones_dropdown
00045     ]:
00046         return stored_value, opciones_dropdown, stored_value
00047
00048     return selected_value, opciones_dropdown, selected_value
```

Gráfico de llamadas para esta función:



4.45. Referencia del Namespace **callbacks.docente.utils.callback_tabs_docente**

Funciones

- def [render_content](#) (tab, selected_docente)

4.45.1. Documentación de las funciones

4.45.1.1. render_content()

```
def callbacks.docente.utils.callback_tabs_docente.render_content (
    tab,
    selected_docente )
```

Renderiza el contenido de las pestañas del dashboard del docente.

Args:

```
    tab (str): Pestaña seleccionada
    selected_docente (str): Docente seleccionado
```

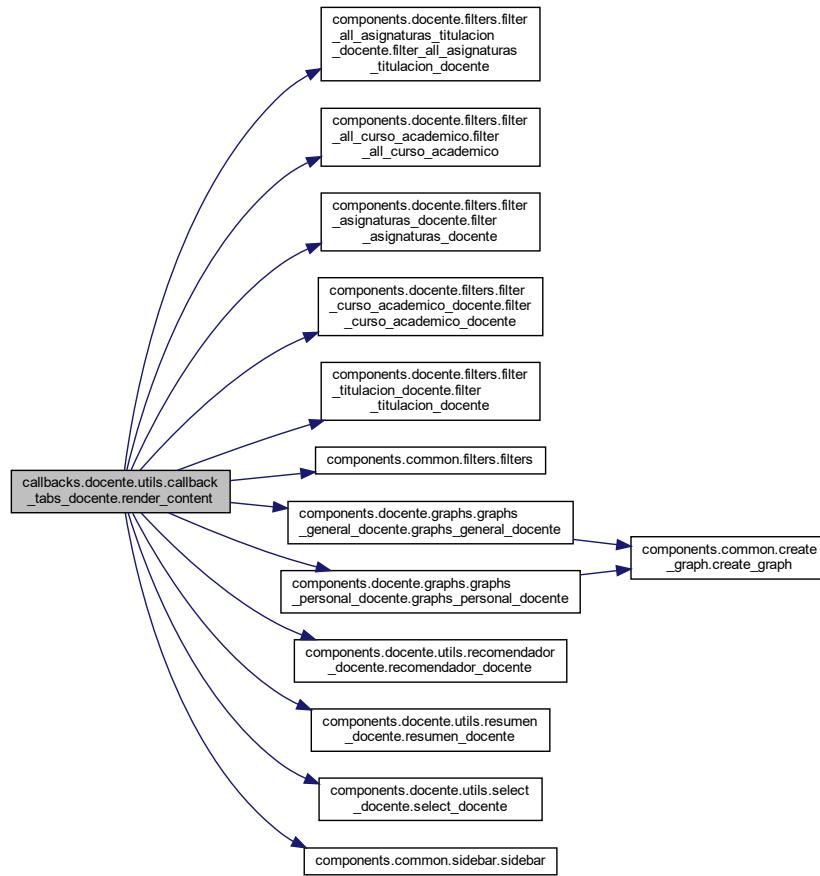
Returns:

```
    ist: Componentes de la pestaña seleccionada
```

Definición en la línea 32 del archivo [callback_tabs_docente.py](#).

```
00032 def render_content(tab, selected_docente):
00033     """
00034     Renderiza el contenido de las pestañas del dashboard del docente.
00035
00036     Args:
00037         tab (str): Pestaña seleccionada
00038         selected_docente (str): Docente seleccionado
00039
00040     Returns:
00041         ist: Componentes de la pestaña seleccionada
00042     """
00043
00044     if tab == 'rendimiento-academico-asignatura-tab':
00045         return html.Div([
00046             select_docente(),
00047             html.H2("Dashboard Docente", style={'text-align': 'center'}),
00048             html.Div([
00049                 sidebar([
00050                     resumen_docente(),
00051                     filters([
00052                         filter_titulacion_docente(),
00053                         filter_asignaturas_docente(),
00054                         filter_curso_academico_docente()
00055                     ]),
00056                 ]),
00057                 graphs_personal_docente()
00058             ], className='content-layout-dashboard')
00059         ])
00060     elif tab == 'rendimiento-academico-tab':
00061         return html.Div([
00062             html.H2("Dashboard Docente", style={'text-align': 'center'}),
00063             html.Div([
00064                 sidebar([
00065                     resumen_docente(),
00066                     filters([
00067                         filter_titulacion_docente(),
00068                         filter_all_curso_academico(),
00069                         filter_all_asignaturas_titulacion_docente()
00070                     ]),
00071                 ]),
00072                 graphs_general_docente()
00073             ], className='content-layout-dashboard')
00074         ])
00075     elif tab == 'recomendaciones-tab':
00076         return html.Div([
00077             recomendador_docente()
00078         ])
```

Gráfico de llamadas para esta función:



4.46. Referencia del Namespace callbacks.gestor

Namespaces

- namespace [filters](#)
- namespace [graphs](#)
- namespace [utils](#)

4.47. Referencia del Namespace callbacks.gestor.filters

Namespaces

- namespace [callback_filter_all_curso_academico_gestor](#)
- namespace [callback_filter_curso_academico_gestor](#)
- namespace [callback_filter_titulaciones_gestor](#)

4.48. Referencia del Namespace callbacks.gestor.filters.callback_filter_all_curso_academico_gestor

Funciones

- def update_filter_all_curso_academico_gestor (gestor_id, n_clicks, existing_options)

4.48.1. Documentación de las funciones

4.48.1.1. update_filter_all_curso_academico_gestor()

```
def callbacks.gestor.filters.callback_filter_all_curso_academico_gestor.update_filter_all_←
curso_academico_gestor (
    gestor_id,
    n_clicks,
    existing_options )
```

Actualiza las opciones del filtro de todos los cursos académicos del perfil "Gestor" en la pestaña "Riesgo académico".

Args:

 gestor_id (str): ID del gestor seleccionado
 n_clicks (int): Número de clicks en el botón "Seleccionar todo"
 existing_options (list): Opciones actuales del filtro de curso académico

Returns:

 list: Opciones del dropdown
 list: Valores seleccionados

Definición en la línea 23 del archivo [callback_filter_all_curso_academico_gestor.py](#).

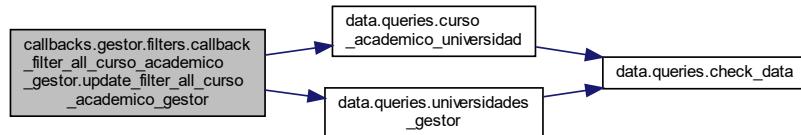
```
00023 def update_filter_all_curso_academico_gestor(gestor_id, n_clicks, existing_options):
00024     """
00025     Actualiza las opciones del filtro de todos los cursos académicos del perfil "Gestor"
00026     en la pestaña "Riesgo académico".
00027
00028     Args:
00029         gestor_id (str): ID del gestor seleccionado
00030         n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00031         existing_options (list): Opciones actuales del filtro de curso académico
00032
00033     Returns:
00034         list: Opciones del dropdown
00035         list: Valores seleccionados
00036     """
00037
00038 ctx = callback_context
00039 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00040
00041 if trigger_id == "select-all-curso-academico-button" and n_clicks > 0:
00042     return existing_options, [option["value"] for option in existing_options]
00043
00044 if not gestor_id:
00045     return [], None
00046
00047 cod_universidad = universidades_gestor(gestor_id)
00048
00049 if not cod_universidad:
00050     return [], None
00051
00052 data = curso_academico_universidad(cod_universidad[0][0])
00053
00054 if not data:
00055     return [], None
```

```

00056
00057     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00058     value = (
00059         [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
00060     )
00061
00062     return opciones_dropdown, value

```

Gráfico de llamadas para esta función:



4.49. Referencia del Namespace

callbacks.gestor.filters.callback_filter_curso_academico_gestor

Funciones

- def [update_filter_curso_academico_gestor](#) (gestor_id)

4.49.1. Documentación de las funciones

4.49.1.1. update_filter_curso_academico_gestor()

```

def callbacks.gestor.filters.callback_filter_curso_academico_gestor.update_filter_curso_academico_gestor (
    gestor_id )

```

Actualiza las opciones del filtro de curso académico del perfil "Gestor" en la pestaña "Indicadores académicos".

Args:
gestor_id (str): ID del gestor seleccionado

Returns:
list: Opciones del dropdown
str: Valor seleccionado

Definición en la línea 22 del archivo [callback_filter_curso_academico_gestor.py](#).

```

00022 def update_filter_curso_academico_gestor(gestor_id):
00023     """
00024     Actualiza las opciones del filtro de curso académico del perfil "Gestor"
00025     en la pestaña "Indicadores académicos".
00026
00027     Args:
00028         gestor_id (str): ID del gestor seleccionado
00029

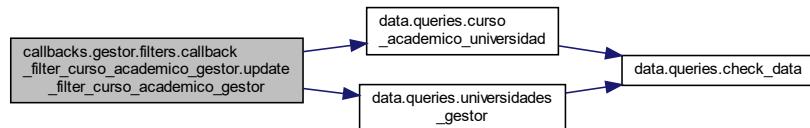
```

```

00030     Returns:
00031         list: Opciones del dropdown
00032         str: Valor seleccionado
00033     """
00034
00035     if not gestor_id:
00036         return [], None
00037
00038     cod_universidad = universidades_gestor(gestor_id)
00039
00040     if not cod_universidad:
00041         return [], None
00042
00043     data = curso_academico_universidad(cod_universidad[0][0])
00044
00045     if not data:
00046         return [], None
00047
00048     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00049     value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00050
00051     return opciones_dropdown, value

```

Gráfico de llamadas para esta función:



4.50. Referencia del Namespace callbacks.gestor.filters.callback_filter_titulaciones_gestor

Funciones

- def [update_filter_titulaciones_gestor](#) (docente_id, curso_academico, n_clicks, existing_options)

4.50.1. Documentación de las funciones

4.50.1.1. update_filter_titulaciones_gestor()

```

def callbacks.gestor.filters.callback_filter_titulaciones_gestor.update_filter_titulaciones_gestor (
    docente_id,
    curso_academico,
    n_clicks,
    existing_options )

```

Actualiza las opciones del filtro de titulaciones en base a la universidad y curso académico seleccionados. También gestiona el evento que permite seleccionar todas las opciones del filtro del perfil "Gestor" en la pestaña "Indicadores académicos".

Args:

```
docente_id (str): ID del docente seleccionado
curso_academico (str): Curso académico seleccionado
n_clicks (int): Número de clicks en el botón "Seleccionar todo"
existing_options (list): Opciones actuales del filtro de titulaciones
```

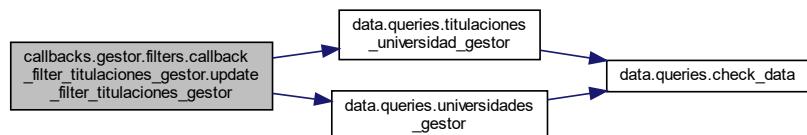
Returns:

```
list: Opciones del dropdown
list: Valores seleccionados
```

Definición en la línea 25 del archivo [callback_filter_titulaciones_gestor.py](#).

```
00025 def update_filter_titulaciones_gestor(docente_id, curso_academico, n_clicks, existing_options):
00026     """
00027     Actualiza las opciones del filtro de titulaciones en base a la universidad y curso académico
00028     seleccionados.
00029     También gestiona el evento que permite seleccionar todas las opciones del filtro del perfil "Gestor"
00030     en
00031     la pestaña "Indicadores académicos".
00032     Args:
00033         docente_id (str): ID del docente seleccionado
00034         curso_academico (str): Curso académico seleccionado
00035         n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00036         existing_options (list): Opciones actuales del filtro de titulaciones
00037     Returns:
00038         list: Opciones del dropdown
00039         list: Valores seleccionados
00040     """
00041 ctx = callback_context
00042 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00043
00044     if trigger_id == "select-all-titulaciones-gestor" and n_clicks > 0:
00045         return existing_options, [option["value"] for option in existing_options]
00046
00047     if not (docente_id and curso_academico):
00048         return [], None
00049
00050 cod_universidad = universidades_gestor(docente_id)
00051
00052 if not cod_universidad:
00053     return [], None
00054
00055 data = titulaciones_universidad_gestor(cod_universidad[0][0], curso_academico)
00056
00057 if not data:
00058     return [], None
00059
00060 opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00061 value = (
00062     [option["value"] for option in opciones_dropdown] if opciones_dropdown else None
00063 )
00064
00065 return opciones_dropdown, value
```

Gráfico de llamadas para esta función:



4.51. Referencia del Namespace callbacks.gestor.graphs

Namespaces

- namespace [indicadores](#)
- namespace [resultados](#)
- namespace [riesgo_abandono](#)

4.52. Referencia del Namespace callbacks.gestor.graphs.indicadores

Namespaces

- namespace [callback_graph_egresados_genero_gestor](#)
- namespace [callback_graph_egresados_nacionalidad_gestor](#)
- namespace [callback_graph_nuevo_ingreso_genero_gestor](#)
- namespace [callback_graph_nuevo_ingreso_nacionalidad_gestor](#)

4.53. Referencia del Namespace callbacks.gestor.graphs.indicadores. ↵ callback_graph_egresados_genero_gestor

Funciones

- def [update_graph_gestor](#) (gestor_id, curso_academico, titulaciones)
- def [toggle_modal](#) (btn, is_open)
- def [update_table](#) (btn, gestor_id, curso_academico, titulaciones)
- def [generate_csv](#) (btn, gestor_id, curso_academico, titulaciones)
- def [get_data](#) (gestor_id, curso_academico, titulaciones)

4.53.1. Documentación de las funciones

4.53.1.1. generate_csv()

```
def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.generate_csv (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )
```

Genera un archivo CSV descargable con los datos de alumnos egresados por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

btn (int): Número de clicks en el botón
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

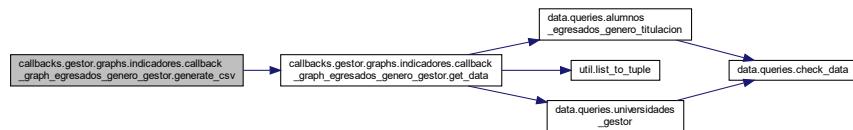
Returns:

str: Enlace al archivo CSV

Definición en la línea 163 del archivo [callback_graph_egresados_genero_gestor.py](#).

```
00163 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00164     """
00165     Genera un archivo CSV descargable con los datos de alumnos egresados por género
00166     y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00167
00168     Args:
00169         btn (int): Número de clicks en el botón
00170         gestor_id (str): ID del gestor seleccionado
00171         curso_academico (list): Lista con los cursos académicos
00172         titulaciones (list): Lista con las titulaciones seleccionadas
00173
00174     Returns:
00175         str: Enlace al archivo CSV
00176
00177     """
00178
00179 if not btn:
00180     return ""
00181
00182 df = get_data(gestor_id, curso_academico, titulaciones)
00183
00184 if df.empty:
00185     return ""
00186
00187 csv_string = df.to_csv(index=False, encoding="utf-8")
00188 csv_string = "data:text/csv; charset=utf-8," + csv_string
00189 return csv_string
00190
00191
```

Gráfico de llamadas para esta función:



4.53.1.2. `get_data()`

```
def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.get_data (
    gestor_id,
    curso_academico,
    titulaciones )
```

Obtiene los datos de la base de datos.

Args:

`gestor_id` (str): ID del gestor seleccionado
`curso_academico` (list): Lista con los cursos académicos
`titulaciones` (list): Lista con las titulaciones seleccionadas

Returns:

`pd.DataFrame`: Datos obtenidos de la base de datos

Definición en la línea 192 del archivo [callback_graph_egresados_genero_gestor.py](#).

```
00192 def get_data(gestor_id, curso_academico, titulaciones):
00193     """
00194     Obtiene los datos de la base de datos.
00195
00196     Args:
00197         gestor_id (str): ID del gestor seleccionado
```

```

00198 curso_academico (list): Lista con los cursos académicos
00199 titulaciones (list): Lista con las titulaciones seleccionadas
00200
00201 Returns:
00202 pd.DataFrame: Datos obtenidos de la base de datos
00203 """
00204 empty = pd.DataFrame()
00205
00206 if not gestor_id or not curso_academico or not titulaciones:
00207     return empty
00208
00209     try:
00210         titulaciones = list_to_tuple(titulaciones)
00211     except Exception as e:
00212         return empty
00213
00214 data_universidad = universidades_gestor(gestor_id)
00215 if not data_universidad:
00216     return empty
00217
00218 data = alumnos_egresados_genero_titulacion(
00219     data_universidad[0][0], curso_academico, titulaciones
00220 )
00221
00222 if not data:
00223     return empty
00224
00225 return pd.DataFrame(data, columns=["titulacion", "genero", "cantidad"])

```

Gráfico de llamadas para esta función:

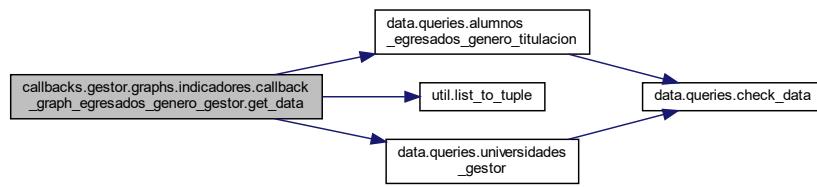
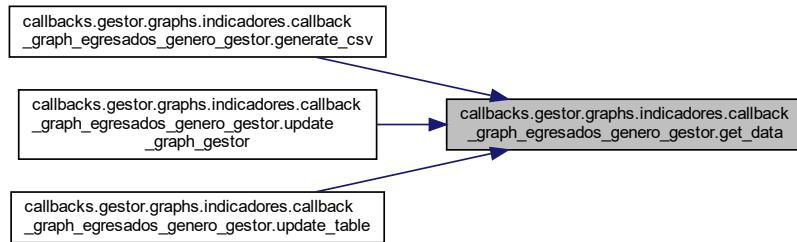


Gráfico de llamadas a esta función:



4.53.1.3. toggle_modal()

```

def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.toggle_modal (
    btn,
    is_open )

```

Altera la visibilidad del modal con los datos de alumnos egresados por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:
 btn (int): Número de clicks en el botón
 is_open (bool): Estado actual del modal

Returns:
 bool: Nuevo estado del modal

Definición en la línea 101 del archivo [callback_graph_egresados_genero_gestor.py](#).

```
00101 def toggle_modal(btn, is_open):
00102     """
00103 Alterna la visibilidad del modal con los datos de alumnos egresados por género
00104 y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00105
00106     Args:
00107         btn (int): Número de clicks en el botón
00108         is_open (bool): Estado actual del modal
00109
00110     Returns:
00111         bool: Nuevo estado del modal
00112
00113     """
00114
00115 if btn:
00116     return not is_open
00117 return is_open
00118
00119
00120 @callback(
00121     Output("table-container-egresados-genero", "children"),
00122     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00123     Input("selected-gestor-store", "data"),
00124     Input("curso-academico-gestor", "value"),
00125     Input("titulaciones-gestor", "value"),
00126 )
```

4.53.1.4. update_graph_gestor()

```
def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.update_graph_←
gestor (
        gestor_id,
        curso_academico,
        titulaciones )
```

Actualiza el gráfico de alumnos egresados por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

Returns:
 go.Figure: Figura con el gráfico

Definición en la línea 27 del archivo [callback_graph_egresados_genero_gestor.py](#).

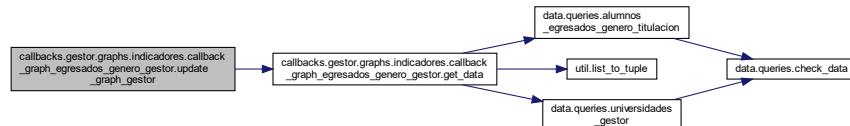
```
00027 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00028     """
00029 Actualiza el gráfico de alumnos egresados por género y titulación
00030 del perfil "Gestor" de la pestaña "Indicadores académicos".
00031
00032     Args:
00033         gestor_id (str): ID del gestor seleccionado
00034         curso_academico (list): Lista con los cursos académicos
00035         titulaciones (list): Lista con las titulaciones seleccionadas
00036 
```

```

00037     Returns:
00038         go.Figure: Figura con el gráfico
00039
00040     """
00041 fig = go.Figure()
00042
00043 fig.update_layout(
00044     barmode="stack",
00045     title={"text": "Alumnos egresados por género y titulación", "x": 0.5},
00046     xaxis=dict(title="Titulaciones"),
00047     yaxis=dict(title="Nº Alumnos"),
00048     showlegend=True,
00049     legend={"title": "Género"},
00050 )
00051
00052 df = get_data(gestor_id, curso_academico, titulaciones)
00053
00054 if df.empty:
00055     return fig
00056
00057 def map_genero(genero):
00058     if "masculin" in genero.lower():
00059         return "Hombres"
00060     elif "fem" in genero.lower():
00061         return "Mujeres"
00062     return genero
00063
00064 df["genero"] = df["genero"].map(map_genero)
00065
00066 # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
00067 df_pivot = df.pivot_table(
00068     index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
00069 ).fillna(0)
00070
00071 if "Hombres" in df_pivot.columns:
00072     fig.add_trace(
00073         go.Bar(
00074             x=df_pivot.index,
00075             y=df_pivot["Hombres"],
00076             name="Hombres",
00077             marker_color="blue",
00078             opacity=0.7,
00079         )
00080     )
00081
00082 if "Mujeres" in df_pivot.columns:
00083     fig.add_trace(
00084         go.Bar(
00085             x=df_pivot.index,
00086             y=df_pivot["Mujeres"],
00087             name="Mujeres",
00088             marker_color="red",
00089             opacity=0.7,
00090         )
00091     )
00092
00093 return fig
00094
00095
00096 @callback(
00097     Output("modal-egresados-genero", "is_open"),
00098     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00099     State("modal-egresados-genero", "is_open"),
00100 )

```

Gráfico de llamadas para esta función:



4.53.1.5. update_table()

```
def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.update_table (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )
```

Actualiza la tabla con los datos de alumnos egresados por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

btn (int): Número de clicks en el botón
gestor_id (str): ID del gestor seleccionado
curso_academico (list): Lista con los cursos académicos
titulaciones (list): Lista con las titulaciones seleccionadas

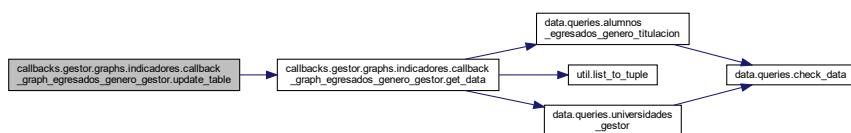
Returns:

dbc.Table: Tabla con los datos

Definición en la línea 127 del archivo [callback_graph_egresados_genero_gestor.py](#).

```
00127 def update_table(btn, gestor_id, curso_academico, titulaciones):
00128     """
00129     Actualiza la tabla con los datos de alumnos egresados por género y titulación
00130     del perfil "Gestor" de la pestaña "Indicadores académicos".
00131
00132     Args:
00133         btn (int): Número de clicks en el botón
00134         gestor_id (str): ID del gestor seleccionado
00135         curso_academico (list): Lista con los cursos académicos
00136         titulaciones (list): Lista con las titulaciones seleccionadas
00137
00138     Returns:
00139         dbc.Table: Tabla con los datos
00140
00141     """
00142
00143 if not btn:
00144     return ""
00145
00146 df = get_data(gestor_id, curso_academico, titulaciones)
00147
00148 if df.empty:
00149     return dbc.Alert("No hay datos disponibles", color="info")
00150
00151 return dbc.Table.from_dataframe(
00152     df.head(50), striped=True, bordered=True, hover=True
00153 )
00154
00155
00156 @callback(
00157     Output("btn-descargar-egresados-genero", "href"),
00158     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00159     State("selected-gestor-store", "data"),
00160     State("curso-academico-gestor", "value"),
00161     State("titulaciones-gestor", "value"),
00162 )
```

Gráfico de llamadas para esta función:



4.54. Referencia del Namespace callbacks.gestor.graphs.indicadores.← callback_graph_egresados_nacionalidad_gestor

Funciones

- def `update_graph_gestor` (gestor_id, curso_academico, titulaciones)
- def `toggle_modal` (btn, is_open)
- def `update_table` (btn, gestor_id, curso_academico, titulaciones)
- def `generate_csv` (btn, gestor_id, curso_academico, titulaciones)
- def `get_data` (gestor_id, curso_academico, titulaciones)

4.54.1. Documentación de las funciones

4.54.1.1. generate_csv()

```
def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.generate←
_csv (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )
```

Genera un archivo CSV descargable con los datos de alumnos egresados por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

`btn` (int): Número de clicks en el botón
`gestor_id` (str): ID del gestor seleccionado
`curso_academico` (list): Lista con los cursos académicos
`titulaciones` (list): Lista con las titulaciones seleccionadas

Returns:

`str`: Enlace al archivo CSV

Definición en la línea 133 del archivo `callback_graph_egresados_nacionalidad_gestor.py`.

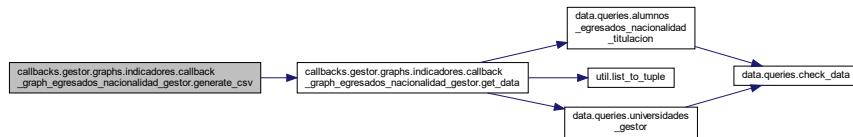
```
00133 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00134     """
00135     Genera un archivo CSV descargable con los datos de alumnos egresados por nacionalidad
00136     y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00137
00138     Args:
00139         btn (int): Número de clicks en el botón
00140         gestor_id (str): ID del gestor seleccionado
00141         curso_academico (list): Lista con los cursos académicos
00142         titulaciones (list): Lista con las titulaciones seleccionadas
00143
00144     Returns:
00145         str: Enlace al archivo CSV
00146     """
00147
00148 if not btn:
00149     return ""
00150
00151 df = get_data(gestor_id, curso_academico, titulaciones)
00152
00153 if df.empty:
00154     return ""
00155
00156 csv_string = df.to_csv(index=False, encoding="utf-8")
00157 csv_string = "data:text/csv;charset=utf-8," + csv_string
```

```

00158     return csv_string
00159
00160

```

Gráfico de llamadas para esta función:



4.54.1.2. get_data()

```

def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.get_data
(
    gestor_id,
    curso_academico,
    titulaciones )

```

Obtiene los datos de alumnos egresados por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

gestor_id (str): ID del gestor seleccionado
curso_academico (list): Lista con los cursos académicos
titulaciones (list): Lista con las titulaciones seleccionadas

Returns:

pd.DataFrame: Datos de la consulta

Definición en la línea 161 del archivo [callback_graph_egresados_nacionalidad_gestor.py](#).

```

00161 def get_data(gestor_id, curso_academico, titulaciones):
00162     """
00163     Obtiene los datos de alumnos egresados por nacionalidad y titulación
00164     del perfil "Gestor" de la pestaña "Indicadores académicos".
00165
00166     Args:
00167         gestor_id (str): ID del gestor seleccionado
00168         curso_academico (list): Lista con los cursos académicos
00169         titulaciones (list): Lista con las titulaciones seleccionadas
00170
00171     Returns:
00172         pd.DataFrame: Datos de la consulta
00173     """
00174
00175     empty = pd.DataFrame()
00176
00177     if not gestor_id or not curso_academico or not titulaciones:
00178         return empty
00179
00180     data_universidad = universidades_gestor(gestor_id)
00181     if not data_universidad:
00182         return empty
00183
00184     try:
00185         titulaciones = list_to_tuple(titulaciones)
00186     except Exception as e:
00187         return empty
00188
00189     data = alumnos_egresados_nacionalidad_titulacion(
00190         data_universidad[0][0], curso_academico, titulaciones
00191     )

```

4.54 Referencia del Namespace

callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor

73

```
00192     if not data:  
00193         return empty  
00194  
00195     return pd.DataFrame(data, columns=["titulacion", "nacionalidad", "cantidad"])
```

Gráfico de llamadas para esta función:

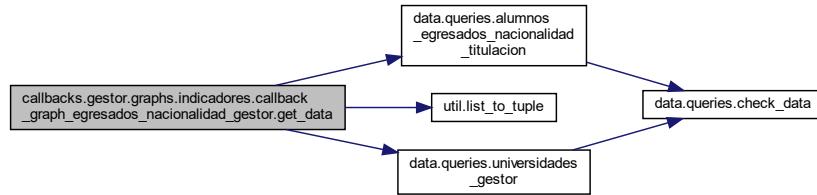
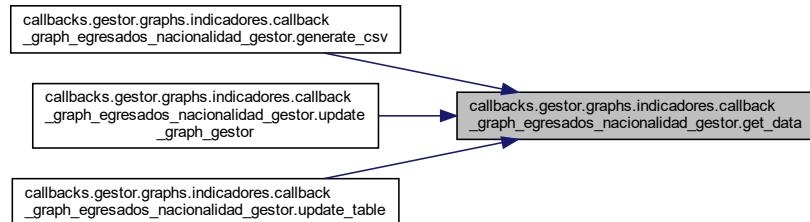


Gráfico de llamadas a esta función:



4.54.1.3. toggle_modal()

```
def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.toggle_modal (   
    btn,  
    is_open )
```

Altera la visibilidad del modal con los datos de alumnos egresados por nacionalidad

Args:

`btn` (int): Número de clicks en el botón
`is_open` (bool): Estado actual del modal

Returns:

`bool`: Nuevo estado del modal

Definición en la línea 75 del archivo [callback_graph_egresados_nacionalidad_gestor.py](#).

```
00075 def toggle_modal(btn, is_open):  
00076     """  
00077     Alterna la visibilidad del modal con los datos de alumnos egresados por nacionalidad  
00078
```

```

00079 Args:
00080 btn (int): Número de clicks en el botón
00081 is_open (bool): Estado actual del modal
00082
00083 Returns:
00084 bool: Nuevo estado del modal
00085 """
00086
00087 if btn:
00088     return not is_open
00089 return is_open
00090
00091
00092 @callback(
00093     Output("table-container-egresados-nacionalidad", "children"),
00094     Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
00095     State("selected-gestor-store", "data"),
00096     Input("curso-academico-gestor", "value"),
00097     Input("titulaciones-gestor", "value"),
00098 )

```

4.54.1.4. update_graph_gestor()

```

def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.update_←
graph_gestor (
    gestor_id,
    curso_academico,
    titulaciones )

```

Actualiza el gráfico de alumnos egresados por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

```

gestor_id (str): ID del gestor seleccionado
curso_academico (list): Lista con los cursos académicos
titulaciones (list): Lista con las titulaciones seleccionadas

```

Returns:

```

go.Figure: Figura con el gráfico

```

Definición en la línea 27 del archivo [callback_graph_egresados_nacionalidad_gestor.py](#).

```

00027 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00028     """
00029 Actualiza el gráfico de alumnos egresados por nacionalidad y titulación
00030 del perfil "Gestor" de la pestaña "Indicadores académicos".
00031
00032     Args:
00033         gestor_id (str): ID del gestor seleccionado
00034         curso_academico (list): Lista con los cursos académicos
00035         titulaciones (list): Lista con las titulaciones seleccionadas
00036
00037     Returns:
00038         go.Figure: Figura con el gráfico
00039
00040     """
00041
00042 fig = go.Figure()
00043
00044 fig.update_layout(
00045     barmode="stack",
00046     title={"text": "Alumnos egresados por nacionalidad y titulación", "x": 0.5},
00047     xaxis=dict(title="Titulaciones"),
00048     yaxis=dict(title="Nº Alumnos"),
00049     showlegend=True,
00050     legend={"title": "Nacionalidad"},
00051 )
00052
00053 df = get_data(gestor_id, curso_academico, titulaciones)
00054
00055 if df.empty:
00056     return fig
00057

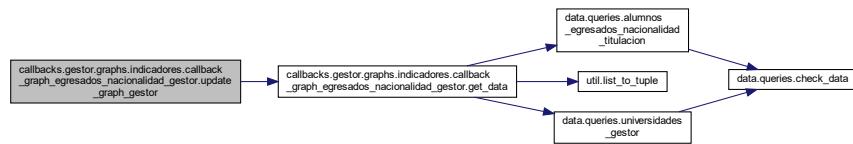
```

```

00058     df_pivot = df.pivot_table(
00059         index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
00060     ).fillna(0)
00061
00062     for nacionalidad in df_pivot.columns:
00063         fig.add_trace(
00064             go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
00065         )
00066
00067     return fig
00068
00069
00070 @callback(
00071     Output("modal-egresados-nacionalidad", "is_open"),
00072     Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
00073     State("modal-egresados-nacionalidad", "is_open"),
00074 )

```

Gráfico de llamadas para esta función:



4.54.1.5. update_table()

```

def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.update_table (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )

```

Actualiza la tabla con los datos de alumnos egresados por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

btn (int): Número de clicks en el botón
gestor_id (str): ID del gestor seleccionado
curso_academico (list): Lista con los cursos académicos

Returns:

dbc.Table: Tabla con los datos

Definición en la línea 99 del archivo [callback_graph_egresados_nacionalidad_gestor.py](#).

```

00099 def update_table(btn, gestor_id, curso_academico, titulaciones):
00100     """
00101 Actualiza la tabla con los datos de alumnos egresados por nacionalidad y titulación
00102 del perfil "Gestor" de la pestaña "Indicadores académicos".
00103
00104     Args:
00105         btn (int): Número de clicks en el botón
00106         gestor_id (str): ID del gestor seleccionado
00107         curso_academico (list): Lista con los cursos académicos
00108
00109     Returns:
00110         dbc.Table: Tabla con los datos
00111     """
00112
00113 if not btn:

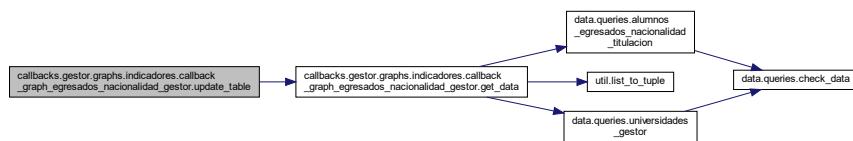
```

```

00114         return ""
00115
00116     df = get_data(gestor_id, curso_academico, titulaciones)
00117
00118     if df.empty:
00119         return dbc.Alert("No hay datos disponibles", color="info")
00120
00121     return dbc.Table.from_dataframe(
00122         df.head(50), striped=True, bordered=True, hover=True
00123     )
00124
00125
00126 @callback(
00127     Output("btn-descargar-egresados-nacionalidad", "href"),
00128     Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
00129     State("selected-gestor-store", "data"),
00130     Input("curso-academico-gestor", "value"),
00131     Input("titulaciones-gestor", "value"),
00132 )

```

Gráfico de llamadas para esta función:



4.55. Referencia del Namespace callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor

Funciones

- def [update_graph_gestor](#) (gestor_id, curso_academico, titulaciones)
- def [toggle_modal](#) (btn, is_open)
- def [update_table](#) (btn, gestor_id, curso_academico, titulaciones)
- def [generate_csv](#) (btn, gestor_id, curso_academico, titulaciones)
- def [get_data](#) (gestor_id, curso_academico, titulaciones)

4.55.1. Documentación de las funciones

4.55.1.1. generate_csv()

```

def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.generate_csv (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )

```

Genera un archivo CSV descargable con los datos de los alumnos de nuevo ingreso por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

btn (int): Número de clicks en el botón
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

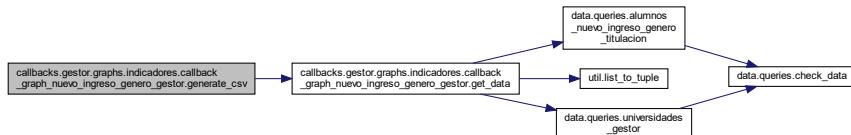
Returns:

str: Contenido del archivo CSV

Definición en la línea 161 del archivo [callback_graph_nuevo_ingreso_genero_gestor.py](#).

```
00161 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00162     """
00163     Genera un archivo CSV descargable con los datos de los alumnos de nuevo ingreso
00164     por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00165
00166     Args:
00167         btn (int): Número de clicks en el botón
00168         gestor_id (str): ID del gestor seleccionado
00169         curso_academico (list): Lista con los cursos académicos
00170         titulaciones (list): Lista con las titulaciones seleccionadas
00171
00172     Returns:
00173         str: Contenido del archivo CSV
00174     """
00175
00176     if not btn:
00177         return ""
00178
00179     df = get_data(gestor_id, curso_academico, titulaciones)
00180     if df.empty:
00181         return ""
00182
00183     csv_string = df.to_csv(index=False, encoding="utf-8")
00184     csv_string = "data:text/csv;charset=utf-8," + csv_string
00185     return csv_string
00186
00187
```

Gráfico de llamadas para esta función:



4.55.1.2. get_data()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.get_data (
    gestor_id,
    curso_academico,
    titulaciones )
```

Obtiene los datos de la base de datos para el gráfico y tabla

Args:

gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

Returns:

pd.DataFrame: Datos para el gráfico y tabla

Definición en la línea 188 del archivo [callback_graph_nuevo_ingreso_genero_gestor.py](#).

```

00188 def get_data(gestor_id, curso_academico, titulaciones):
00189 """
00190 Obtiene los datos de la base da datos para el gráfico y tabla
00191
00192 Args:
00193     gestor_id (str): ID del gestor seleccionado
00194     curso_academico (list): Lista con los cursos académicos
00195     titulaciones (list): Lista con las titulaciones seleccionadas
00196
00197 Returns:
00198     pd.DataFrame: Datos para el gráfico y tabla
00199 """
00200
00201 empty = pd.DataFrame()
00202
00203 if not gestor_id or not curso_academico or not titulaciones:
00204     return empty
00205
00206 try:
00207     titulaciones = list_to_tuple(titulaciones)
00208 except Exception as e:
00209     return empty, None
00210
00211 data_universidad = universidades_gestor(gestor_id)
00212 if not data_universidad:
00213     return empty
00214
00215 data = alumnos_nuevo_ingreso_genero_titulacion(
00216     curso_academico, titulaciones, data_universidad[0][0]
00217 )
00218
00219 if not data:
00220     return empty
00221
00222 return pd.DataFrame(
00223     data, columns=["curso_academico", "titulacion", "genero", "cantidad"]
00224 )

```

Gráfico de llamadas para esta función:

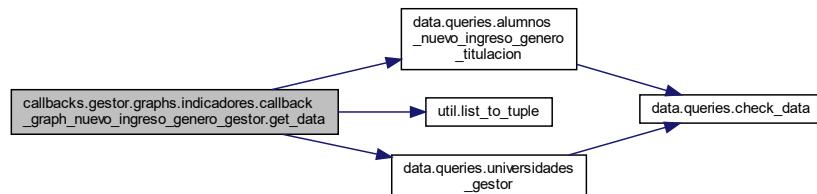
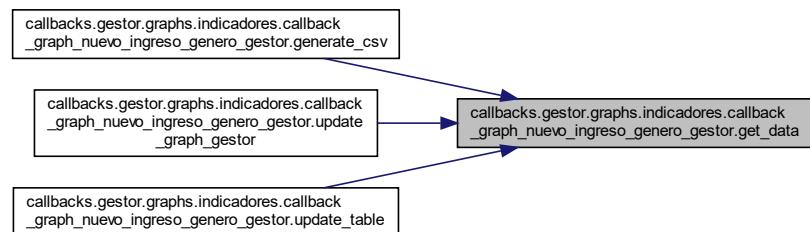


Gráfico de llamadas a esta función:



4.55.1.3. toggle_modal()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.toggle_←
modal (
    btn,
    is_open )
```

Altera la visibilidad del modal que muestra los datos de los alumnos de nuevo ingreso

Args:

btn (int): Número de clicks en el botón
is_open (bool): Estado actual del modal

Returns:

bool: Nuevo estado del modal

Definición en la línea 103 del archivo [callback_graph_nuevo_ingreso_genero_gestor.py](#).

```
00103 def toggle_modal(btn, is_open):
00104     """
00105 Alterna la visibilidad del modal que muestra los datos de los alumnos de nuevo ingreso
00106
00107 Args:
00108     btn (int): Número de clicks en el botón
00109     is_open (bool): Estado actual del modal
00110
00111 Returns:
00112     bool: Nuevo estado del modal
00113 """
00114 if btn:
00115     return not is_open
00116 return is_open
00117
00118
00119 @callback(
00120     Output("table-container-nuevo-ingreso-genero", "children"),
00121     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00122     Input("selected-gestor-store", "data"),
00123     Input("curso-academico-gestor", "value"),
00124     Input("titulaciones-gestor", "value"),
00125 )
```

4.55.1.4. update_graph_gestor()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.update_←
graph_gestor (
    gestor_id,
    curso_academico,
    titulaciones )
```

Actualiza el gráfico de alumnos de nuevo ingreso por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

gestor_id (str): ID del gestor seleccionado
curso_academico (list): Lista con los cursos académicos
titulaciones (list): Lista con las titulaciones seleccionadas

Returns:

go.Figure: Figura con el gráfico

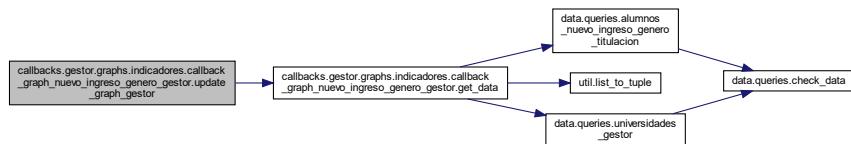
Definición en la línea 27 del archivo [callback_graph_nuevo_ingreso_genero_gestor.py](#).

```

00027 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00028     """
00029     Actualiza el gráfico de alumnos de nuevo ingreso por género y titulación
00030     del perfil "Gestor" de la pestaña "Indicadores académicos".
00031
00032     Args:
00033         gestor_id (str): ID del gestor seleccionado
00034         curso_academico (list): Lista con los cursos académicos
00035         titulaciones (list): Lista con las titulaciones seleccionadas
00036
00037     Returns:
00038         go.Figure: Figura con el gráfico
00039
00040     """
00041 fig = go.Figure()
00042
00043 fig.update_layout(
00044     barmode="stack",
00045     title={"text": "Alumnos de nuevo ingreso por género y titulación", "x": 0.5},
00046     xaxis=dict(title="Titulaciones"),
00047     yaxis=dict(title="Nº Alumnos"),
00048     showlegend=True,
00049     legend={"title": "Género"},
00050 )
00051
00052     # Asegurarse de que las columnas sean consistentes
00053 def map_genero(genero):
00054     if "masculin" in genero.lower():
00055         return "Hombres"
00056     elif "fem" in genero.lower():
00057         return "Mujeres"
00058     return genero
00059
00060 df = get_data(gestor_id, curso_academico, titulaciones)
00061
00062 if df.empty:
00063     return fig
00064
00065 df["genero"] = df["genero"].map(map_genero)
00066
00067     # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
00068 df_pivot = df.pivot_table(
00069     index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
00070 ).fillna(0)
00071
00072     # Asegurarse de que las columnas existen antes de graficar
00073 if "Hombres" in df_pivot.columns:
00074     fig.add_trace(
00075         go.Bar(
00076             x=df_pivot.index,
00077             y=df_pivot["Hombres"],
00078             name="Hombres",
00079             marker_color="blue",
00080             opacity=0.7,
00081         )
00082     )
00083
00084 if "Mujeres" in df_pivot.columns:
00085     fig.add_trace(
00086         go.Bar(
00087             x=df_pivot.index,
00088             y=df_pivot["Mujeres"],
00089             name="Mujeres",
00090             marker_color="red",
00091             opacity=0.7,
00092         )
00093     )
00094
00095 return fig
00096
00097
00098 @callback(
00099     Output("modal-nuevo-ingreso-genero", "is_open"),
00100     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00101     State("modal-nuevo-ingreso-genero", "is_open"),
00102 )

```

Gráfico de llamadas para esta función:



4.55.1.5. update_table()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.update_table (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )
```

Actualiza la tabla de datos de los alumnos de nuevo ingreso por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:
 btn (int): Número de clicks en el botón
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

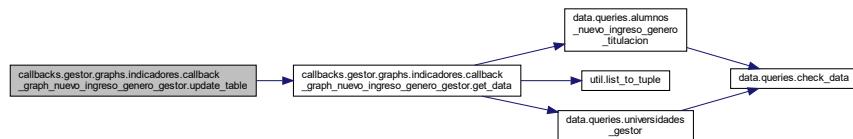
Returns:
 dbc.Table: Tabla con los datos

Definición en la línea 126 del archivo [callback_graph_nuevo_ingreso_genero_gestor.py](#).

```
00126 def update_table(btn, gestor_id, curso_academico, titulaciones):
00127     """
00128     Actualiza la tabla de datos de los alumnos de nuevo ingreso por género y titulación
00129     del perfil "Gestor" de la pestaña "Indicadores académicos".
00130
00131     Args:
00132         btn (int): Número de clicks en el botón
00133         gestor_id (str): ID del gestor seleccionado
00134         curso_academico (list): Lista con los cursos académicos
00135         titulaciones (list): Lista con las titulaciones seleccionadas
00136
00137     Returns:
00138         dbc.Table: Tabla con los datos
00139     """
00140
00141 if not btn:
00142     return ""
00143
00144 df = get_data(gestor_id, curso_academico, titulaciones)
00145
00146 if df.empty:
00147     return dbc.Alert("No hay datos disponibles", color="info")
00148
00149 return dbc.Table.from_dataframe(
00150     df.head(50), striped=True, bordered=True, hover=True
00151 )
00152
00153
00154 @callback(
00155     Output("btn-descargar-nuevo-ingreso-genero", "href"),
00156     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00157     State("selected-gestor-store", "data"),
```

```
00158     State("curso-academico-gestor", "value"),
00159     State("titulaciones-gestor", "value"),
00160 )
```

Gráfico de llamadas para esta función:



4.56. Referencia del Namespace callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor

Funciones

- def [update_graph_gestor](#) (gestor_id, curso_academico, titulaciones)
- def [toggle_modal](#) (btn, is_open)
- def [update_table](#) (btn, gestor_id, curso_academico, titulaciones)
- def [generate_csv](#) (btn, gestor_id, curso_academico, titulaciones)
- def [get_data](#) (gestor_id, curso_academico, titulaciones)

4.56.1. Documentación de las funciones

4.56.1.1. generate_csv()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.generate_csv (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )
```

Genera un archivo CSV descargable con los datos de alumnos de nuevo ingreso por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

btn (int): Número de clicks en el botón "Ver datos"
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

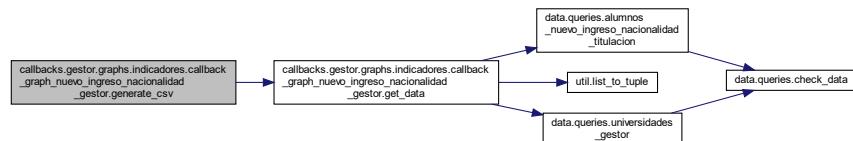
Returns:

str: Enlace al archivo CSV

Definición en la línea 145 del archivo [callback_graph_nuevo_ingreso_nacionalidad_gestor.py](#).

```
00145 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00146     """
00147     Genera un archivo CSV descargable con los datos de alumnos de nuevo ingreso
00148     por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00149
00150     Args:
00151         btn (int): Número de clicks en el botón "Ver datos"
00152         gestor_id (str): ID del gestor seleccionado
00153         curso_academico (list): Lista con los cursos académicos
00154         titulaciones (list): Lista con las titulaciones seleccionadas
00155
00156     Returns:
00157         str: Enlace al archivo CSV
00158
00159     """
00160
00161 if not btn:
00162     return ""
00163
00164 df = get_data(gestor_id, curso_academico, titulaciones)
00165
00166 if df.empty:
00167     return ""
00168
00169 csv_string = df.to_csv(index=False, encoding="utf-8")
00170 csv_string = "data:text/csv;charset=utf-8," + csv_string
00171 return csv_string
00172
00173
```

Gráfico de llamadas para esta función:



4.56.1.2. get_data()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.get_data (
    gestor_id,
    curso_academico,
    titulaciones )
```

Obtiene los datos de la base da datos para el gráfico y tabla.

Args:

gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

Returns:

pd.DataFrame: Datos para el gráfico y tabla

Definición en la línea 174 del archivo [callback_graph_nuevo_ingreso_nacionalidad_gestor.py](#).

```
00174 def get_data(gesto_id, curso_academico, titulaciones):
00175     """
00176     Obtiene los datos de la base da datos para el gráfico y tabla.
00177
```

```

00178 Args:
00179 gesto_id (str): ID del gestor seleccionado
00180 curso_academico (list): Lista con los cursos académicos
00181 titulaciones (list): Lista con las titulaciones seleccionadas
00182
00183 Returns:
00184 pd.DataFrame: Datos para el gráfico y tabla
00185 """
00186 """
00187
00188 empty = pd.DataFrame()
00189
00190 if not gesto_id or not curso_academico or not titulaciones:
00191     return empty
00192
00193     data_universidad = universidades_gestor(gesto_id)
00194     if not data_universidad:
00195         return empty
00196
00197 try:
00198     titulaciones = list_to_tuple(titulaciones)
00199 except Exception as e:
00200     return empty
00201
00202 data = alumnos_nuevo_ingreso_nacionalidad_titulacion(
00203     data_universidad[0][0], curso_academico, titulaciones
00204 )
00205
00206 if not data:
00207     return empty
00208
00209 return pd.DataFrame(data, columns=["titulacion", "nacionalidad", "cantidad"])

```

Gráfico de llamadas para esta función:

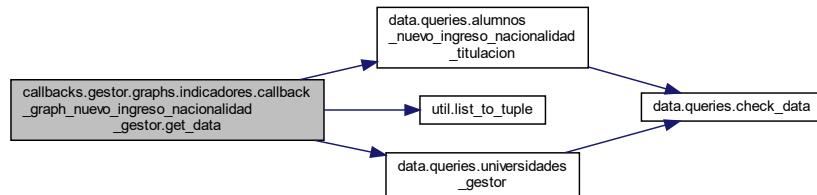
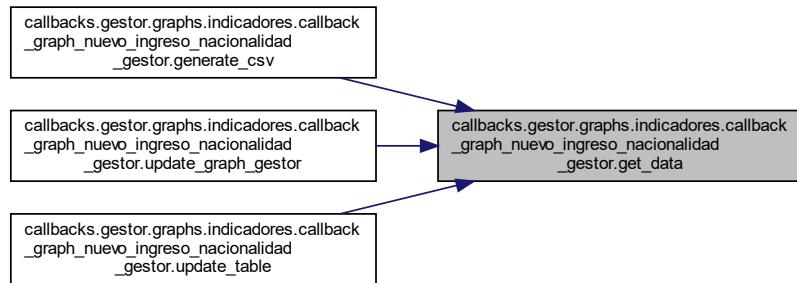


Gráfico de llamadas a esta función:



4.56.1.3. toggle_modal()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.←
    toggle_modal (
        btn,
        is_open )
```

Altera la visibilidad del modal con la tabla de datos de alumnos de nuevo ingreso por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

btn (int): Número de clicks en el botón "Ver datos"
is_open (bool): Estado actual del modal

Returns:

bool: Nuevo estado del modal

Definición en la línea 83 del archivo [callback_graph_nuevo_ingreso_nacionalidad_gestor.py](#).

```
00083 def toggle_modal(btn, is_open):
00084     """
00085 Alterna la visibilidad del modal con la tabla de datos de alumnos de nuevo ingreso
00086 por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00087
00088     Args:
00089         btn (int): Número de clicks en el botón "Ver datos"
00090         is_open (bool): Estado actual del modal
00091
00092     Returns:
00093         bool: Nuevo estado del modal
00094
00095     """
00096
00097 if btn:
00098     return not is_open
00099 return is_open
00100
00101
00102 @callback(
00103     Output("table-container-nuevo-ingreso-nacionalidad", "children"),
00104     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00105     State("selected-gestor-store", "data"),
00106     Input("curso-academico-gestor", "value"),
00107     Input("titulaciones-gestor", "value"),
00108 )
```

4.56.1.4. update_graph_gestor()

```
def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.←
    update_graph_gestor (
        gestor_id,
        curso_academico,
        titulaciones )
```

Actualiza el gráfico de alumnos de nuevo ingreso por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

gestor_id (str): ID del gestor seleccionado
curso_academico (list): Lista con los cursos académicos
titulaciones (list): Lista con las titulaciones seleccionadas

Returns:

go.Figure: Figura con el gráfico

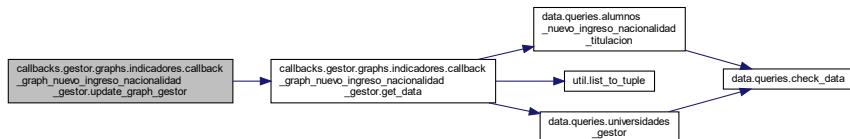
Definición en la línea 31 del archivo [callback_graph_nuevo_ingreso_nacionalidad_gestor.py](#).

```

00031 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00032 """
00033 Actualiza el gráfico de alumnos de nuevo ingreso por nacionalidad y titulación
00034 del perfil "Gestor" de la pestaña "Indicadores académicos".
00035
00036     Args:
00037         gestor_id (str): ID del gestor seleccionado
00038         curso_academico (list): Lista con los cursos académicos
00039         titulaciones (list): Lista con las titulaciones seleccionadas
00040
00041     Returns:
00042         go.Figure: Figura con el gráfico
00043
00044 """
00045
00046 fig = go.Figure()
00047
00048 fig.update_layout(
00049     barmode="stack",
00050     title={
00051         "text": "Alumnos de nuevo ingreso por nacionalidad y titulación",
00052         "x": 0.5,
00053     },
00054     xaxis_title="Titulaciones",
00055     yaxis_title="Nº Alumnos",
00056     showlegend=True,
00057     legend={"title": "Nacionalidad"},
00058 )
00059
00060 df = get_data(gestor_id, curso_academico, titulaciones)
00061
00062 if df.empty:
00063     return fig
00064
00065 # Pivotear el DataFrame para obtener las cantidades por nacionalidad en columnas separadas
00066 df_pivot = df.pivot_table(
00067     index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
00068 ).fillna(0)
00069
00070 for nacionalidad in df_pivot.columns:
00071     fig.add_trace(
00072         go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
00073     )
00074
00075 return fig
00076
00077
00078 @callback(
00079     Output("modal-nuevo-ingreso-nacionalidad", "is_open"),
00080     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00081     State("modal-nuevo-ingreso-nacionalidad", "is_open"),
00082 )

```

Gráfico de llamadas para esta función:



4.56.1.5. update_table()

```

def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.←
update_table (
    btn,
    gestor_id,
    curso_academico,
    titulaciones )

```

Actualiza la tabla de datos de alumnos de nuevo ingreso por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".

Args:

btn (int): Número de clicks en el botón "Ver datos"
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos
 titulaciones (list): Lista con las titulaciones seleccionadas

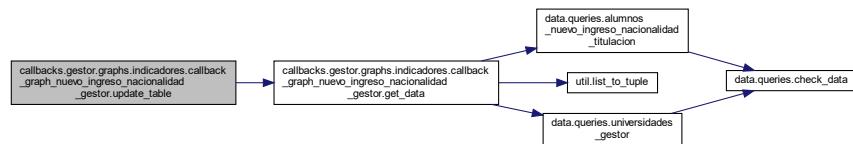
Returns:

dbc.Table: Tabla con los datos

Definición en la línea 109 del archivo [callback_graph_nuevo_ingreso_nacionalidad_gestor.py](#).

```
00109 def update_table(btn, gestor_id, curso_academico, titulaciones):
00110     """
00111     Actualiza la tabla de datos de alumnos de nuevo ingreso por nacionalidad y titulación
00112     del perfil "Gestor" de la pestaña "Indicadores académicos".
00113
00114     Args:
00115         btn (int): Número de clicks en el botón "Ver datos"
00116         gestor_id (str): ID del gestor seleccionado
00117         curso_academico (list): Lista con los cursos académicos
00118         titulaciones (list): Lista con las titulaciones seleccionadas
00119
00120     Returns:
00121         dbc.Table: Tabla con los datos
00122
00123     """
00124
00125 if not btn:
00126     return ""
00127
00128 df = get_data(gestor_id, curso_academico, titulaciones)
00129
00130 if df.empty:
00131     return dbc.Alert("No hay datos disponibles", color="info")
00132
00133 return dbc.Table.from_dataframe(
00134     df.head(50), striped=True, bordered=True, hover=True
00135 )
00136
00137
00138 @callback(
00139     Output("btn-descargar-nuevo-ingreso-nacionalidad", "href"),
00140     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00141     State("selected-gestor-store", "data"),
00142     Input("curso-academico-gestor", "value"),
00143     Input("titulaciones-gestor", "value"),
00144 )
```

Gráfico de llamadas para esta función:



4.57. Referencia del Namespace callbacks.gestor.graphs.resultados

Namespaces

- namespace [callback_graph_duracion_media_estudios_nota_gestor](#)
- namespace [callback_graph_nota_acceso_titulacion_gestor](#)

4.58. Referencia del Namespace callbacks.gestor.graphs.resultados.← callback_graph_duracion_media_estudios_nota_gestor

Funciones

- def [update_graph_gestor](#) (gestor_id)
- def [toggle_modal](#) (btn, is_open)
- def [update_table](#) (btn, gestor_id)
- def [generate_csv](#) (btn, gestor_id)
- def [get_data](#) (gestor_id)

4.58.1. Documentación de las funciones

4.58.1.1. generate_csv()

```
def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.←
generate_csv (
    btn,
    gestor_id )
```

Genera un archivo CSV descargable con los datos de duración media de los estudios con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".

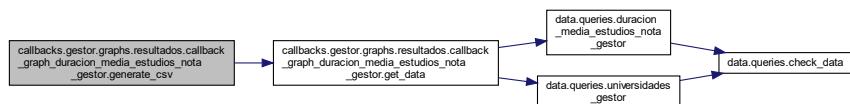
Args:
 btn (int): Número de clicks en el botón
 gestor_id (str): ID del gestor seleccionado

Returns:
 str: Enlace al archivo CSV

Definición en la línea 145 del archivo [callback_graph_duracion_media_estudios_nota_gestor.py](#).

```
00145 def generate_csv(btn, gestor_id):
00146     """
00147     Genera un archivo CSV descargable con los datos de duración media de los estudios
00148     con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".
00149
00150     Args:
00151         btn (int): Número de clicks en el botón
00152         gestor_id (str): ID del gestor seleccionado
00153
00154     Returns:
00155         str: Enlace al archivo CSV
00156     """
00157
00158     if not btn:
00159         return ""
00160
00161     df = get_data(gestor_id)
00162
00163     if df.empty:
00164         return ""
00165
00166     csv_string = df.to_csv(index=False, encoding="utf-8")
00167     csv_string = "data:text/csv;charset=utf-8," + csv_string
00168     return csv_string
00169
00170
```

Gráfico de llamadas para esta función:



4.58.1.2. get_data()

```
def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.←
get_data (←
    gestor_id )
```

Obtiene los datos de la base de datos.

Args:
gestor_id (str): ID del gestor seleccionado

Returns:
pd.DataFrame: Datos de la base de datos

Definición en la línea 171 del archivo [callback_graph_duracion_media_estudios_nota_gestor.py](#).

```
00171 def get_data(gestor_id):
00172     """
00173 Obtiene los datos de la base de datos.
00174
00175 Args:
00176     gestor_id (str): ID del gestor seleccionado
00177
00178 Returns:
00179     pd.DataFrame: Datos de la base de datos
00180     """
00181     empty = pd.DataFrame()
00182
00183     if not gestor_id:
00184         return empty
00185
00186     data_universidad = universidades_gestor(gestor_id)
00187     if not data_universidad:
00188         return empty
00189
00190     data = duracion_media_estudios_nota_gestor(data_universidad[0][0])
00191     if not data:
00192         return empty
00193
00194     df = pd.DataFrame(
00195         data,
00196         columns=[
00197             "nota_media",
00198             "titulacion",
00199             "rama",
00200             "curso_academico",
00201             "duracion_media",
00202             "numero_alumnos",
00203         ],
00204     )
00205     return df
```

Gráfico de llamadas para esta función:

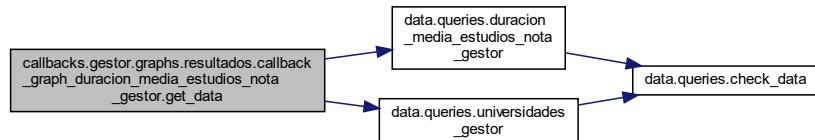
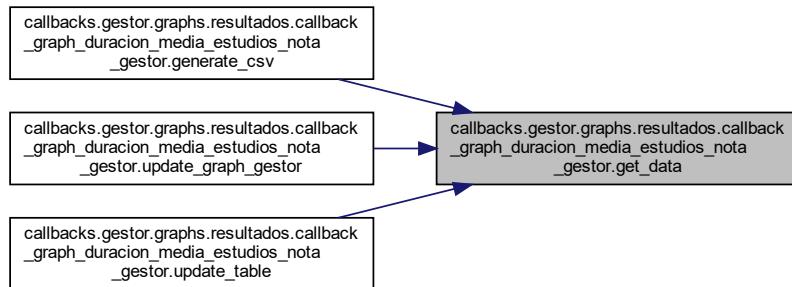


Gráfico de llamadas a esta función:



4.58.1.3. toggle_modal()

```

def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.toggle_modal (
    btn,
    is_open )
  
```

Altera la visibilidad del modal con los datos de duración media de los estudios con respecto a la nota media.

Args:
 btn (int): Número de clicks en el botón
 is_open (bool): Estado actual del modal

Returns:
 bool: Nuevo estado del modal

Definición en la línea 93 del archivo [callback_graph_duracion_media_estudios_nota_gestor.py](#).

```

00093 def toggle_modal(btn, is_open):
00094     """
00095 Alterna la visibilidad del modal con los datos de duración media de los estudios
00096 con respecto a la nota media.
00097
00098 Args:
00099 btn (int): Número de clicks en el botón
00100 is_open (bool): Estado actual del modal
00101
  
```

```

00102 Returns:
00103 bool: Nuevo estado del modal
00104 """
00105 if btn:
00106     return not is_open
00107     return is_open
00108
00109
00110 @callback(
00111     Output("table-container-duracion-estudios", "children"),
00112     Input("btn-ver-datos-duracion-estudios", "n_clicks"),
00113     State("selected-gestor-store", "data"),
00114 )

```

4.58.1.4. update_graph_gestor()

```

def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.←
update_graph_gestor (
    gestor_id )

```

Actualiza el gráfico de duración media de los estudios con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".

Args:

gestor_id (str): ID del gestor seleccionado

Returns:

go.Figure: Figura con el gráfico

Definición en la línea 25 del archivo [callback_graph_duracion_media_estudios_nota_gestor.py](#).

```

00025 def update_graph_gestor(gestor_id):
00026 """
00027 Actualiza el gráfico de duración media de los estudios con respecto a la nota media
00028 del perfil "Gestor" de la pestaña "Resultados académicos".
00029
00030     Args:
00031         gestor_id (str): ID del gestor seleccionado
00032
00033     Returns:
00034         go.Figure: Figura con el gráfico
00035     """
00036
00037 fig = go.Figure()
00038
00039 if not gestor_id:
00040     return fig
00041
00042 data = get_data(gestor_id)
00043
00044 if data.empty:
00045     return fig
00046
00047 fig = px.scatter(
00048     data,
00049     x="nota_media",
00050     y="duracion_media",
00051     color="rama",
00052     size="numero_alumnos",
00053     hover_name="titulacion",
00054     animation_frame="curso_academico",
00055     animation_group="rama",
00056     category_orders={
00057         "rama": sorted(data["rama"].unique())
00058     },
00059 )
00060
00061 fig.update_layout(
00062     title={
00063         "text": "Duración media de los estudios con respecto a la nota media",
00064         "x": 0.5,
00065     },
00066     xaxis_title="Nota media",
00067     yaxis_title="Duración media de los estudios",

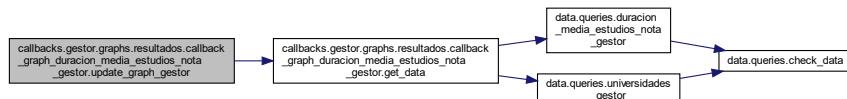
```

```

00068         legend={"title": "Rama de conocimiento"},
00069     )
00070
00071     fig.update_traces(
00072         marker=dict(line=dict(width=1, color="DarkSlateGrey")),
00073         textposition="top center"
00074     )
00075
00076     # Ajustar el rango de los ejes para que las burbujas no se corten
00077     fig.update_xaxes(range=[data["nota_media"].min() - 1, data["nota_media"].max() + 1])
00078     fig.update_yaxes(
00079         range=[data["duracion_media"].min() - 1, data["duracion_media"].max() + 1]
00080     )
00081
00082     if len(data["curso_academico"].unique()) > 1:
00083         fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 1000
00084
00085     return fig
00086
00087
00088 @callback(
00089     Output("modal-duracion-estudios", "is_open"),
00090     Input("btn-ver-datos-duracion-estudios", "n_clicks"),
00091     State("modal-duracion-estudios", "is_open"),
00092 )

```

Gráfico de llamadas para esta función:



4.58.1.5. update_table()

```

def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.←
update_table (
    btn,
    gestor_id )

```

Actualiza la tabla con los datos de duración media de los estudios con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".

Args:
 btn (int): Número de clicks en el botón
 gestor_id (str): ID del gestor seleccionado

Returns:
 dbc.Table: Tabla con los datos

Definición en la línea 115 del archivo [callback_graph_duracion_media_estudios_nota_gestor.py](#).

```

00115 def update_table(btn, gestor_id):
00116     """
00117 Actualiza la tabla con los datos de duración media de los estudios
00118 con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".
00119
00120     Args:
00121         btn (int): Número de clicks en el botón
00122         gestor_id (str): ID del gestor seleccionado
00123
00124     Returns:
00125         dbc.Table: Tabla con los datos
00126     """

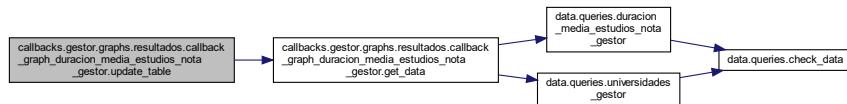
```

```

00127 if not btn:
00128     return ""
00129
00130 df = get_data(gestor_id)
00131
00132 if df.empty:
00133     return dbc.Alert("No hay datos disponibles", color="info")
00134
00135 return dbc.Table.from_dataframe(
00136     df.head(50), striped=True, bordered=True, hover=True
00137 )
00138
00139
00140 @callback(
00141     Output("btn-descargar-csv-duracion-estudios", "href"),
00142     Input("btn-descargar-csv-duracion-estudios", "n_clicks"),
00143     State("selected-gestor-store", "data"),
00144 )

```

Gráfico de llamadas para esta función:



4.59. Referencia del Namespace callbacks.gestor.graphs.resultados.← callback_graph_nota_acceso_titulacion_gestor

Funciones

- def `update_grph_gestor` (gestor_id)
- def `toggle_modal` (btn, is_open)
- def `update_table` (btn, gestor_id)
- def `generate_csv` (btn, gestor_id)
- def `get_data` (gestor_id)

4.59.1. Documentación de las funciones

4.59.1.1. generate_csv()

```

def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.generate←
_csv (
    btn,
    gestor_id )

```

Genera un archivo CSV con los datos de la nota de acceso por titulación del perfil "Gestor" de la pestaña "Resultados académicos".

Args:

btn (int): Número de clicks en el botón
gestor_id (str): ID del gestor seleccionado

Returns:

str: Enlace al archivo CSV

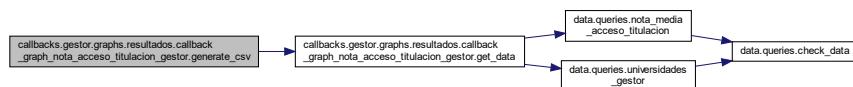
Definición en la línea 124 del archivo [callback_graph_nota_acceso_titulacion_gestor.py](#).

```

00124 def generate_csv(btn, gestor_id):
00125     """
00126 Genera un archivo CSV con los datos de la nota de acceso por titulación
00127 del perfil "Gestor" de la pestaña "Resultados académicos".
00128
00129     Args:
00130         btn (int): Número de clicks en el botón
00131         gestor_id (str): ID del gestor seleccionado
00132
00133     Returns:
00134         str: Enlace al archivo CSV
00135
00136     """
00137
00138 if not btn:
00139     return ""
00140
00141 df = get_data(gestor_id)
00142
00143 if df.empty:
00144     return ""
00145
00146 csv_string = df.to_csv(index=False, encoding="utf-8")
00147 csv_string = "data:text/csv; charset=utf-8," + csv_string
00148 return csv_string
00149
00150

```

Gráfico de llamadas para esta función:



4.59.1.2. [get_data\(\)](#)

```

def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.get_data (
    gestor_id )

```

Obtiene los datos de la base de datos.

Args:
gestor_id (str): ID del gestor seleccionado

Returns:
pd.DataFrame: Datos de la nota de acceso por titulación

Definición en la línea 151 del archivo [callback_graph_nota_acceso_titulacion_gestor.py](#).

```

00151 def get_data(gestor_id):
00152     """
00153 Obtiene los datos de la base de datos.
00154
00155     Args:
00156         gestor_id (str): ID del gestor seleccionado
00157
00158     Returns:
00159         pd.DataFrame: Datos de la nota de acceso por titulación
00160     """
00161 empty = pd.DataFrame()
00162
00163 if not gestor_id:
00164     return empty
00165
00166     data_universidad = universidades_gestor(gestor_id)

```

```

00167
00168     if not data_universidad:
00169         return empty
00170
00171     data = nota_media_acceso_titulacion(data_universidad[0][0])
00172
00173     if not data:
00174         return empty
00175
00176     return pd.DataFrame(data, columns=["curso_academico", "titulacion", "nota"])

```

Gráfico de llamadas para esta función:

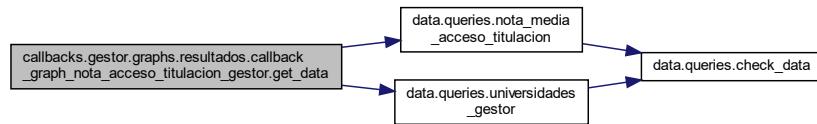
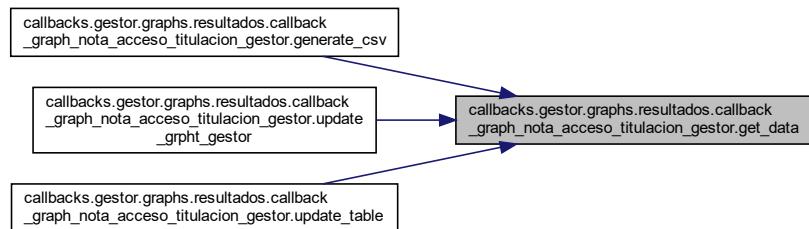


Gráfico de llamadas a esta función:



4.59.1.3. toggle_modal()

```

def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.toggle_modal (
    btn,
    is_open )

```

Alternar la visibilidad del modal de datos de la nota de acceso por titulación del perfil "Gestor" de la pestaña "Resultados académicos".

Args:
 btn (int): Número de clicks en el botón
 is_open (bool): Estado actual del modal

Returns:
 bool: Nuevo estado del modal

Definición en la línea 69 del archivo [callback_graph_nota_acceso_titulacion_gestor.py](#).

```
00069 def toggle_modal(btn, is_open):
00070     """
00071 Alternar la visibilidad del modal de datos de la nota de acceso por titulación
00072 del perfil "Gestor" de la pestaña "Resultados académicos".
00073
00074     Args:
00075         btn (int): Número de clicks en el botón
00076         is_open (bool): Estado actual del modal
00077
00078     Returns:
00079         bool: Nuevo estado del modal
00080
00081     """
00082 if btn:
00083     return not is_open
00084 return is_open
00085
00086
00087 @callback(
00088     Output("table-container-nota-acceso", "children"),
00089     Input("btn-ver-datos-nota-acceso", "n_clicks"),
00090     State("selected-gestor-store", "data"),
00091 )
```

4.59.1.4. update_grph_gestor()

```
def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.update_←
grph_gestor (
    gestor_id )
```

Actualiza el gráfico de evolución de la nota de acceso por titulación del perfil "Gestor" de la pestaña "Resultados académicos".

Args:
gestor_id (str): ID del gestor seleccionado

Returns:
go.Figure: Figura con el gráfico

Definición en la línea 24 del archivo [callback_graph_nota_acceso_titulacion_gestor.py](#).

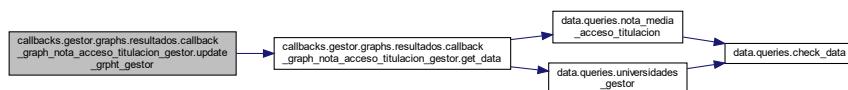
```
00024 def update_grph_gestor(gestor_id):
00025     """
00026 Actualiza el gráfico de evolución de la nota de acceso por titulación
00027 del perfil "Gestor" de la pestaña "Resultados académicos".
00028
00029     Args:
00030         gestor_id (str): ID del gestor seleccionado
00031
00032     Returns:
00033         go.Figure: Figura con el gráfico
00034
00035     """
00036
00037 fig = go.Figure()
00038
00039 fig.update_layout(
00040     title={"text": "Evolución nota de acceso por titulación", "x": 0.5},
00041     xaxis_title="Curso académico",
00042     yaxis_title="Nota media de acceso",
00043     legend={"title": "Titulaciones"},
00044 )
00045
00046 df = get_data(gestor_id)
00047
00048 if df.empty:
00049     return fig
00050
00051 for titulation, group in df.groupby("titulacion"):
00052     fig.add_trace(
00053         go.Scatter(
00054             x=group["curso_academico"],
00055             y=group["nota"],
```

```

00056             mode="lines+markers",
00057             name=titulation,
00058         )
00059     )
00060
00061     return fig
00062
00063
00064 @callback(
00065     Output("modal-nota-acceso", "is_open"),
00066     Input("btn-ver-datos-nota-acceso", "n_clicks"),
00067     State("modal-nota-acceso", "is_open"),
00068 )

```

Gráfico de llamadas para esta función:



4.59.1.5. update_table()

```

def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.update_←
table (
    btn,
    gestor_id )

```

Actualiza la tabla con los datos de la nota de acceso por titulación del perfil "Gestor" de la pestaña "Resultados académicos".

Args:
 btn (int): Número de clicks en el botón
 gestor_id (str): ID del gestor seleccionado

Returns:
 dbc.Table: Tabla con los datos

Definición en la línea 92 del archivo [callback_graph_nota_acceso_titulacion_gestor.py](#).

```

00092 def update_table(btn, gestor_id):
00093     """
00094 Actualiza la tabla con los datos de la nota de acceso por titulación
00095 del perfil "Gestor" de la pestaña "Resultados académicos".
00096
00097     Args:
00098         btn (int): Número de clicks en el botón
00099         gestor_id (str): ID del gestor seleccionado
00100
00101     Returns:
00102         dbc.Table: Tabla con los datos
00103
00104     """
00105
00106 if not btn:
00107     return ""
00108
00109 df = get_data(gestor_id)
00110
00111 if df.empty:
00112     return dbc.Alert("No hay datos disponibles", color="info")
00113
00114 return dbc.Table.from_dataframe(
00115     df.head(50), striped=True, bordered=True, hover=True
00116 )

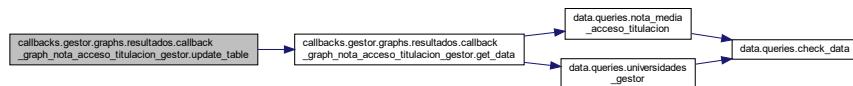
```

```

00117
00118
00119 @callback(
00120     Output("btn-descargar-csv-nota-acceso", "href"),
00121     Input("btn-ver-datos-nota-acceso", "n_clicks"),
00122     State("selected-gestor-store", "data"),
00123 )

```

Gráfico de llamadas para esta función:



4.60. Referencia del Namespace `callbacks.gestor.graphs riesgo_abandono`

Namespaces

- namespace `callback_graph_tasa_abandono_gestor`
- namespace `callback_graph_tasa_graduacion_gestor`

4.61. Referencia del Namespace `callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor`

Funciones

- def `update_graph_gestor` (curso_academico, gestor_id)
- def `toggle_modal` (btn, is_open)
- def `update_table` (btn, gestor_id, curso_academico)
- def `generate_csv` (btn, gestor_id, curso_academico)
- def `get_data` (gestor_id, curso_academico)

4.61.1. Documentación de las funciones

4.61.1.1. generate_csv()

```
def callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor.generate_csv (
    btn,
    gestor_id,
    curso_academico )
```

Genera un archivo CSV descargable con los datos de la tasa de abandono por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:

btn (int): Número de clicks en el botón
gestor_id (str): ID del gestor seleccionado
curso_academico (list): Lista con los cursos académicos

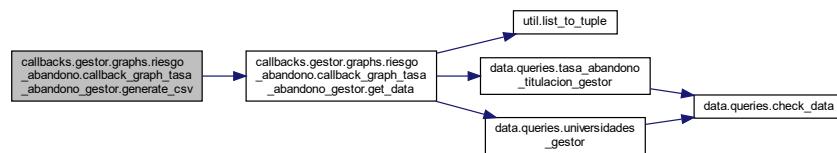
Returns:

str: Enlace al archivo CSV

Definición en la línea 132 del archivo [callback_graph_tasa_abandono_gestor.py](#).

```
00132 def generate_csv(btn, gestor_id, curso_academico):
00133     """
00134     Genera un archivo CSV descargable con los datos de la tasa de abandono por titulación
00135     del perfil "Gestor" de la pestaña "Riesgo de abandono".
00136
00137     Args:
00138         btn (int): Número de clicks en el botón
00139         gestor_id (str): ID del gestor seleccionado
00140         curso_academico (list): Lista con los cursos académicos
00141
00142     Returns:
00143         str: Enlace al archivo CSV
00144     """
00145
00146 if not btn:
00147     return ""
00148 df = get_data(gestor_id, curso_academico)
00149
00150 if df.empty:
00151     return ""
00152
00153 df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00154 csv_string = df.to_csv(index=False, encoding="utf-8")
00155 csv_string = "data:text/csv;charset=utf-8," + csv_string
00156 return csv_string
00157
00158
```

Gráfico de llamadas para esta función:

**4.61.1.2. get_data()**

```
def callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor.get_data (
    gestor_id,
    curso_academico )
```

Obtiene los datos de la base de datos.

Args:
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos

Returns:
 pd.DataFrame: Datos de la tasa de abandono por titulación

Definición en la línea 159 del archivo [callback_graph_tasa_abandono_gestor.py](#).

```
00159 def get_data(gestor_id, curso_academico):
00160     """
00161     Obtiene los datos de la base de datos.
00162
00163     Args:
00164         gestor_id (str): ID del gestor seleccionado
00165         curso_academico (list): Lista con los cursos académicos
00166
00167     Returns:
00168         pd.DataFrame: Datos de la tasa de abandono por titulación
00169     """
00170     empty = pd.DataFrame()
00171
00172     if not gestor_id or not curso_academico:
00173         return empty
00174
00175     data_universidad = universidades_gestor(gestor_id)
00176     if not data_universidad:
00177         return empty
00178     try:
00179         curso_academico = list_to_tuple(curso_academico)
00180     except Exception as e:
00181         print("Error:", e)
00182         return empty
00183
00184     data = tasa_abandono_titulacion_gestor(data_universidad[0][0], curso_academico)
00185
00186     if not data:
00187         return empty
00188
00189     return pd.DataFrame(
00190         data,
00191         columns=[
00192             "curso_academico",
00193             "titulacion",
00194             "numero_matriculados",
00195             "numero_abandonos",
00196         ],
00197     )
```

Gráfico de llamadas para esta función:

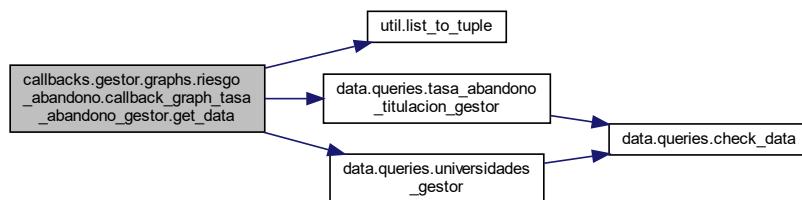
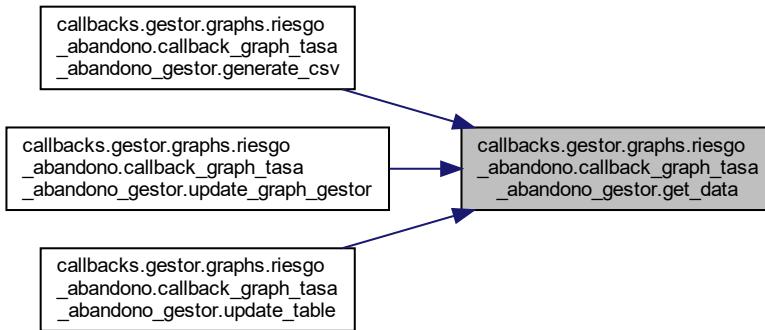


Gráfico de llamadas a esta función:



4.61.1.3. toggle_modal()

```
def callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor.toggle_modal (
    btn,
    is_open )
```

Alternar la visibilidad del modal de datos de la tasa de abandono por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:

btn (int): Número de clicks en el botón
is_open (bool): Estado actual del modal

Returns:

bool: Nuevo estado del modal

Definición en la línea 74 del archivo [callback_graph_tasa_abandono_gestor.py](#).

```
00074 def toggle_modal(btn, is_open):
00075     """
00076     Alternar la visibilidad del modal de datos de la tasa
00077     de abandono por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".
00078
00079     Args:
00080         btn (int): Número de clicks en el botón
00081         is_open (bool): Estado actual del modal
00082
00083     Returns:
00084         bool: Nuevo estado del modal
00085     """
00086
00087     if btn:
00088         return not is_open
00089     return is_open
00090
00091
00092     @callback(
00093         Output("table-container-tasa-abandono", "children"),
00094         Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00095         State("selected-gestor-store", "data"),
00096         Input("curso-all-academico-gestor", "value"),
00097     )
```

4.61.1.4. update_graph_gestor()

```
def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor.update_graph←
_gestor (
    curso_academico,
    gestor_id )
```

Actualiza el gráfico de la tasa de abandono por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:

curso_academico (list): Lista con los cursos académicos
gestor_id (str): ID del gestor seleccionado

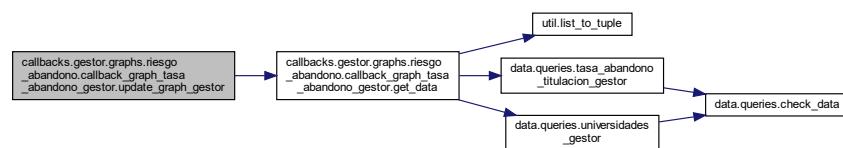
Returns:

go.Figure: Figura con el gráfico

Definición en la línea 26 del archivo [callback_graph_tasa_abandono_gestor.py](#).

```
00026 def update_graph_gestor(curso_academico, gestor_id):
00027     """
00028 Actualiza el gráfico de la tasa de abandono por titulación
00029 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00030
00031     Args:
00032         curso_academico (list): Lista con los cursos académicos
00033         gestor_id (str): ID del gestor seleccionado
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """
00038 fig = go.Figure()
00039
00040 fig.update_layout(
00041     title={"text": "Tasa de abandono por titulación ", "x": 0.5},
00042         xaxis_title="Curso académico",
00043         yaxis_title="Tasa de abandono (%)",
00044         showlegend=True,
00045         legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
00046     )
00047
00048 df = get_data(gestor_id, curso_academico)
00049
00050 if df.empty:
00051     return fig
00052
00053 df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00054
00055 for titulacion in df["titulacion"].unique():
00056     df_titulacion = df[df["titulacion"] == titulacion]
00057     fig.add_trace(
00058         go.Scatter(
00059             x=df_titulacion["curso_academico"],
00060             y=df_titulacion["tasa_abandono"],
00061             mode="lines+markers",
00062             name=titulacion,
00063         )
00064     )
00065
00066 return fig
00067
00068
00069 @callback(
00070     Output("modal-tasa-abandono", "is_open"),
00071     Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00072     State("modal-tasa-abandono", "is_open"),
00073 )
```

Gráfico de llamadas para esta función:



4.61.1.5. update_table()

```
def callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor.update_table (
    btn,
    gestor_id,
    curso_academico )
```

Actualiza la tabla con los datos de la tasa de abandono por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:

btn (int): Número de clicks en el botón
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos

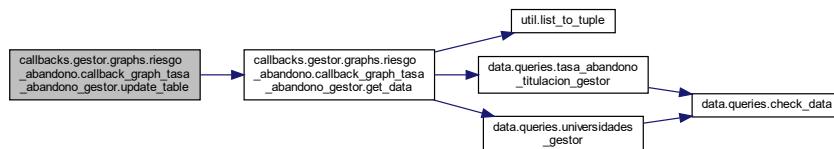
Returns:

dbc.Table: Tabla con los datos

Definición en la línea 98 del archivo [callback_graph_tasa_abandono_gestor.py](#).

```
00098 def update_table(btn, gestor_id, curso_academico):
00099     """
00100     Actualiza la tabla con los datos de la tasa de abandono por titulación
00101     del perfil "Gestor" de la pestaña "Riesgo de abandono".
00102
00103     Args:
00104         btn (int): Número de clicks en el botón
00105         gestor_id (str): ID del gestor seleccionado
00106         curso_academico (list): Lista con los cursos académicos
00107
00108     Returns:
00109         dbc.Table: Tabla con los datos
00110     """
00111
00112     if not btn:
00113         return ""
00114
00115     df = get_data(gestor_id, curso_academico)
00116
00117     if df.empty:
00118         return dbc.Alert("No hay datos disponibles", color="info")
00119
00120     df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00121     return dbc.Table.from_dataframe(
00122         df.head(50), striped=True, bordered=True, hover=True, responsive=True
00123     )
00124
00125
00126 @callback(
00127     Output("btn-descargar-tasa-abandono", "href"),
00128     Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00129     State("selected-gestor-store", "data"),
00130     Input("curso-all-academico-gestor", "value"),
00131 )
```

Gráfico de llamadas para esta función:



4.62. Referencia del Namespace callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor

Funciones

- def [update_graph_gestor](#) (curso_academico, gestor_id)
- def [toggle_modal](#) (btn, is_open)
- def [update_table](#) (btn, curso_academico, gestor_id)
- def [generate_csv](#) (n, curso_academico, gestor_id)
- def [get_data](#) (gestor_id, curso_academico)

4.62.1. Documentación de las funciones

4.62.1.1. generate_csv()

```
def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.generate_csv
(
    n,
    curso_academico,
    gestor_id )
```

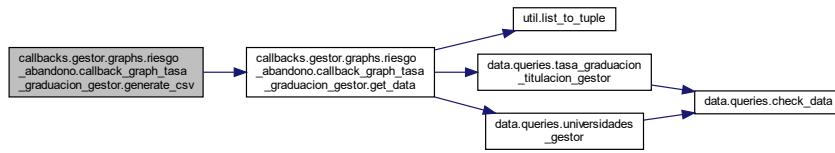
Genera un archivo CSV descargable con los datos de la tasa de graduación por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:
 n (int): Número de clicks en el botón
 curso_academico (list): Lista con los cursos académicos
 gestor_id (str): ID del gestor seleccionado

Definición en la línea 125 del archivo [callback_graph_tasa_graduacion_gestor.py](#).

```
00125 def generate_csv(n, curso_academico, gestor_id):
00126     """
00127     Genera un archivo CSV descargable con los datos de la tasa de graduación por titulación
00128     del perfil "Gestor" de la pestaña "Riesgo de abandono".
00129
00130     Args:
00131         n (int): Número de clicks en el botón
00132         curso_academico (list): Lista con los cursos académicos
00133         gestor_id (str): ID del gestor seleccionado
00134     """
00135
00136 if not n:
00137     return None
00138
00139 df = get_data(gestor_id, curso_academico)
00140
00141 if df is None:
00142     return None
00143
00144 csv_string = df.to_csv(index=False, encoding="utf-8")
00145 csv_string = "data:text/csv; charset=utf-8," + csv_string
00146
00147 return csv_string
00148
00149
```

Gráfico de llamadas para esta función:



4.62.1.2. get_data()

```
def callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor.get_data (
    gestor_id,
    curso_academico )
```

Obtiene los datos de la tasa de graduación por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:
 gestor_id (str): ID del gestor seleccionado
 curso_academico (list): Lista con los cursos académicos

Returns:
 pd.DataFrame: Datos de la tasa de graduación

Definición en la línea 150 del archivo [callback_graph_tasa_graduacion_gestor.py](#).

```
00150 def get_data(gestor_id, curso_academico):
00151     """
00152     Obtiene los datos de la tasa de graduación por titulación
00153     del perfil "Gestor" de la pestaña "Riesgo de abandono".
00154
00155     Args:
00156         gestor_id (str): ID del gestor seleccionado
00157         curso_academico (list): Lista con los cursos académicos
00158
00159     Returns:
00160         pd.DataFrame: Datos de la tasa de graduación
00161     """
00162
00163 empty = pd.DataFrame()
00164
00165 if not gestor_id or not curso_academico:
00166     return empty
00167
00168 try:
00169     curso_academico = list_to_tuple(curso_academico)
00170 except Exception as e:
00171     print("Error:", e)
00172     return empty
00173
00174 data_universidad = universidades_gestor(gestor_id)
00175
00176 if not data_universidad:
00177     return empty
00178
00179 data = tasa_graduacion_titulacion_gestor(data_universidad[0][0], curso_academico)
00180
00181 if not data:
00182     return empty
00183
00184 df = pd.DataFrame(
00185     data,
00186     columns=[
00187         "numero_matriculados",
```

```

00188         "numero_egresados",
00189         "curso_academico",
00190         "titulacion",
00191     ],
00192 )
00193 df["tasa_graduacion"] = (df["numero_egresados"] / df["numero_matriculados"]) * 100
00194
00195 return df

```

Gráfico de llamadas para esta función:

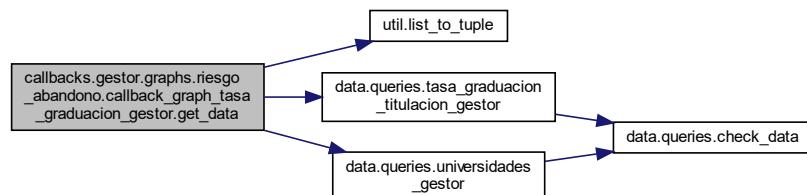
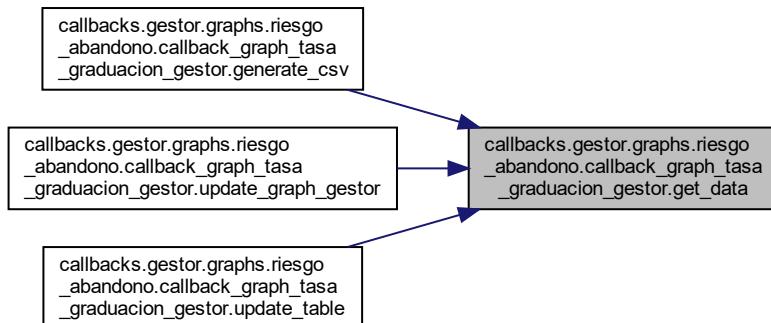


Gráfico de llamadas a esta función:



4.62.1.3. toggle_modal()

```

def callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor.toggle_modal(
(
    btn,
    is_open )

```

Alternar la visibilidad del modal de datos de la tasa de graduación por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:

- btn (int): Número de clicks en el botón
- is_open (bool): Estado actual del modal

Definición en la línea 72 del archivo [callback_graph_tasa_graduacion_gestor.py](#).

```
00072 def toggle_modal(btn, is_open):
00073     """
00074 Alternar la visibilidad del modal de datos de la tasa de graduación por titulación
00075 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00076
00077     Args:
00078         btn (int): Número de clicks en el botón
00079         is_open (bool): Estado actual del modal
00080     """
00081
00082     if btn:
00083         return not is_open
00084     return is_open
00085
00086
00087     @callback(
00088         Output("table-container-tasa-graduacion", "children"),
00089         Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
00090         State("curso-all-academico-gestor", "value"),
00091         State("selected-gestor-store", "data"),
00092     )
```

4.62.1.4. update_graph_gestor()

```
def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.update_←
graph_gestor (
    curso_academico,
    gestor_id )
```

Actualiza el gráfico de la tasa de graduación por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:

 curso_academico (list): Lista con los cursos académicos
 gestor_id (str): ID del gestor seleccionado

Returns:

 go.Figure: Figura con el gráfico

Definición en la línea 26 del archivo [callback_graph_tasa_graduacion_gestor.py](#).

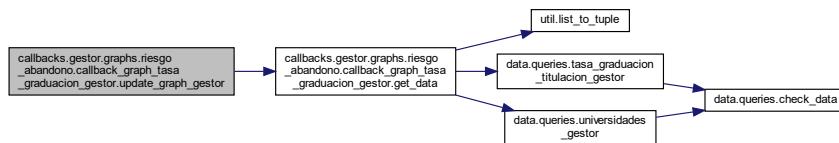
```
00026 def update_graph_gestor(curso_academico, gestor_id):
00027     """
00028 Actualiza el gráfico de la tasa de graduación por titulación
00029 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00030
00031     Args:
00032         curso_academico (list): Lista con los cursos académicos
00033         gestor_id (str): ID del gestor seleccionado
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     title={"text": "Tasa de graduación por titulación ", "x": 0.5},
00043     xaxis_title="Curso académico",
00044     yaxis_title="Tasa de graduación (%)",
00045     showlegend=True,
00046     legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
00047 )
00048
00049 df = get_data(gestor_id, curso_academico)
00050
00051 if df.empty:
00052     return fig
00053
00054 for titulacion, group in df.groupby("titulacion"):
00055     fig.add_trace(
00056         go.Scatter(
00057             x=group["curso_academico"],
```

```

00058         y=group["tasa_graduacion"],
00059         mode="lines+markers",
00060         name=titulacion,
00061     )
00062   )
00063
00064   return fig
00065
00066
00067 @callback(
00068   Output("modal-tasa-graduacion", "is_open"),
00069   Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
00070   State("modal-tasa-graduacion", "is_open"),
00071 )

```

Gráfico de llamadas para esta función:



4.62.1.5. update_table()

```

def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.update_table
(
    btn,
    curso_academico,
    gestor_id )

```

Actualiza la tabla de datos de la tasa de graduación por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".

Args:

btn (int): Número de clicks en el botón
 curso_academico (list): Lista con los cursos académicos
 gestor_id (str): ID del gestor seleccionado

Returns:

dbc.Table: Tabla con los datos

Definición en la línea 93 del archivo [callback_graph_tasa_graduacion_gestor.py](#).

```

00093 def update_table(btn, curso_academico, gestor_id):
00094     """
00095     Actualiza la tabla de datos de la tasa de graduación por titulación
00096     del perfil "Gestor" de la pestaña "Riesgo de abandono".
00097
00098     Args:
00099     btn (int): Número de clicks en el botón
00100     curso_academico (list): Lista con los cursos académicos
00101     gestor_id (str): ID del gestor seleccionado
00102
00103     Returns:
00104     dbc.Table: Tabla con los datos
00105     """
00106 if not btn:
00107     return ""
00108
00109 df = get_data(gestor_id, curso_academico)
00110
00111 if df.empty:

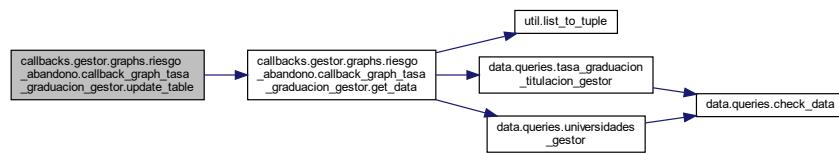
```

```

00112     return dbc.Alert("No hay datos disponibles", color="info")
00113
00114     return dbc.Table.from_dataframe(
00115         df.head(50), striped=True, bordered=True, hover=True, responsive=True
00116     )
00117
00118
00119 @callback(
00120     Output("btn-descargar-tasa-graduacion", "href"),
00121     Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
00122     State("curso-all-academico-gestor", "value"),
00123     State("selected-gestor-store", "data"),
00124 )

```

Gráfico de llamadas para esta función:



4.63. Referencia del Namespace callbacks.gestor.utils

Namespaces

- namespace `callback_resumen_gestor`
- namespace `callback_select_gestor`
- namespace `callback_tabs_gestor`

4.64. Referencia del Namespace callbacks.gestor.utils.callback_resumen_gestor

Funciones

- def `update_resumen_gestor (gestor_id)`
- def `not_data ()`

4.64.1. Documentación de las funciones

4.64.1.1. not_data()

```
def callbacks.gestor.utils.callback_resumen_gestor.not_data ( )
```

Definición en la línea 57 del archivo [callback_resumen_gestor.py](#).

```
00057 def not_data():
00058     return html.Div(
00059         [
00060             html.H2("Resumen"),
00061             html.P("Universidad:", className="resumen-label"),
00062             html.P("No disponible"),
00063             html.P("Gestor:", className="resumen-label"),
00064             html.P("No disponible"),
00065             html.P("Número de alumnos matriculados:", className="resumen-label"),
00066             html.P("No disponible"),
00067             html.Hr(),
00068         ]
00069     )
```

Gráfico de llamadas a esta función:



4.64.1.2. update_resumen_gestor()

```
def callbacks.gestor.utils.callback_resumen_gestor.update_resumen_gestor (
    gestor_id )
```

Actualiza el resumen del gestor seleccionado.

Args:

gestor_id (str): ID del gestor seleccionado

Returns:

html.Div: Resumen del gestor seleccionado

Definición en la línea 19 del archivo [callback_resumen_gestor.py](#).

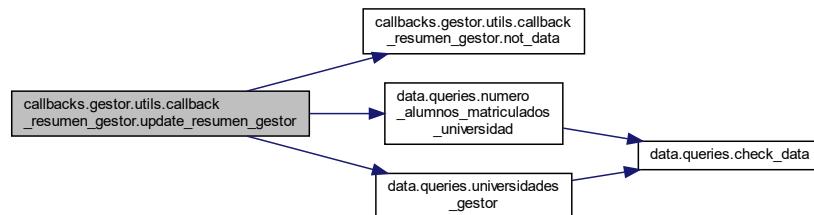
```
00019 def update_resumen_gestor(gestor_id):
00020     """
00021 Actualiza el resumen del gestor seleccionado.
00022
00023 Args:
00024     gestor_id (str): ID del gestor seleccionado
00025
00026 Returns:
00027     html.Div: Resumen del gestor seleccionado
00028 """
00029
00030     if not gestor_id:
00031         return not_data()
00032
00033     data = universidades_gestor(gestor_id)
00034
00035     if not data:
00036         return not_data()
00037
00038     data_alumnos = numero_alumnos_matriculados_universidad(data[0][1])
00039
```

```

00040     if not data_alumnos:
00041         return not_data()
00042
00043     return html.Div(
00044         [
00045             html.H2("Resumen"),
00046             html.P("Universidad:", className="resumen-label"),
00047             html.P(data[0][1]),
00048             html.P("Gestor:", className="resumen-label"),
00049             html.P(gestor_id),
00050             html.P("Número de alumnos matriculados:", className="resumen-label"),
00051             html.P(data_alumnos[0][0]),
00052             html.Hr(),
00053         ]
00054     )
00055
00056

```

Gráfico de llamadas para esta función:



4.65. Referencia del Namespace callbacks.gestor.utils.callback_select_gestor

Funciones

- def [store_selected_gestor](#) (selected_value, stored_value)

4.65.1. Documentación de las funciones

4.65.1.1. store_selected_gestor()

```
def callbacks.gestor.utils.callback_select_gestor.store_selected_gestor (
    selected_value,
    stored_value )
```

Almacena el gestor seleccionado en un store.

Args:

 selected_value (str): Valor seleccionado en el dropdown
 stored_value (dict): Datos almacenados en el store

Returns:

 str: Valor seleccionado en el dropdown
 list: Opciones del dropdown
 str: Valor almacenado en el store

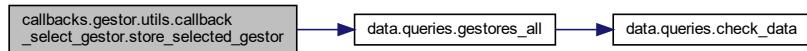
Definición en la línea 22 del archivo [callback_select_gestor.py](#).

```

00022 def store_selected_gestor(selected_value, stored_value):
00023 """
00024 Almacena el gestor seleccionado en un store.
00025
00026 Args:
00027 selected_value (str): Valor seleccionado en el dropdown
00028 stored_value (dict): Datos almacenados en el store
00029
00030 Returns:
00031 str: Valor seleccionado en el dropdown
00032 list: Opciones del dropdown
00033 str: Valor almacenado en el store
00034 """
00035
00036 data = gestores_all()
00037
00038 if not data:
00039     return None, [], None
00040
00041 opciones_dropdown = [{"label": gestor[0], "value": gestor[0]} for gestor in data]
00042
00043 if selected_value is None and stored_value in [
00044     op["value"] for op in opciones_dropdown
00045 ]:
00046     return stored_value, opciones_dropdown, stored_value
00047
00048 return selected_value, opciones_dropdown, selected_value

```

Gráfico de llamadas para esta función:



4.66. Referencia del Namespace callbacks.gestor.utils.callback_tabs_gestor

Funciones

- def [render_content](#) (tab, selected_gestor)

4.66.1. Documentación de las funciones

4.66.1.1. render_content()

```

def callbacks.gestor.utils.callback_tabs_gestor.render_content (
    tab,
    selected_gestor )

```

Renderiza el contenido de las pestañas de la sección "Gestor".

Args:

- tab (str): Pestaña seleccionada
- selected_gestor (dict): Datos del gestor seleccionado

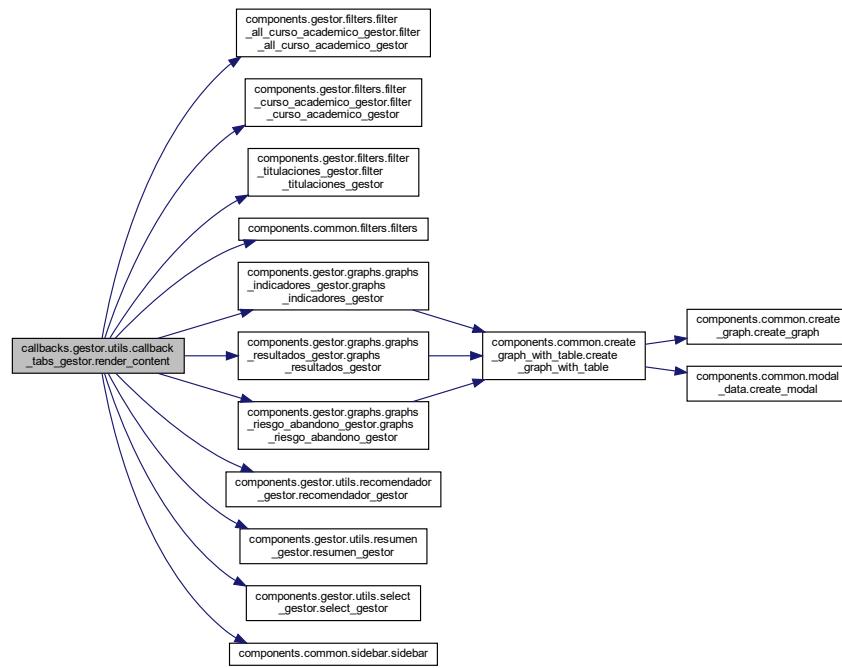
Returns:

- html.Div: Contenido de la pestaña seleccionada

Definición en la línea 30 del archivo [callback_tabs_gestor.py](#).

```
00030 def render_content(tab, selected_gestor):
00031     """
00032     Renderiza el contenido de las pestañas de la sección "Gestor".
00033
00034     Args:
00035         tab (str): Pestaña seleccionada
00036         selected_gestor (dict): Datos del gestor seleccionado
00037
00038     Returns:
00039         html.Div: Contenido de la pestaña seleccionada
00040     """
00041
00042 if tab == 'indicadores-academicos-tab':
00043     return html.Div([
00044         select_gestor(),
00045         html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
00046         html.Div([
00047             sidebar([
00048                 resumen_gestor(),
00049                 filters([
00050                     filter_curso_academico_gestor(),
00051                     filter_titulaciones_gestor()
00052                 ]),
00053                 graphs_indicadores_gestor()
00054             ], className='content-layout-dashboard')
00055         ]),
00056     ])
00057 elif tab == 'resultados-academicos-tab':
00058     return html.Div([
00059         html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
00060         html.Div([
00061             sidebar([
00062                 resumen_gestor(),
00063                 graphs_resultados_gestor()
00064             ], className='content-layout-dashboard')
00065         ]),
00066     ])
00067 elif tab == 'riesgo-abandono-tab':
00068     return html.Div([
00069         html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
00070         html.Div([
00071             sidebar([
00072                 resumen_gestor(),
00073                 filters([
00074                     filter_all_curso_academico_gestor()
00075                 ]),
00076                 graphs_riesgo_abandono_gestor()
00077             ], className='content-layout-dashboard')
00078         ]),
00079     ])
00080 elif tab == 'recomendaciones-tab':
00081     return html.Div([
00082         recomendador_gestor()
00083     ])
00084
00085
00086
00087
```

Gráfico de llamadas para esta función:



4.67. Referencia del Namespace components

Namespaces

- namespace [alumnado](#)
- namespace [common](#)
- namespace [docente](#)
- namespace [gestor](#)

4.68. Referencia del Namespace components.alumnado

Namespaces

- namespace [filters](#)
- namespace [graphs](#)
- namespace [utils](#)

4.69. Referencia del Namespace components.alumnado.filters

Namespaces

- namespace [filter_asignaturas_matri_alumnado](#)
- namespace [filter_curso_academico_alumnado](#)
- namespace [filter_titulacion_alumnado](#)

4.70. Referencia del Namespace components.alumnado.filters.filter_asignaturas_matri_alumnado

Funciones

- def filter_asignaturas_matri_alumnado ()

4.70.1. Documentación de las funciones

4.70.1.1. filter_asignaturas_matri_alumnado()

```
def components.alumnado.filters.filter_asignaturas_matri_alumnado.filter_asignaturas_matri_←
alumnado ( )
```

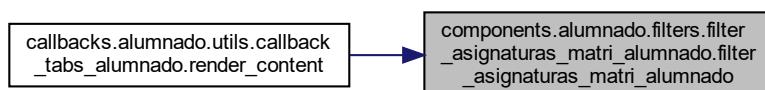
Crea un componente con un dropdown que contiene las asignaturas matriculadas por el alumno.

Returns:
html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas

Definición en la línea 15 del archivo [filter_asignaturas_matri_alumnado.py](#).

```
00015 def filter_asignaturas_matri_alumnado():
00016     """
00017     Crea un componente con un dropdown que contiene las asignaturas matriculadas por el alumno.
00018
00019     Returns:
00020     html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
00021     """
00022     return html.Div(
00023         [
00024             html.Br(),
00025             html.Label("Asignaturas matriculadas"),
00026             dcc.Dropdown(
00027                 id="asignaturas-matriculadas",
00028                 searchable=True,
00029                 multi=True,
00030                 value=None,
00031                 clearable=True,
00032                 options=[],
00033                 maxHeight=300,
00034                 optionHeight=50,
00035                 placeholder="Seleccione una opción",
00036             ),
00037             html.Button(
00038                 "Seleccionar todo",
00039                 id="select-all-button",
00040                 className="button-select-all-filter",
00041                 n_clicks=0,
00042             ),
00043         ],
00044     )
```

Gráfico de llamadas a esta función:



4.71. Referencia del Namespace

components.alumnado.filters.filter_curso_academico_alumnado

Funciones

- def filter_curso_academico_alumnado ()

4.71.1. Documentación de las funciones

4.71.1.1. filter_curso_academico_alumnado()

```
def components.alumnado.filters.filter_curso_academico_alumnado.filter_curso_academico_←
alumnado ( )
```

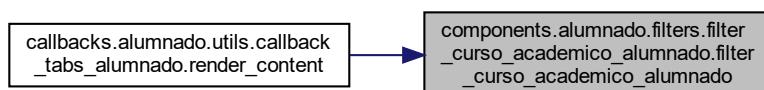
Crea un componente con un dropdown que contiene los cursos académicos.

Returns:
 html.Div: Componente con un dropdown y un botón para seleccionar todos los cursos académicos

Definición en la línea 15 del archivo [filter_curso_academico_alumnado.py](#).

```
00015 def filter_curso_academico_alumnado():
00016     """
00017     Crea un componente con un dropdown que contiene los cursos académicos.
00018
00019     Returns:
00020     html.Div: Componente con un dropdown y un botón para seleccionar todos los cursos académicos
00021     """
00022     return html.Div(
00023         [
00024             html.Br(),
00025             html.Label("Curso académico"),
00026             dcc.Dropdown(
00027                 id="curso-academico",
00028                 searchable=False,
00029                 multi=True,
00030                 clearable=True,
00031                 options=[],
00032                 value=None,
00033                 maxHeight=300,
00034                 placeholder="Seleccione una opción",
00035             ),
00036             html.Button(
00037                 "Seleccionar todo",
00038                 id="select-all-cursos-academicos",
00039                 className="button-select-all-filter",
00040                 n_clicks=0,
00041             ),
00042         ],
00043     )
```

Gráfico de llamadas a esta función:



4.72. Referencia del Namespace components.alumnado.filters.filter_titulacion_alumnado

Funciones

- def filter_titulacion_alumnado ()

4.72.1. Documentación de las funciones

4.72.1.1. filter_titulacion_alumnado()

```
def components.alumnado.filters.filter_titulacion_alumnado.filter_titulacion_alumnado ( )
```

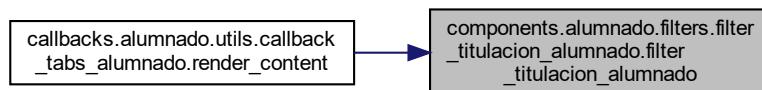
Crea un componente con un dropdown que contiene las titulaciones del perfil "Alumno".

Returns:
 html.Div: Componente con un dropdown y almacenamiento de la titulación seleccionada

Definición en la línea 15 del archivo [filter_titulacion_alumnado.py](#).

```
00015 def filter_titulacion_alumnado():
00016     """
00017     Crea un componente con un dropdown que contiene las titulaciones del perfil "Alumno".
00018
00019     Returns:
00020         html.Div: Componente con un dropdown y almacenamiento de la titulación seleccionada
00021     """
00022     return html.Div(
00023         [
00024             html.Label("Titulación"),
00025             dcc.Dropdown(
00026                 id="titulacion-alumnado",
00027                 options=[],
00028                 value=None,
00029                 searchable=False,
00030                 clearable=False,
00031                 optionHeight=50,
00032                 maxHeight=300,
00033                 placeholder="Seleccione una opción",
00034             ),
00035             dcc.Store(id="selected-titulacion-alumnado-store", storage_type="local"),
00036         ]
00037     )
```

Gráfico de llamadas a esta función:



4.73. Referencia del Namespace components.alumnado.graphs

Namespaces

- namespace [graphs_general_alumnado](#)
- namespace [graphs_personal_alumnado](#)

4.74. Referencia del Namespace components.alumnado.graphs.graphs_general_alumnado

Funciones

- def [graphs_general_alumnado\(\)](#)

4.74.1. Documentación de las funciones

4.74.1.1. graphs_general_alumnado()

```
def components.alumnado.graphs.graphs_general_alumnado.graphs_general_alumnado( )
```

Retona los gráficos de la pestaña "Rendimiento Académico general" del perfil "Alumnado"

Returns:
 html.Div: Gráficos

Definición en la línea 18 del archivo [graphs_general_alumnado.py](#).

```
00018 def graphs_general_alumnado():
00019     """
00020     Retona los gráficos de la pestaña "Rendimiento Académico general" del perfil "Alumnado"
00021
00022     Returns:
00023         html.Div: Gráficos
00024     """
00025 graph_ids = [
00026     "asignaturas-superadas-general-mi-nota",
00027     "nota-cualitativa-general-mi-nota",
00028     "nota-media-general-mi-nota",
00029 ]
00030
00031 config = {"displayModeBar": False}
00032 item_class = "graph-item-general-alumnado"
00033
00034 return html.Div(
00035     [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
00036     className="graphs-container-general-alumnado",
00037 )
```

Gráfico de llamadas para esta función:

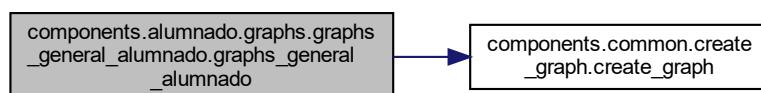
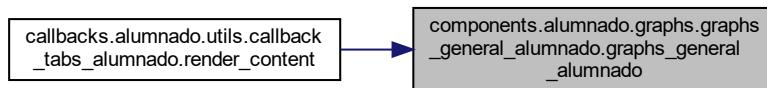


Gráfico de llamadas a esta función:



4.75. Referencia del Namespace components.alumnado.graphs.graphs_personal_alumnado

Funciones

- def [graphs_personal_alumnado\(\)](#)

4.75.1. Documentación de las funciones

4.75.1.1. graphs_personal_alumnado()

```
def components.alumnado.graphs.graphs_personal_alumnado.graphs_personal_alumnado( )
```

Retorna los gráficos de la pestaña "Expediente académico Personal" del perfil "Alumnado"

Returns:
html.Div: Gráficos

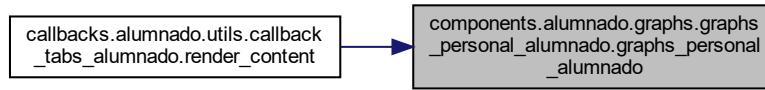
Definición en la línea 19 del archivo [graphs_personal_alumnado.py](#).

```
00019 def graphs_personal_alumnado():
00020     """
00021     Retorna los gráficos de la pestaña "Expediente académico Personal" del perfil "Alumnado"
00022
00023     Returns:
00024         html.Div: Gráficos
00025     """
00026 graph_ids = [
00027     "graph-evolucion-progreso-academico",
00028     "graph-bar-evolucion-asignaturas-matriculadas",
00029     "graph-bar-calificaciones-por-asignatura",
00030     "graph-bar-tasa-rendimiento",
00031 ]
00032 config = {"displayModeBar": False}
00033 item_class = "graph-item-personal-alumnado"
00035
00036     return html.Div(
00037         [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
00038         className="graphs-container-personal-alumnado",
00039     )
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.76. Referencia del Namespace `components.alumnado.utils`

Namespaces

- namespace `recomendador_alumnado`
- namespace `resumen_alumnado`
- namespace `select_alumnado`
- namespace `tabs_alumnado`

4.77. Referencia del Namespace `components.alumnado.utils.recomendador_alumnado`

Funciones

- def `recomendador_alumnado ()`

4.77.1. Documentación de las funciones

4.77.1.1. recomendador_alumnado()

```
def components.alumnado.utils.recomendador_alumnado.recomendador_alumnado ( )
```

Esta función retorna el contenido de la pestaña "Recomendador" del perfil "Alumnado".

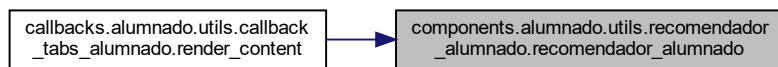
Returns:

html.Div: Layout de la pestaña "Recomendador" del perfil "Alumnado"

Definición en la línea 14 del archivo [recomendador_alumnado.py](#).

```
00014 def recomendador_alumnado():
00015     """
00016 Esta función retorna el contenido de la pestaña "Recomendador" del perfil "Alumnado".
00017
00018     Returns:
00019         html.Div: Layout de la pestaña "Recomendador" del perfil "Alumnado"
00020     """
00021
00022     return html.Div([
00023         html.H1("Deserción universitaria: ¿Cuáles son las razones y cómo prevenirla?", className='titulo-recomendador-alumnado'),
00024         html.Img(src='assets/images/abandono_academico.jpg', className='imagen-recomendador-alumnado'),
00025         html.P("La deserción universitaria es un problema que puede afectar a muchas instituciones educativas, pero que puede evitarse con las estrategias adecuadas. Reducir las tasas de deserción universitaria es uno de los grandes retos de las Instituciones de Educación Superior (IES).", className='p-recomendador-alumnado'),
00026         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-alumnado'),
00027         html.P("La deserción universitaria es el abandono de los estudios universitarios por parte de un estudiante, sin haber finalizado la carrera. Esto puede tener graves consecuencias para tu futuro y tu bienestar.", className='p-recomendador-alumnado'),
00028         html.Br(),
00029         html.Br(),
00030         html.H2("¿Cuáles son las razones de la deserción universitaria?", className='sb-recomendador-alumnado'),
00031         html.P("Existen muchas razones por las que un estudiante puede abandonar sus estudios universitarios. Algunas de las razones más comunes son:", className='p-recomendador-alumnado'),
00032         html.Ul([
00033             html.Li(html.Span([html.Strong("Problemas económicos:"), " La falta de recursos económicos puede dificultar tu acceso a la educación superior."]), className='li-recomendador-alumnado'),
00034             html.Li(html.Span([html.Strong("Problemas académicos:"), " Las dificultades para aprobar asignaturas pueden hacer que te sientas frustrado."]), className='li-recomendador-alumnado'),
00035             html.Li(html.Span([html.Strong("Problemas personales:"), " Situaciones como problemas de salud, familiares o relaciones personales pueden afectarte."]), className='li-recomendador-alumnado'),
00036             html.Li(html.Span([html.Strong("Falta de motivación:"), " La falta de interés en los estudios puede llevarte a abandonar la universidad."]), className='li-recomendador-alumnado'),
00037             html.Li(html.Span([html.Strong("Falta de orientación:"), " No recibir la orientación adecuada puede dificultar tu adaptación a la vida universitaria."]), className='li-recomendador-alumnado'),
00038         ], className='ul-recomendador-alumnado'),
00039         html.H2("¿Cómo prevenir la deserción universitaria?", className='sb-recomendador-alumnado'),
00040         html.P("Para prevenir la deserción, puedes buscar ayuda en los siguientes recursos que la universidad ofrece:", className='p-recomendador-alumnado'),
00041         html.Ul([
00042             html.Li("Solicitar becas y ayudas económicas si tienes dificultades financieras.", className='li-recomendador-alumnado'),
00043             html.Li("Participar en programas de tutoría y apoyo académico si tienes problemas con tus asignaturas.", className='li-recomendador-alumnado'),
00044             html.Li("Aprovechar los programas de orientación y asesoramiento para adaptarte mejor a la vida universitaria.", className='li-recomendador-alumnado'),
00045             html.Li("Asistir a actividades y talleres que mejoren tu motivación y habilidades de estudio.", className='li-recomendador-alumnado'),
00046         ], className='ul-recomendador-alumnado'),
00047     ])
```

Gráfico de llamadas a esta función:



4.78. Referencia del Namespace components.alumnado.utils.resumen_alumnado

Funciones

- def [resumen_alumnado\(\)](#)

4.78.1. Documentación de las funciones

4.78.1.1. [resumen_alumnado\(\)](#)

```
def components.alumnado.utils.resumen_alumnado.resumen_alumnado ( )
```

Crea el layout del resumen del alumnado.

Returns:
 html.Div: Layout del resumen del alumnado

Definición en la línea 15 del archivo [resumen_alumnado.py](#).

```
00015 def resumen_alumnado():
00016     """
00017     Crea el layout del resumen del alumnado.
00018
00019 Returns:
00020     html.Div: Layout del resumen del alumnado
00021 """
00022 return html.Div([], id="resumen-alumnado")
```

Gráfico de llamadas a esta función:



4.79. Referencia del Namespace components.alumnado.utils.select_alumnado

Funciones

- def [select_alumnado\(\)](#)

4.79.1. Documentación de las funciones

4.79.1.1. select_alumnado()

```
def components.alumnado.utils.select_alumnado.select_alumnado ( )
```

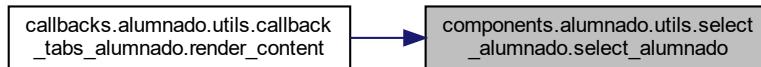
Crea el desplegable para seleccionar un alumno.

Returns:
html.Div: Desplegable para seleccionar un alumno

Definición en la línea 15 del archivo [select_alumnado.py](#).

```
00015 def select_alumnado():
00016     """
00017 Crea el desplegable para seleccionar un alumno.
00018
00019 Returns:
00020 html.Div: Desplegable para seleccionar un alumno
00021 """
00022 return html.Div(
00023     [
00024         html.Div(
00025             [
00026                 dcc.Dropdown(
00027                     options=[],
00028                     value=None,
00029                     id="alumnado-dropdown",
00030                     clearable=False,
00031                     placeholder="Seleccione un alumno",
00032                 )
00033             ],
00034             className="select-alumnado",
00035         ),
00036     ]
00037 )
```

Gráfico de llamadas a esta función:



4.80. Referencia del Namespace components.alumnado.utils.tabs_alumnado

Funciones

- def tabs_alumnado ()

4.80.1. Documentación de las funciones

4.80.1.1. tabs_alumnado()

```
def components.alumnado.utils.tabs_alumnado( )
```

Contiene las pestañas de la vista del alumnado.

Returns:
 html.Div: Pestañas del alumnado

Definición en la línea 15 del archivo [tabs_alumnado.py](#).

```
00015 def tabs_alumnado():
00016     """
00017     Contiene las pestañas de la vista del alumnado.
00018
00019     Returns:
00020     html.Div: Pestañas del alumnado
00021     """
00022     return html.Div(
00023         [
00024             dcc.Tabs(
00025                 id="tabs-alumnado",
00026                 value="expediente-personal-tab",
00027                 children=[
00028                     dcc.Tab(
00029                         label="Expediente académico personal",
00030                         value="expediente-personal-tab",
00031                     ),
00032                     dcc.Tab(
00033                         label="Rendimiento académico general",
00034                         value="rendimiento-academico-tab",
00035                     ),
00036                     dcc.Tab(label="Recomendaciones", value="recomendador-tab"),
00037                 ],
00038                 className="tabs",
00039             ),
00040             html.Div(id="tabs-alumnado-content"),
00041             dcc.Store(id="selected-alumnado-store", storage_type="local"),
00042             dcc.Location(id="url", refresh=False),
00043         ]
00044     )
```

4.81. Referencia del Namespace components.common

Namespaces

- namespace [create_graph](#)
- namespace [create_graph_with_table](#)
- namespace [filters](#)
- namespace [footer](#)
- namespace [header](#)
- namespace [modal_data](#)
- namespace [sidebar](#)

4.82. Referencia del Namespace components.common.create_graph

Funciones

- def [create_graph](#) (graph_id, item_class, config)

4.82.1. Documentación de las funciones

4.82.1.1. create_graph()

```
def components.common.create_graph.create_graph (
    graph_id,
    item_class,
    config )
```

Crea un gráfico con un id, una clase y una configuración específica.

Args:

- graph_id (str): ID del gráfico
- item_class (str): Clase del gráfico
- config (dict): Configuración del gráfico

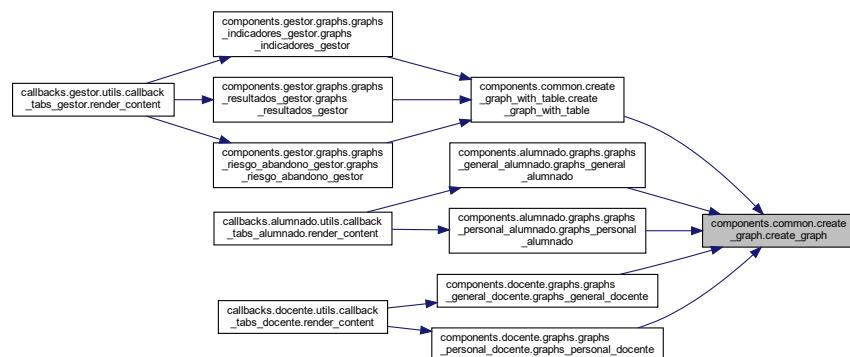
Returns:

- html.Div: Gráfico

Definición en la línea 14 del archivo [create_graph.py](#).

```
00014 def create_graph(graph_id, item_class, config):
00015     """
00016     Crea un gráfico con un id, una clase y una configuración específica.
00017
00018 Args:
00019     graph_id (str): ID del gráfico
00020     item_class (str): Clase del gráfico
00021     config (dict): Configuración del gráfico
00022
00023 Returns:
00024     html.Div: Gráfico
00025 """
00026     return html.Div(
00027         [dcc.Loading(children=[dcc.Graph(id=graph_id, figure={}, config=config)]),
00028          className=item_class,
00029      )
```

Gráfico de llamadas a esta función:



4.83. Referencia del Namespace components.common.create_graph_with_table

Funciones

- def [create_graph_with_table](#) (graph_id, item_class, config, modal_config)

4.83.1. Documentación de las funciones

4.83.1.1. create_graph_with_table()

```
def components.common.create_graph_with_table.create_graph_with_table (
    graph_id,
    item_class,
    config,
    modal_config )
```

Crea un gráfico con un modal para mostrar los datos y descargarlos.

Args:

```
graph_id (str): ID del gráfico
item_class (str): Clase del gráfico
config (dict): Configuración del gráfico
modal_config (dict): Configuración del modal
```

Returns:

```
html.Div: Gráfico con modal para mostrar los datos
```

Definición en la línea 16 del archivo [create_graph_with_table.py](#).

```
00016 def create_graph_with_table(graph_id, item_class, config, modal_config):
00017     """
00018     Crea un gráfico con un modal para mostrar los datos y descargarlos.
00019
00020 Args:
00021     graph_id (str): ID del gráfico
00022     item_class (str): Clase del gráfico
00023     config (dict): Configuración del gráfico
00024     modal_config (dict): Configuración del modal
00025
00026 Returns:
00027     html.Div: Gráfico con modal para mostrar los datos
00028     """
00029     return html.Div(
00030         [create_graph(graph_id, item_class, config), create_modal(**modal_config)],
00031         className=item_class,
00032     )
```

Gráfico de llamadas para esta función:

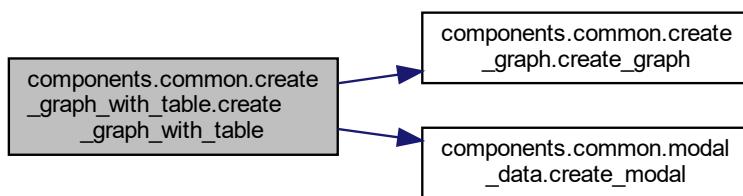
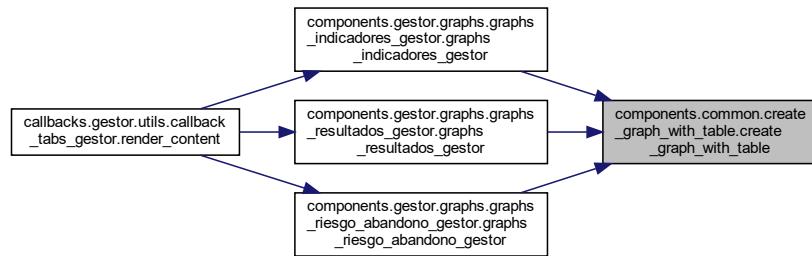


Gráfico de llamadas a esta función:



4.84. Referencia del Namespace components.common.filters

Funciones

- def **filters** (filters)

4.84.1. Documentación de las funciones

4.84.1.1. filters()

```
def components.common.filters.filters (
    filters )
```

Crea un componente que contiene los filtros de la aplicación.

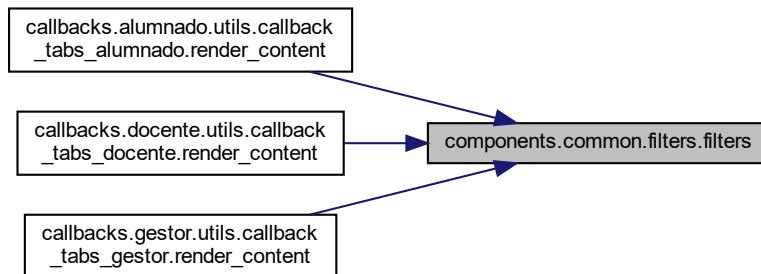
Args:
filters (list): Lista de filtros

Returns:
html.Div: Componente que contiene los filtros

Definición en la línea 14 del archivo [filters.py](#).

```
00014 def filters(filters):
00015     """
00016     Crea un componente que contiene los filtros de la aplicación.
00017
00018     Args:
00019     filters (list): Lista de filtros
00020
00021     Returns:
00022     html.Div: Componente que contiene los filtros
00023     """
00024     return html.Div([html.H2("Filtros"), html.Div(filters)], className="filters")
```

Gráfico de llamadas a esta función:



4.85. Referencia del Namespace components.common.footer

Funciones

- def `footer()`

4.85.1. Documentación de las funciones

4.85.1.1. `footer()`

```
def components.common.footer.footer( )
```

Retorna el footer de la aplicación

Returns:
`html.Div`: Footer de la aplicación

Definición en la línea 14 del archivo `footer.py`.

```
00014 def footer():
00015     """
00016     Retorna el footer de la aplicación
00017
00018     Returns:
00019     html.Div: Footer de la aplicación
00020     """
00021     return html.Div([
00022         html.P('2023-2024 Universidad de la Laguna - Visualización de datos académicos'),
00023         html.P('Pabellón de Gobierno, C/ Padre Herrera s/n Apartado Postal 456 38200, San Cristóbal de
La Laguna'),
00024         html.P('Santa Cruz de Tenerife - España')], className='footer')
```

4.86. Referencia del Namespace components.common.header

Funciones

- def `header(store_role)`

4.86.1. Documentación de las funciones

4.86.1.1. header()

```
def components.common.header.header (
    store_role )
```

Contiene el header de la aplicación y el desplegable para seleccionar el rol.

Args:
`store_role: str: ID del store que contiene el rol del usuario`

Returns:
`html.Div: Header de la aplicación`

Definición en la línea 15 del archivo [header.py](#).

```
00015 def header(store_role):
00016     """
00017     Contiene el header de la aplicación y el desplegable para seleccionar el rol.
00018
00019     Args:
00020         store_role: str: ID del store que contiene el rol del usuario
00021
00022     Returns:
00023         html.Div: Header de la aplicación
00024     """
00025     return html.Div(
00026         [
00027             html.Img(src="assets/images/logoULL.png", className="logo"),
00028             html.H1("Visualización de datos académicos", className="title"),
00029             dcc.Dropdown(
00030                 options=[
00031                     {"label": "Alumno", "value": "Alumno"},
00032                     {"label": "Docente", "value": "Docente"},
00033                     {"label": "Gestor", "value": "Gestor"},
00034                 ],
00035                 id="dropdown_role",
00036                 className="dropdown_role",
00037                 clearable=False,
00038                 searchable=False,
00039                 placeholder="Selecciona un rol",
00040                 value="Alumno",
00041             ),
00042             dcc.Store(id=store_role, storage_type="session"),
00043         ],
00044         className="header",
00045     )
```

4.87. Referencia del Namespace components.common.modal_data

Funciones

- def [create_modal](#) (modal_id, table_container_id, download_button_id, view_data_button_id)

4.87.1. Documentación de las funciones

4.87.1.1. create_modal()

```
def components.common.modal_data.create_modal (
    modal_id,
    table_container_id,
    download_button_id,
    view_data_button_id )
```

Crea el modal para mostrar los datos de una tabla. El modal tiene un botón para descargar los datos en formato CSV.

Args:

```
    modal_id (str): ID del modal
    table_container_id (str): ID del contenedor de la tabla
    download_button_id (str): ID del botón para descargar los datos
    view_data_button_id (str): ID del botón para abrir el modal
```

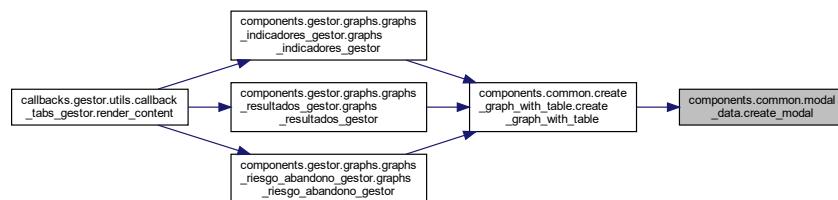
Returns:

```
    dbc.Modal: Modal para mostrar los datos
```

Definición en la línea 15 del archivo [modal_data.py](#).

```
00015 def create_modal(modal_id, table_container_id, download_button_id, view_data_button_id):
00016     """
00017     Crea el modal para mostrar los datos de una tabla. El modal tiene un botón para descargar los datos en
00018     formato CSV.
00019     Args:
00020         modal_id (str): ID del modal
00021         table_container_id (str): ID del contenedor de la tabla
00022         download_button_id (str): ID del botón para descargar los datos
00023         view_data_button_id (str): ID del botón para abrir el modal
00024
00025     Returns:
00026         dbc.Modal: Modal para mostrar los datos
00027     """
00028     return html.Div([
00029         dbc.Button('Ver datos', id=view_data_button_id, n_clicks=0, color='primary'),
00030         dbc.Modal([
00031             dbc.ModalHeader("Datos"),
00032             dbc.ModalBody(html.Div(id=table_container_id), style={'maxHeight': 'calc(100vh - 200px)' ,
00033             'overflowY': 'auto'}),
00034             dbc.ModalFooter(
00035                 dbc.Button("Descargar CSV", id=download_button_id, color='primary')
00036             ), id=modal_id, is_open=False, size="lg")
00037     ])
```

Gráfico de llamadas a esta función:



4.88. Referencia del Namespace `components.common.sidebar`

Funciones

- def `sidebar` (`components`)

4.88.1. Documentación de las funciones

4.88.1.1. sidebar()

```
def components.common.sidebar.sidebar (
    components )
```

Sidebar del dashboard donde se encuentra el resumen de la información y los filtros

Args:

components (list): Componentes que se desplegarán en el sidebar

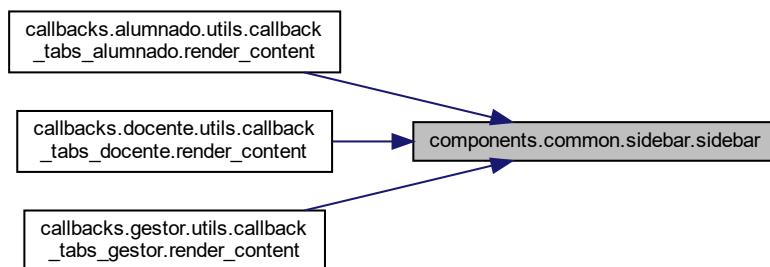
Returns:

html.Div: Sidebar del dashboard

Definición en la línea 16 del archivo [sidebar.py](#).

```
00016 def sidebar(components):
00017     """
00018 Sidebar del dashboard donde se encuentra el resumen de la información y los filtros
00019
00020 Args:
00021 components (list): Componentes que se desplegarán en el sidebar
00022
00023 Returns:
00024 html.Div: Sidebar del dashboard
00025 """
00026 sidebar_content = html.Div(components, className="sidebar")
00027     sidebarCollapse = dbc.Collapse(sidebar_content, id="collapse", is_open=True)
00028
00029 return html.Div(sidebarCollapse)
```

Gráfico de llamadas a esta función:



4.89. Referencia del Namespace components.docente

Namespaces

- namespace [filters](#)
- namespace [graphs](#)
- namespace [utils](#)

4.90. Referencia del Namespace components.docente.filters

Namespaces

- namespace `filter_all_asignaturas_titulacion_docente`
- namespace `filter_all_curso_academico`
- namespace `filter_asignaturas_docente`
- namespace `filter_curso_academico_docente`
- namespace `filter_titulacion_docente`

4.91. Referencia del Namespace components.docente.filters.filter_all_asignaturas_titulacion_docente

Funciones

- def `filter_all_asignaturas_titulacion_docente ()`

4.91.1. Documentación de las funciones

4.91.1.1. filter_all_asignaturas_titulacion_docente()

```
def components.docente.filters.filter_all_asignaturas_titulacion_docente.filter_all_asignaturas_titulacion_docente ( )
```

Crea un componente con un dropdown que contiene todas las asignaturas de la titulación del perfil "Docente" en la pestaña "Rendimiento académico general".

Returns:

`html.Div`: Componente con un dropdown y un botón para seleccionar todas las asignaturas

Definición en la línea 16 del archivo `filter_all_asignaturas_titulacion_docente.py`.

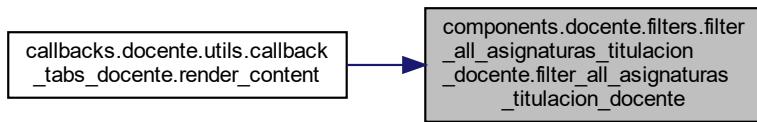
```
00016 def filter_all_asignaturas_titulacion_docente():
00017     """
00018     Crea un componente con un dropdown que contiene todas las asignaturas de la titulación del perfil
00019     "Docente"
00020     en la pestaña "Rendimiento académico general".
00021
00022     Returns:
00023         html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
00024
00025     return html.Div(
00026         [
00027             html.Br(),
00028             html.Label("Asignaturas"),
00029             dcc.Dropdown(
00030                 id="all-asignaturas-titulacion-docente",
00031                 searchable=True,
00032                 value=None,
00033                 clearable=True,
00034                 options=[],
00035                 maxHeight=300,
00036                 optionHeight=50,
00037                 multi=True,
00038             ),
00039             html.Button(
00040                 "Seleccionar todo",
00041             )
00042         ]
00043     )
00044
00045     
```

```

00040         id="select-all-asignaturas-titulacion-docente",
00041         className="button-select-all-filter",
00042         n_clicks=0,
00043     ),
00044 ]
00045 )

```

Gráfico de llamadas a esta función:



4.92. Referencia del Namespace components.docente.filters.filter_all_curso_academico

Funciones

- def [filter_all_curso_academico\(\)](#)

4.92.1. Documentación de las funciones

4.92.1.1. filter_all_curso_academico()

```
def components.docente.filters.filter_all_curso_academico.filter_all_curso_academico ( )
```

Crea un componente con un dropdown que contiene todos los cursos académicos del perfil "Docente" en la pestaña "Rendimiento académico general".

Returns:
.Div: Componente con un dropdown

Definición en la línea 15 del archivo [filter_all_curso_academico.py](#).

```

00015 def filter_all_curso_academico():
00016     """
00017     Crea un componente con un dropdown que contiene todos los cursos académicos del perfil "Docente"
00018     en la pestaña "Rendimiento académico general".
00019
00020     Returns:
00021         .Div: Componente con un dropdown
00022     """
00023     return html.Div(
00024         [
00025             html.Br(),
00026             html.Label("Curso académico"),
00027             dcc.Dropdown(
00028                 id="all-cursos-academicos-docente",
00029                 searchable=False,
00030                 multi=False,

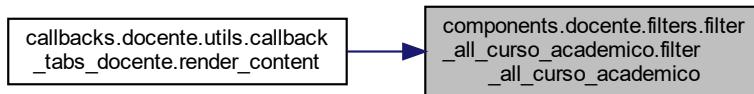
```

```

00031         clearable=False,
00032         options=[],
00033         value=None,
00034         maxHeight=300,
00035     ),
00036 ],
00037 )

```

Gráfico de llamadas a esta función:



4.93. Referencia del Namespace

`components.docente.filters.filter_asignaturas_docente`

Funciones

- def `filter_asignaturas_docente()`

4.93.1. Documentación de las funciones

4.93.1.1. `filter_asignaturas_docente()`

```
def components.docente.filters.filter_asignaturas_docente.filter_asignaturas_docente( )
```

Crea un componente con un dropdown que contiene las asignaturas del perfil "Docente" en la pestaña "Rendimiento académico personal".

Returns:
`html.Div`: Componente con un dropdown

Definición en la línea 15 del archivo `filter_asignaturas_docente.py`.

```

00015 def filter_asignaturas_docente():
00016     """
00017     Crea un componente con un dropdown que contiene las asignaturas del perfil "Docente"
00018     en la pestaña "Rendimiento académico personal".
00019
00020     Returns:
00021         html.Div: Componente con un dropdown
00022     """
00023
00024     return html.Div([
00025         html.Br(),
00026         html.Label("Asignaturas"),
00027         dcc.Dropdown(
00028             id="asignaturas-docente",
00029             searchable=True,
00030             value=None,

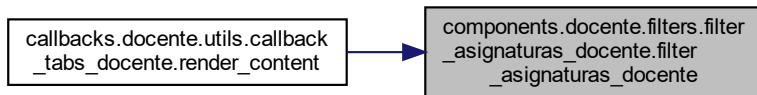
```

```

00031         clearable=True,
00032         options=[],
00033         maxHeight=300,
00034         optionHeight=50,
00035     )
00036 ]
)

```

Gráfico de llamadas a esta función:



4.94. Referencia del Namespace components.docente.filters.filter_curso_academico_docente

Funciones

- def [filter_curso_academico_docente \(\)](#)

4.94.1. Documentación de las funciones

4.94.1.1. filter_curso_academico_docente()

```
def components.docente.filters.filter_curso_academico_docente.filter_curso_academico_docente (
)
```

Crea un componente con un dropdown que contiene los cursos académicos del perfil "Docente" en la pestaña "Rendimiento académico personal".

Returns:
html.Div: Componente con un dropdown

Definición en la línea 14 del archivo [filter_curso_academico_docente.py](#).

```

00014 def filter_curso_academico_docente():
00015     """
00016 Crea un componente con un dropdown que contiene los cursos académicos del perfil "Docente"
00017     en la pestaña "Rendimiento académico personal".
00018
00019     Returns:
00020         html.Div: Componente con un dropdown
00021     """
00022 return html.Div(
00023     [
00024         html.Br(),
00025         html.Label("Curso académico"),
00026         dcc.Dropdown(
00027             id="curso-academico-docente",
00028             searchable=False,
00029             multi=True,

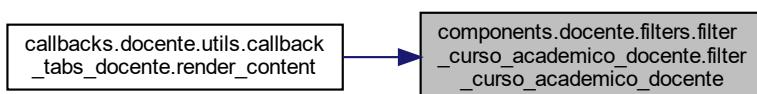
```

```

00030         clearable=True,
00031         options=[],
00032         value=None,
00033         maxHeight=300,
00034     ),
00035     html.Button(
00036         "Seleccionar todo",
00037         id="select-all-cursos-academicos-docente",
00038         className="button-select-all-filter",
00039         n_clicks=0,
00040     ),
00041 ],
00042 )

```

Gráfico de llamadas a esta función:



4.95. Referencia del Namespace **components.docente.filters.filter_titulacion_docente**

Funciones

- def **filter_titulacion_docente ()**

4.95.1. Documentación de las funciones

4.95.1.1. filter_titulacion_docente()

```
def components.docente.filters.filter_titulacion_docente.filter_titulacion_docente ( )
```

Crea un componente con un dropdown que contiene las titulaciones del perfil "Docente" en la pestaña "Rendimiento académico personal" y "Rendimiento académico general".

Returns:
html.Div: Componente con un dropdown

Definición en la línea 15 del archivo **filter_titulacion_docente.py**.

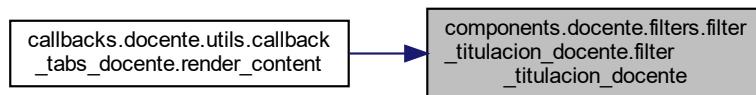
```

00015 def filter_titulacion_docente():
00016     """
00017     Crea un componente con un dropdown que contiene las titulaciones del perfil "Docente"
00018     en la pestaña "Rendimiento académico personal" y "Rendimiento académico general".
00019
00020     Returns:
00021         html.Div: Componente con un dropdown
00022     """
00023
00024 return html.Div(

```

```
00025      [
00026          html.Label("Titulación"),
00027          dcc.Dropdown(
00028              id="titulacion-docente",
00029              options=[],
00030              value=None,
00031              searchable=False,
00032              clearable=False,
00033              optionHeight=50,
00034              maxHeight=300,
00035          ),
00036          dcc.Store(id="selected-titulacion-docente-store", storage_type="local"),
00037      ]
00038  )
```

Gráfico de llamadas a esta función:



4.96. Referencia del Namespace components.docente.graphs

Namespaces

- namespace [graphs_general_docente](#)
- namespace [graphs_personal_docente](#)

4.97. Referencia del Namespace

components.docente.graphs.graphs_general_docente

Funciones

- def [graphs_general_docente \(\)](#)

4.97.1. Documentación de las funciones

4.97.1.1. graphs_general_docente()

```
def components.docente.graphs.graphs_general_docente().graphs_general_docente( )
```

Retorna los gráficos de la pestaña "Rendimiento académico general" del perfil "Docente"

Returns:
html.Div: Gráficos

Definición en la línea 19 del archivo [graphs_general_docente.py](#).

```
00019 def graphs_general_docente():
00020     """
00021     Retorna los gráficos de la pestaña "Rendimiento académico general"
00022         del perfil "Docente"
00023
00024     Returns:
00025         html.Div: Gráficos
00026     """
00027
00028 graphs_ids = [
00029     "calificaciones-cuali-all-asig-docente",
00030     "calificaciones-media-all-asig-docente",
00031 ]
00032
00033     item_class = "graph-item-general-docente"
00034     config = config_mode_bar_buttons_gestor
00035
00036     return html.Div(
00037         [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
00038         className="graphs-container-general-docente",
00039     )
```

Gráfico de llamadas para esta función:

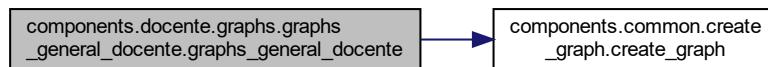
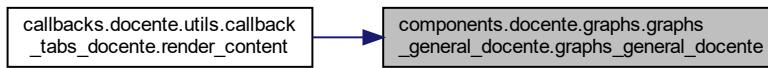


Gráfico de llamadas a esta función:



4.98. Referencia del Namespace components.docente.graphs.graphs_personal_docente

Funciones

- [def graphs_personal_docente \(\)](#)

4.98.1. Documentación de las funciones

4.98.1.1. graphs_personal_docente()

```
def components.docente.graphs.graphs_personal_docente().graphs_personal_docente( )
```

Retorna los gráficos de la pestaña "Rendimiento académico personal" del perfil "Docente"

Returns:
 html.Div: Gráficos

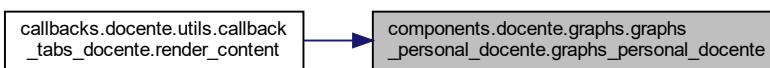
Definición en la línea 21 del archivo [graphs_personal_docente.py](#).

```
00021 def graphs_personal_docente():
00022     """
00023     Retorna los gráficos de la pestaña "Rendimiento académico personal"
00024     del perfil "Docente"
00025
00026     Returns:
00027         html.Div: Gráficos
00028     """
00029 graphs_ids = [
00030     "graph-alumnos-repetidores-nuevos",
00031     "graph-alumnos-matri-genero",
00032     "graph-alumnos-nota-media",
00033     "graph-alumnos-nota-cualitativa",
00034 ]
00035
00036     item_class = "graph-item-personal-docente"
00037     config = config_mode_bar_buttons_gestor
00038
00039     return html.Div(
00040         [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
00041         className="graphs-container-personal-docente",
00042     )
00043
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.99. Referencia del Namespace components.docente.utils

Namespaces

- namespace [recomendador_docente](#)
- namespace [resumen_docente](#)
- namespace [select_docente](#)
- namespace [tabs_docente](#)

4.100. Referencia del Namespace components.docente.utils.recomendador_docente

Funciones

- def [recomendador_docente \(\)](#)

4.100.1. Documentación de las funciones

4.100.1.1. [recomendador_docente\(\)](#)

```
def components.docente.utils.recomendador_docente ()
```

Retorna el componente del recomendador para docentes.

Returns:
 html.Div: Componente del recomendador para docentes

Definición en la línea 13 del archivo [recomendador_docente.py](#).

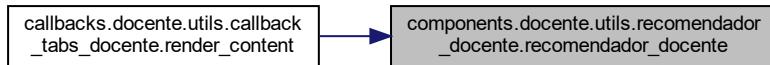
```
00013 def recomendador_docente():
00014     """
00015 Retorna el componente del recomendador para docentes.
00016
00017 Returns:
00018 html.Div: Componente del recomendador para docentes
00019 """
00020
00021     return html.Div([
00022         html.H1("Deserción universitaria: ¿Cómo pueden ayudar los docentes a prevenirla?", className='titulo-recomendador-docente'),
00023         html.Img(src='assets/images/docente_clase.jpg', className='imagen-recomendador-docente'),
00024         html.P("La deserción universitaria es un problema serio que afecta a muchas instituciones educativas. Los docentes juegan un papel crucial en la identificación y prevención de este fenómeno.", className='p-recomendador-docente'),
00025         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-docente'),
00026         html.P("La deserción universitaria es el abandono de los estudios universitarios por parte de un estudiante, sin haber finalizado la carrera. Este problema puede tener graves consecuencias tanto para los estudiantes como para la institución.", className='p-recomendador-docente'),
00027         html.Br(),
00028         html.H2("¿Cómo pueden los docentes ayudar a prevenir la deserción universitaria?", className='sb-recomendador-docente'),
00029         html.Ul([
00030             html.Li("Identificar y apoyar a los estudiantes con dificultades académicas proporcionando tutorías y asesorías.", className='li-recomendador-docente'),
00031             html.Li("Fomentar un ambiente de clase inclusivo y motivador que promueva la participación y el interés por el aprendizaje.", className='li-recomendador-docente'),
00032             html.Li("Mantener una comunicación abierta y constante con los estudiantes para entender sus problemas y necesidades.", className='li-recomendador-docente'),
```

```

00033         html.Li("Colaborar con los servicios de apoyo estudiantil para proporcionar la ayuda
00034         necesaria en caso de problemas personales o económicos.", className='li-recomendador-docente'),
00035             html.Li("Participar en programas de formación continua para mejorar las estrategias
00036             pedagógicas y adaptarse a las necesidades cambiantes de los estudiantes.", className='li-recomendador-docente'),
00037         ], className='ul-recomendador-docente'),
00038     )

```

Gráfico de llamadas a esta función:



4.101. Referencia del Namespace components.docente.utils.resumen_docente

Funciones

- def `resumen_docente ()`

4.101.1. Documentación de las funciones

4.101.1.1. `resumen_docente()`

```
def components.docente.utils.resumen_docente.resumen_docente ( )
```

Crea el layout del resumen del docente

Returns:
`html.Div`: Layout del resumen del docente

Definición en la línea 15 del archivo `resumen_docente.py`.

```

00015 def resumen_docente():
00016     """
00017 Crea el layout del resumen del docente
00018
00019 Returns:
00020     html.Div: Layout del resumen del docente
00021 """
00022 return html.Div([], id="resumen-docente")

```

Gráfico de llamadas a esta función:



4.102. Referencia del Namespace

`components.docente.utils.select_docente`

Funciones

- def `select_docente ()`

4.102.1. Documentación de las funciones

4.102.1.1. `select_docente()`

```
def components.docente.utils.select_docente ()
```

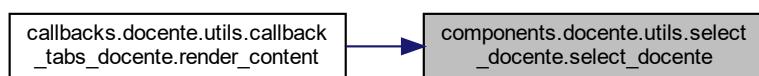
Crea el desplegable para seleccionar un docente.

Returns:
`html.Div`: Desplegable para seleccionar un docente

Definición en la línea 15 del archivo `select_docente.py`.

```
00015 def select_docente():
00016     """
00017     Crea el desplegable para seleccionar un docente.
00018
00019     Returns:
00020     html.Div: Desplegable para seleccionar un docente
00021     """
00022     return html.Div(
00023         [
00024             html.Div(
00025                 [
00026                     dcc.Dropdown(
00027                         options=[],
00028                         value=None,
00029                         id="docente-dropdown",
00030                         clearable=False,
00031                         placeholder="Selecciona un docente",
00032                     )
00033                 ],
00034                 className="select-docente",
00035             ),
00036         ]
00037     )
```

Gráfico de llamadas a esta función:



4.103. Referencia del Namespace components.docente.utils.tabs_docente

Funciones

- def tabs_docente ()

4.103.1. Documentación de las funciones

4.103.1.1. tabs_docente()

```
def components.docente.utils.tabs_docente ( )
```

Crea las pestañas del perfil "Docente"

Returns:
 html.Div: Pestañas del docente

Definición en la línea 15 del archivo tabs_docente.py.

```
00015 def tabs_docente():
00016     """
00017 Crea las pestañas del perfil "Docente"
00018
00019     Returns:
00020         html.Div: Pestañas del docente
00021     """
00022
00023     return html.Div(
00024         [
00025             dcc.Tabs(
00026                 id="tabs-docente",
00027                 value="rendimiento-academico-asignatura-tab",
00028                 children=[
00029                     dcc.Tab(
00030                         label="Rendimiento académico personal",
00031                         value="rendimiento-academico-asignatura-tab",
00032                     ),
00033                     dcc.Tab(
00034                         label="Rendimiento académico general",
00035                         value="rendimiento-academico-tab",
00036                     ),
00037                     dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
00038                 ],
00039                 className="tabs",
00040             ),
00041             html.Div(id="tabs-docente-content"),
00042             dcc.Store(id="selected-docente-store", storage_type="local"),
00043         ]
00044     )
```

4.104. Referencia del Namespace components.gestor

Namespaces

- namespace filters
- namespace graphs
- namespace utils

4.105. Referencia del Namespace components.gestor.filters

Namespaces

- namespace [filter_all_curso_academico_gestor](#)
- namespace [filter_curso_academico_gestor](#)
- namespace [filter_titulaciones_gestor](#)

4.106. Referencia del Namespace components.gestor.filters.filter_all_curso_academico_gestor

Funciones

- def [filter_all_curso_academico_gestor \(\)](#)

4.106.1. Documentación de las funciones

4.106.1.1. filter_all_curso_academico_gestor()

```
def components.gestor.filters.filter_all_curso_academico_gestor.filter_all_curso_academico_gestor ( )
```

Crea el filtro que muestra todos los cursos académicos disponibles para el perfil "Gestor" de la pestaña "Riesgo académico".

Returns:
 html.Div: Componente con el filtro de curso académico

Definición en la línea 16 del archivo [filter_all_curso_academico_gestor.py](#).

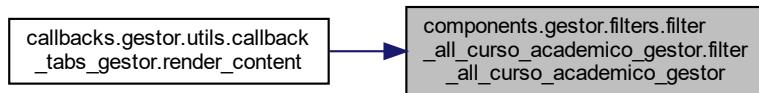
```
00016 def filter_all_curso_academico_gestor():
00017     """
00018     Crea el filtro que muestra todos los cursos académicos disponibles
00019     para el perfil "Gestor" de la pestaña "Riesgo académico".
00020
00021     Returns:
00022         html.Div: Componente con el filtro de curso académico
00023     """
00024
00025     return html.Div(
00026         [
00027             html.Label("Curso académico"),
00028             dcc.Dropdown(
00029                 id="curso-all-academico-gestor",
00030                 searchable=True,
00031                 multi=True,
00032                 clearable=True,
00033                 options=[],
00034                 value=None,
00035                 maxHeight=200,
00036                 placeholder="Seleccione una opción",
00037             ),
00038             html.Button(
00039                 "Seleccionar todo",
00040                 id="select-all-curso-academico-button",
00041                 className="button-select-all-filter",
00042                 n_clicks=0,
00043             ),
00044         ]
```

```

00044     dcc.Store(id="curso-all-academico-gestor-store", storage_type="local"),
00045 ]
00046 )

```

Gráfico de llamadas a esta función:



4.107. Referencia del Namespace components.gestor.filters.filter_curso_academico_gestor

Funciones

- def [filter_curso_academico_gestor\(\)](#)

4.107.1. Documentación de las funciones

4.107.1.1. filter_curso_academico_gestor()

```
def components.gestor.filters.filter_curso_academico_gestor.filter_curso_academico_gestor ( )
```

Crea el filtro de curso académico para el perfil "Gestor" de la pestaña "Indicadores académicos".

Returns:
html.Div: Componente con el filtro de curso académico

Definición en la línea 16 del archivo [filter_curso_academico_gestor.py](#).

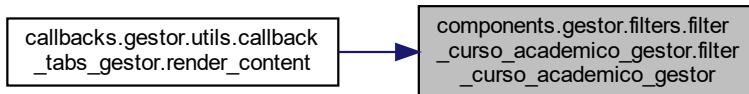
```

00016 def filter_curso_academico_gestor():
00017     """
00018     Crea el filtro de curso académico para el perfil "Gestor" de la pestaña
00019     "Indicadores académicos".
00020
00021     Returns:
00022         html.Div: Componente con el filtro de curso académico
00023     """
00024     return html.Div(
00025         [
00026             html.Label("Curso académico"),
00027             dcc.Dropdown(
00028                 id="curso-academico-gestor",
00029                 searchable=True,
00030                 multi=False,
00031                 clearable=False,
00032                 options=[],
00033                 value=None,
00034                 maxHeight=300,
00035                 placeholder="Selecciona una opción",

```

```
00036         ),
00037     ],
00038 )
```

Gráfico de llamadas a esta función:



4.108. Referencia del Namespace

components.gestor.filters.filter_titulaciones_gestor

Funciones

- def [filter_titulaciones_gestor\(\)](#)

4.108.1. Documentación de las funciones

4.108.1.1. filter_titulaciones_gestor()

```
def components.gestor.filters.filter_titulaciones_gestor().filter_titulaciones_gestor( )
```

Crea el filtro de titulaciones para el perfil "Gestor" de la pestaña "Indicadores académicos".

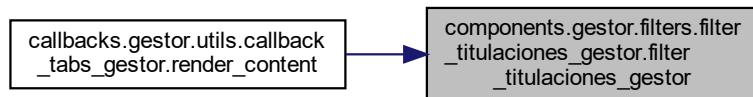
Returns:
`html.Div`: Componente con el filtro de titulaciones

Definición en la línea 16 del archivo [filter_titulaciones_gestor.py](#).

```
00016 def filter_titulaciones_gestor():
00017     """
00018     Crea el filtro de titulaciones para el perfil "Gestor" de la pestaña
00019     "Indicadores académicos".
00020
00021     Returns:
00022         html.Div: Componente con el filtro de titulaciones
00023     """
00024     return html.Div(
00025         [
00026             html.Br(),
00027             html.Label("Titulaciones"),
00028             dcc.Dropdown(
00029                 id="titulaciones-gestor",
00030                 searchable=True,
00031                 multi=True,
00032                 clearable=True,
00033                 options=[],
00034                 value=None,
00035                 maxHeight=200,
```

```
00036         placeholder="Seleccione una opción",
00037     ),
00038     html.Button(
00039         "Seleccionar todo",
00040         id="select-all-titulaciones-gestor",
00041         className="button-select-all-filter",
00042         n_clicks=0,
00043     ),
00044 ],
00045 )
```

Gráfico de llamadas a esta función:



4.109. Referencia del Namespace components.gestor.graphs

Namespaces

- namespace [graphs_indicadores_gestor](#)
- namespace [graphs_resultados_gestor](#)
- namespace [graphs_riesgo_abandono_gestor](#)

4.110. Referencia del Namespace components.gestor.graphs.graphs_indicadores_gestor

Funciones

- def [graphs_indicadores_gestor \(\)](#)

4.110.1. Documentación de las funciones

4.110.1.1. graphs_indicadores_gestor()

```
def components.gestor.graphs.graphs_indicadores_gestor.graphs_indicadores_gestor( )
```

Retorna los gráficos de la pestaña "Indicadores académicos" del perfil "Gestor". Incluye un modal y una tabla para cada gráfico.

Returns:
html.Div: Gráficos

Definición en la línea 21 del archivo graphs_indicadores_gestor.py.

```
00021 def graphs_indicadores_gestor():
00022     """
00023     Retorna los gráficos de la pestaña "Indicadores académicos" del perfil
00024     "Gestor". Incluye un modal y una tabla para cada gráfico.
00025
00026     Returns:
00027         html.Div: Gráficos
00028     """
00029
00030 graphs_info = [
00031     {
00032         "graph_id": "nuevo-ingreso-genero-gestor",
00033             "modal_id": "modal-nuevo-ingreso-genero",
00034             "table_container_id": "table-container-nuevo-ingreso-genero",
00035             "download_button_id": "btn-descargar-nuevo-ingreso-genero",
00036             "view_data_button_id": "btn-ver-datos-nuevo-ingreso-genero",
00037     },
00038     {
00039         "graph_id": "nuevo_ingreso_nacionalidad-gestor",
00040             "modal_id": "modal-nuevo-ingreso-nacionalidad",
00041             "table_container_id": "table-container-nuevo-ingreso-nacionalidad",
00042             "download_button_id": "btn-descargar-nuevo-ingreso-nacionalidad",
00043             "view_data_button_id": "btn-ver-datos-nuevo-ingreso-nacionalidad",
00044     },
00045     {
00046         "graph_id": "egresados-genero-gestor",
00047             "modal_id": "modal-egresados-genero",
00048             "table_container_id": "table-container-egresados-genero",
00049             "download_button_id": "btn-descargar-egresados-genero",
00050             "view_data_button_id": "btn-ver-datos-egresados-genero",
00051     },
00052     {
00053         "graph_id": "egresados-nacionalidad-gestor",
00054             "modal_id": "modal-egresados-nacionalidad",
00055             "table_container_id": "table-container-egresados-nacionalidad",
00056             "download_button_id": "btn-descargar-egresados-nacionalidad",
00057             "view_data_button_id": "btn-ver-datos-egresados-nacionalidad",
00058     },
00059 ]
00060
00061 graph_elements = [
00062     create_graph_with_table(
00063         graph_info["graph_id"],
00064         "graph-item-indicadores-gestor",
00065         config_mode_bar_buttons_gestor,
00066         {
00067             "modal_id": graph_info["modal_id"],
00068             "table_container_id": graph_info["table_container_id"],
00069             "download_button_id": graph_info["download_button_id"],
00070             "view_data_button_id": graph_info["view_data_button_id"],
00071         },
00072     )
00073     for graph_info in graphs_info
00074 ]
00075
00076 return html.Div(graph_elements, className="graphs-container-indicadores-gestor")
```

Gráfico de llamadas para esta función:

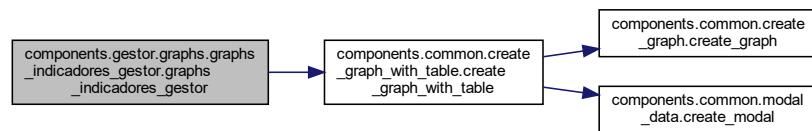
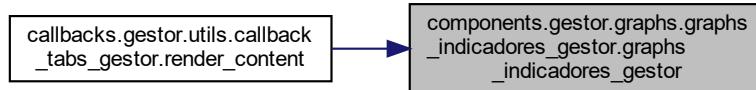


Gráfico de llamadas a esta función:



4.111. Referencia del Namespace **components.gestor.graphs.graphs_resultados_gestor**

Funciones

- def [graphs_resultados_gestor\(\)](#)

4.111.1. Documentación de las funciones

4.111.1.1. [graphs_resultados_gestor\(\)](#)

```
def components.gestor.graphs.graphs_resultados_gestor.graphs_resultados_gestor ( )
```

Retorna los gráficos de la pestaña "Resultados académicos" del perfil "Gestor". Incluye un modal y una tabla para cada gráfico.

Returns:
html.Div: Gráficos

Definición en la línea 19 del archivo [graphs_resultados_gestor.py](#).

```
00019 def graphs_resultados_gestor():
00020     """
00021     Retorna los gráficos de la pestaña "Resultados académicos" del perfil
00022     "Gestor". Incluye un modal y una tabla para cada gráfico.
00023
00024     Returns:
00025         html.Div: Gráficos
00026     """
00027 graphs_info = [
00028     {
00029         "graph_id": "duracion-estudios-nota-media-gestor",
00030         "modal_id": "modal-duracion-estudios",
00031         "table_container_id": "table-container-duracion-estudios",
00032         "download_button_id": "btn-descargar-csv-duracion-estudios",
00033         "view_data_button_id": "btn-ver-datos-duracion-estudios",
00034     },
00035     {
00036         "graph_id": "nota-acceso-titulacion",
00037         "modal_id": "modal-nota-acceso",
00038         "table_container_id": "table-container-nota-acceso",
00039         "download_button_id": "btn-descargar-csv-nota-acceso",
00040         "view_data_button_id": "btn-ver-datos-nota-acceso",
00041     },
00042 ]
00043
```

```

00044     graph_elements = [
00045         create_graph_with_table(
00046             graph_info["graph_id"],
00047             "graph-item=resultado-gestor",
00048             config_mode_bar_buttons_gestor,
00049             {
00050                 "modal_id": graph_info["modal_id"],
00051                 "table_container_id": graph_info["table_container_id"],
00052                 "download_button_id": graph_info["download_button_id"],
00053                 "view_data_button_id": graph_info["view_data_button_id"],
00054             },
00055         ),
00056     for graph_info in graphs_info
00057 ]
00058
00059     return html.Div(graph_elements, className="graphs-container-resultado-gestor")

```

Gráfico de llamadas para esta función:

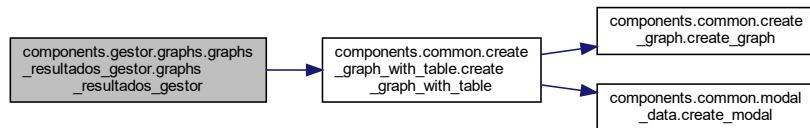
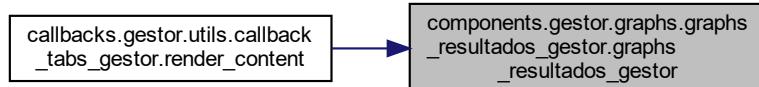


Gráfico de llamadas a esta función:



4.112. Referencia del Namespace components.gestor.graphs.graphs_riesgo_abandono_gestor

Funciones

- def `graphs_riesgo_abandono_gestor ()`

4.112.1. Documentación de las funciones

4.112.1.1. graphs_riesgo_abandono_gestor()

```
def components.gestor.graphs.graphs_riesgo_abandono_gestor( )
```

Retorna los gráficos de la pestaña "Riesgo de abandono" del perfil "Gestor". Incluye un modal y una tabla para cada gráfico.

Returns:
 html.Div: Gráficos

Definición en la línea 19 del archivo [graphs_riesgo_abandono_gestor.py](#).

```
00019 def graphs_riesgo_abandono_gestor():
00020     """
00021     Retorna los gráficos de la pestaña "Riesgo de abandono" del perfil
00022     "Gestor". Incluye un modal y una tabla para cada gráfico.
00023
00024     Returns:
00025         html.Div: Gráficos
00026     """
00027 graphs_info = [
00028     {
00029         "graph_id": "tasa-abandono-gestor",
00030         "modal_id": "modal-tasa-abandono",
00031         "table_container_id": "table-container-tasa-abandono",
00032         "download_button_id": "btn-descargar-tasa-abandono",
00033         "view_data_button_id": "btn-ver-datos-tasa-abandono",
00034     },
00035     {
00036         "graph_id": "tasa-graduacion-gestor",
00037         "modal_id": "modal-tasa-graduacion",
00038         "table_container_id": "table-container-tasa-graduacion",
00039         "download_button_id": "btn-descargar-tasa-graduacion",
00040         "view_data_button_id": "btn-ver-datos-tasa-graduacion",
00041     },
00042 ]
00043
00044 graph_elements = [
00045     create_graph_with_table(
00046         graph_info["graph_id"],
00047         "graph-item-riesgo-abandono-gestor",
00048         config_mode_bar_buttons_gestor,
00049         {
00050             "modal_id": graph_info["modal_id"],
00051             "table_container_id": graph_info["table_container_id"],
00052             "download_button_id": graph_info["download_button_id"],
00053             "view_data_button_id": graph_info["view_data_button_id"],
00054         },
00055     )
00056     for graph_info in graphs_info
00057 ]
00058
00059 return html.Div(graph_elements, className="graphs-container-riesgo-abandono-gestor")
```

Gráfico de llamadas para esta función:

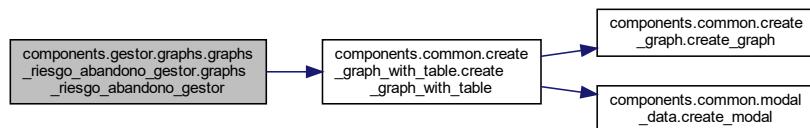
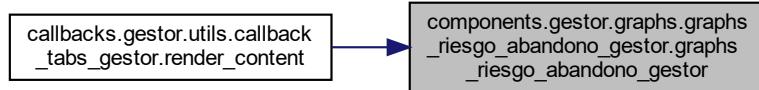


Gráfico de llamadas a esta función:



4.113. Referencia del Namespace `components.gestor.utils`

Namespaces

- namespace `recomendador_gestor`
- namespace `resumen_gestor`
- namespace `select_gestor`
- namespace `tabs_gestor`

4.114. Referencia del Namespace `components.gestor.utils.recomendador_gestor`

Funciones

- def `recomendador_gestor()`

4.114.1. Documentación de las funciones

4.114.1.1. `recomendador_gestor()`

```
def components.gestor.utils.recomendador_gestor.recomendador_gestor ( )
```

Crea un contenedor con información y recomendaciones sobre la deserción universitaria para los gestores de la sección "Recomendaciones" de la pestaña "Gestor".

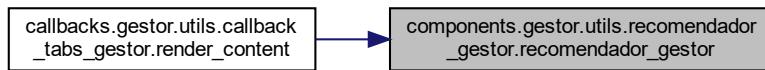
Returns:

html.Div: Contenedor con información y recomendaciones sobre la deserción universitaria

Definición en la línea 14 del archivo [recomendador_gestor.py](#).

```
00014 def recomendador_gestor():
00015     """
00016     Crea un contenedor con información y recomendaciones sobre la deserción universitaria
00017     para los gestores de la sección "Recomendaciones" de la pestaña "Gestor".
00018
00019     Returns:
00020         html.Div: Contenedor con información y recomendaciones sobre la deserción universitaria
00021     """
00022     return html.Div([
00023         html.H1("Deserción universitaria", className='titulo-recomendador-gestor'),
00024         html.Img(src='assets/images/abandono-gestores.jpg', className='imagen-recomendador-gestor'),
00025         html.P("La deserción universitaria es un desafío significativo para las instituciones de
00026             educación superior. Los gestores tienen la responsabilidad de desarrollar e implementar políticas
00027             efectivas para reducirla.", className='p-recomendador-gestor'),
00028         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-gestor'),
00029         html.P("La deserción universitaria se refiere al abandono de los estudios por parte de los
00030             estudiantes antes de completar su formación. Este fenómeno tiene implicaciones negativas para la
00031             universidad y para la sociedad.", className='p-recomendador-gestor'),
00032         html.Br(),
00033         html.H2("Estrategias para prevenir la deserción universitaria",
00034             className='sb-recomendador-gestor'),
00035         html.Ul([
00036             html.Li("Desarrollar programas de becas y ayudas financieras para apoyar a los estudiantes
00037                 con dificultades económicas.", className='li-recomendador-gestor'),
00038             html.Li("Implementar sistemas de tutoría y mentoría para ofrecer apoyo académico y
00039                 emocional a los estudiantes.", className='li-recomendador-gestor'),
00040             html.Li("Establecer programas de orientación y adaptación para nuevos estudiantes,
00041                 facilitando su integración en la vida universitaria.", className='li-recomendador-gestor'),
00042             html.Li("Promover la formación continua del personal docente en metodologías pedagógicas
00043                 innovadoras.", className='li-recomendador-gestor'),
00044             html.Li("Utilizar sistemas de seguimiento y análisis de datos para identificar a
00045                 estudiantes en riesgo y actuar de manera preventiva.", className='li-recomendador-gestor'),
00046             html.Li("Fomentar un clima institucional inclusivo y de apoyo que motive a los estudiantes
00047                 a continuar con sus estudios.", className='li-recomendador-gestor'),
00048         ], className='ul-recomendador-gestor'),
00049     ])
00050 
```

Gráfico de llamadas a esta función:



4.115. Referencia del Namespace **components.gestor.utils.resumen_gestor**

Funciones

- `def resumen_gestor ()`

4.115.1. Documentación de las funciones

4.115.1.1. resumen_gestor()

```
def components.gestor.utils.resumen_gestor.resumen_gestor ( )
```

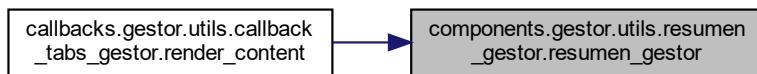
Crea un contenedor para mostrar un resumen de los datos del gestor seleccionado.

Returns:
 html.Div: Contenedor para mostrar el resumen

Definición en la línea 15 del archivo [resumen_gestor.py](#).

```
00015 def resumen_gestor():
00016     """
00017 Crea un contenedor para mostrar un resumen de los datos del gestor seleccionado.
00018
00019 Returns:
00020 html.Div: Contenedor para mostrar el resumen
00021 """
00022
00023 return html.Div([], id="resumen-gestor")
```

Gráfico de llamadas a esta función:



4.116. Referencia del Namespace `components.gestor.utils.select_gestor`

Funciones

- `def select_gestor ()`

4.116.1. Documentación de las funciones

4.116.1.1. select_gestor()

```
def components.gestor.utils.select_gestor.select_gestor ( )
```

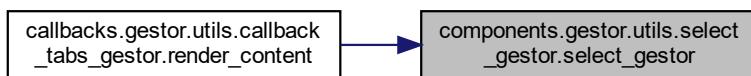
Crea un dropdown para seleccionar un gestor.

Returns:
 html.Div: Dropdown para seleccionar un gestor

Definición en la línea 15 del archivo [select_gestor.py](#).

```
00015 def select_gestor():
00016     """
00017 Crea un dropdown para seleccionar un gestor.
00018
00019 Returns:
00020 html.Div: Dropdown para seleccionar un gestor
00021 """
00022
00023 return html.Div(
00024     [
00025         html.Div(
00026             [
00027                 dcc.Dropdown(
00028                     options=[],
00029                     value=None,
00030                     id="gestor-dropdown",
00031                     clearable=False,
00032                     placeholder="Seleccione un gestor",
00033                 )
00034             ],
00035             className="select-gestor",
00036         ),
00037     ]
00038 )
```

Gráfico de llamadas a esta función:



4.117. Referencia del Namespace components.gestor.utils.tabs_gestor

Funciones

- [def tabs_gestor \(\)](#)

4.117.1. Documentación de las funciones

4.117.1.1. tabs_gestor()

```
def components.gestor.utils.tabs_gestor.tabs_gestor ( )
```

Crea las pestañas de la sección "Gestor". Cada pestaña contiene un conjunto de gráficos y tablas con los datos de los alumnos del perfil "Gestor".

Returns:
html.Div: Pestañas de la sección "Gestor"

Definición en la línea 15 del archivo `tabs_gestor.py`.

```

00015 def tabs_gestor():
00016     """
00017     Crea las pestañas de la sección "Gestor". Cada pestaña contiene un conjunto de gráficos
00018     y tablas con los datos de los alumnos del perfil "Gestor".
00019
00020     Returns:
00021         html.Div: Pestañas de la sección "Gestor"
00022     """
00023
00024     return html.Div(
00025         [
00026             dcc.Tabs(
00027                 id="tabs-gestor",
00028                 value="indicadores-academicos-tab",
00029                 children=[
00030                     dcc.Tab(
00031                         label="Indicadores académicos",
00032                         value="indicadores-academicos-tab",
00033                     ),
00034                     dcc.Tab(
00035                         label="Resultados académicos", value="resultados-academicos-tab"
00036                     ),
00037                     dcc.Tab(label="Riesgo de abandono", value="riesgo-abandono-tab"),
00038                     dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
00039                 ],
00040                 className="tabs",
00041             ),
00042             html.Div(id="tabs-gestor-content"),
00043             dcc.Store(id="selected-gestor-store", storage_type="local"),
00044             dcc.Location(id="url", refresh=False),
00045         ]
00046     )

```

4.118. Referencia del Namespace data

Namespaces

- namespace `db_connector`
- namespace `generate_synthetic_data`
- namespace `queries`
- namespace `queries_dictionary`

4.119. Referencia del Namespace data.db_connector

Clases

- class `DatabaseConnector`

Variables

- `config` = json.load(f)
- `db`

4.119.1. Documentación de las variables

4.119.1.1. config

```
data.db_connector.config = json.load(f)
```

Definición en la línea 65 del archivo `db_connector.py`.

4.119.1.2. db

```
data.db_connector.db
```

Valor inicial:

```
00001 = DatabaseConnector(  
00002     dbname=config["dbname"],  
00003     user=config["user"],  
00004     password=config["password"],  
00005     host=config["host"],  
00006     port=config["port"],  
00007 )
```

Definición en la línea 68 del archivo `db_connector.py`.

4.120. Referencia del Namespace `data.generate_synthetic_data`

Funciones

- def `generate_asignaturas(educacion, min_asignaturas=38)`
- def `generate_unique_cod_asignaturas(asignaturas_dict)`
- def `generate_alumnos(n)`
- def `generate_curso_aca(start_year, end_year)`
- def `generate_matricula(alumnos_df)`
- def `generate_asignaturas_matriculadas(matricula_df, cod_asignaturas)`
- def `generate_lineas_actas(asignaturas_matriculadas_df)`
- def `generate_docentes(n, universidades, cod_asignaturas, asignaturas_dict)`
- def `generate_ebau_prueba(n, alumnos_ids, universidades)`
- def `generate_egresados(alumnos_df, matricula_df, lineas_actas_df)`
- def `generate_gestores(n, universidades)`

Variables

- `fake = Faker('es_ES')`
- dictionary `educacion`
- dictionary `universidades_dict`
- def `asignaturas_dict = generate_asignaturas(educacion)`
- def `cod_asignaturas_dict = generate_unique_cod_asignaturas(asignaturas_dict)`
- dictionary `cod_plan_dict = {titulacion: fake.bothify(text='PLAN###??') for titulacion in asignaturas_dict.keys()}`
- def `cursos_academicos = generate_curso_aca(2017, 2024)`
- int `num_alumnos = 5000`
- int `num_docentes = 100`
- int `num_gestores = 50`
- def `alumnos_df = generate_alumnos(num_alumnos)`
- def `matricula_df = generate_matricula(alumnos_df)`
- def `asignaturas_matriculadas_df = generate_asignaturas_matriculadas(matricula_df, cod_asignaturas_dict)`
- def `lineas_actas_df = generate_lineas_actas(asignaturas_matriculadas_df)`
- def `docentes_df = generate_docentes(num_docentes, universidades_dict, cod_asignaturas_dict, asignaturas_dict)`
- def `ebau_prueba_df = generate_ebau_prueba(num_alumnos, alumnos_df['id'].tolist(), universidades_dict)`
- def `egresados_df = generate_egresados(alumnos_df, matricula_df, lineas_actas_df)`
- def `gestores_df = generate_gestores(num_gestores, universidades_dict)`
- `index`

4.120.1. Documentación de las funciones

4.120.1.1. generate_alumnos()

```
def data.generate_synthetic_data.generate_alumnos (
    n )
```

Genera datos sintéticos para la tabla 'alumnos'.

Args:
`n` (int): Número de alumnos a generar.

Returns:
`pd.DataFrame`: DataFrame con los datos de los alumnos.

Definición en la línea 129 del archivo [generate_synthetic_data.py](#).

```
00129 def generate_alumnos(n):
00130     """
00131     Genera datos sintéticos para la tabla 'alumnos'.
00132
00133     Args:
00134         n (int): Número de alumnos a generar.
00135
00136     Returns:
00137         pd.DataFrame: DataFrame con los datos de los alumnos.
00138     """
00139     ids = [fake.unique.bothify(text='????#####') for _ in range(n)]
00140     anios_nac = np.random.randint(1980, 2000, n)
00141     universidades = random.choices(list(universidades_dict.keys()), k=n)
00142     cod_universidades = [universidades_dict[uni] for uni in universidades]
00143     abandona = random.choices(['si', 'no'], k=n)
00144     data = {
00145         'id': ids,
00146         'anio_nac': anios_nac,
00147         'universidad': universidades,
00148         'cod_universidad': cod_universidades,
00149         'abandona': abandona
00150     }
00151     return pd.DataFrame(data)
00152
00153
```

4.120.1.2. generate_asignaturas()

```
def data.generate_synthetic_data.generate_asignaturas (
    educación,
    min_asignaturas = 38 )
```

Genera un diccionario con asignaturas aleatorias para cada titulación.

Args:
`educación` (dict): Diccionario con ramas, titulaciones y palabras clave.
`min_asignaturas` (int): Número mínimo de asignaturas por titulación.

Returns:
`dict`: Diccionario con asignaturas para cada titulación.

Definición en la línea 84 del archivo `generate_synthetic_data.py`.

```

00084 def generate_asignaturas(educacion, min_asignaturas=38):
00085     """
00086     Genera un diccionario con asignaturas aleatorias para cada titulación.
00087
00088     Args:
00089         educacion (dict): Diccionario con ramas, titulaciones y palabras clave.
00090         min_asignaturas (int): Número mínimo de asignaturas por titulación.
00091
00092     Returns:
00093         dict: Diccionario con asignaturas para cada titulación.
00094     """
00095     asignaturas_dict = {}
00096     for rama, titulaciones in educacion.items():
00097         for titulacion, palabras in titulaciones.items():
00098             num_asignaturas = min(min_asignaturas, len(palabras))
00099             asignaturas = random.sample(palabras, num_asignaturas)
00100             asignaturas_dict[titulacion] = asignaturas
00101
00102     return asignaturas_dict
00102

```

4.120.1.3. `generate_asignaturas_matriculadas()`

```

def data.generate_synthetic_data.generate_asignaturas_matriculadas (
    matricula_df,
    cod_asignaturas )

```

Genera datos sintéticos para la tabla 'asignaturas_matriculadas'.

Args:

```

matricula_df (pd.DataFrame): DataFrame con los datos de matrícula.
cod_asignaturas (dict): Diccionario con códigos únicos para cada asignatura.

Returns:
pd.DataFrame: DataFrame con los datos de asignaturas matriculadas.

```

Definición en la línea 231 del archivo `generate_synthetic_data.py`.

```

00231 def generate_asignaturas_matriculadas(matricula_df, cod_asignaturas):
00232     """
00233     Genera datos sintéticos para la tabla 'asignaturas_matriculadas'.
00234
00235     Args:
00236         matricula_df (pd.DataFrame): DataFrame con los datos de matrícula.
00237         cod_asignaturas (dict): Diccionario con códigos únicos para cada asignatura.
00238
00239     Returns:
00240         pd.DataFrame: DataFrame con los datos de asignaturas matriculadas.
00241     """
00242     asignaturas_data = []
00243     for i, row in tqdm(matricula_df.iterrows(), total=len(matricula_df), desc="asignaturas_matriculadas"):
00244         num_asignaturas = min(random.randint(1, 20), len(asignaturas_dict[row['titulacion']]))
00245         asignaturas = random.sample(asignaturas_dict[row['titulacion']], num_asignaturas)
00246         for asignatura in asignaturas:
00247             cod_asignatura = cod_asignaturas[asignatura]
00248             asignaturas_data.append({
00249                 'indice': len(asignaturas_data) + 1,
00250                 'asignatura': asignatura,
00251                 'cod_asignatura': cod_asignatura,
00252                 'cod_plan': row['cod_plan'],
00253                 'curso_aca': row['curso_aca'],
00254                 'id': row['id'],
00255                 'cod_tipologia': fake.bothify(text='T##'),
00256                 'plan': row['titulacion'],
00257                 'tipologia': fake.word(),
00258                 'universidad': row['universidad'],
00259                 'cod_universidad': row['cod_universidad']
00260             })
00261     return pd.DataFrame(asignaturas_data)
00262
00263

```

4.120.1.4. generate_curso_aca()

```
def data.generate_synthetic_data.generate_curso_aca (
    start_year,
    end_year )
```

Genera una lista de cursos académicos con el formato 'YYYY/YYYY+1'.

Args:
 start_year (int): Año de inicio.
 end_year (int): Año de fin.

Returns:
 list: Lista de cursos académicos.

Definición en la línea 154 del archivo [generate_synthetic_data.py](#).

```
00154 def generate_curso_aca(start_year, end_year):
00155     """
00156     Genera una lista de cursos académicos con el formato 'YYYY/YYYY+1'.
00157
00158     Args:
00159         start_year (int): Año de inicio.
00160         end_year (int): Año de fin.
00161
00162     Returns:
00163         list: Lista de cursos académicos.
00164     """
00165     return [f"{year}/{year+1}" for year in range(start_year, end_year)]
00166
00167 # Generar una lista de cursos académicos
```

4.120.1.5. generate_docentes()

```
def data.generate_synthetic_data.generate_docentes (
    n,
    universidades,
    cod_asignaturas,
    asignaturas_dict )
```

Genera datos sintéticos para la tabla 'docentes'.

Args:
 n (int): Número de docentes a generar.
 universidades (dict): Diccionario con universidades y sus códigos.
 cod_asignaturas (dict): Diccionario con códigos únicos para cada asignatura.
 asignaturas_dict (dict): Diccionario con asignaturas para cada titulación.

Returns:
 pd.DataFrame: DataFrame con los datos de los docentes.

Definición en la línea 303 del archivo [generate_synthetic_data.py](#).

```
00303 def generate_docentes(n, universidades, cod_asignaturas, asignaturas_dict):
00304     """
00305     Genera datos sintéticos para la tabla 'docentes'.
00306
00307     Args:
00308         n (int): Número de docentes a generar.
00309         universidades (dict): Diccionario con universidades y sus códigos.
00310         cod_asignaturas (dict): Diccionario con códigos únicos para cada asignatura.
00311         asignaturas_dict (dict): Diccionario con asignaturas para cada titulación.
00312
00313     Returns:
00314         pd.DataFrame: DataFrame con los datos de los docentes.
00315     """
```

```

00316 docentes_data = []
00317
00318 for i in tqdm(range(n), desc="docentes"):
00319     id_docente = fake.bothify(text='D#####')
00320     universidad, cod_universidad = random.choice(list(universidades.items()))
00321
00322     num_asignaturas = random.randint(1, 5)
00323     asignaturas_seleccionadas = random.sample(list(cod_asignaturas.items()), num_asignaturas)
00324
00325     for asignatura, cod_asignatura in asignaturas_seleccionadas:
00326         titulacion = next(titulacion for titulacion, asignaturas in asignaturas_dict.items() if
00327             asignatura in asignaturas)
00328         cod_plan = cod_plan_dict[titulacion]
00329         docentes_data.append({
00330             'indice': len(docentes_data) + 1,
00331             'id_docente': id_docente,
00332             'cod_universidad': cod_universidad,
00333             'universidad': universidad,
00334             'titulacion': titulacion,
00335             'cod_plan': cod_plan,
00336             'cod_asignatura': cod_asignatura,
00337             'asignatura': asignatura,
00338             'curso_aca': random.choice(cursos_academicos)
00339         })
00340
00341 return pd.DataFrame(docentes_data)
00342

```

4.120.1.6. `generate_ebau_prueba()`

```

def data.generate_synthetic_data.generate_ebau_prueba (
    n,
    alumnos_ids,
    universidades )

```

Genera datos sintéticos para la tabla 'ebau_prueba'.

Args:

n (int): Número de registros a generar.
alumnos_ids (list): Lista de IDs de alumnos.
universidades (dict): Diccionario con universidades y sus códigos.

Returns:

`pd.DataFrame`: DataFrame con los datos de las pruebas EBAU.

Definición en la línea 343 del archivo `generate_synthetic_data.py`.

```

00343 def generate_ebau_prueba(n, alumnos_ids, universidades):
00344 """
00345 Genera datos sintéticos para la tabla 'ebau_prueba'.
00346
00347     Args:
00348         n (int): Número de registros a generar.
00349         alumnos_ids (list): Lista de IDs de alumnos.
00350         universidades (dict): Diccionario con universidades y sus códigos.
00351
00352     Returns:
00353         pd.DataFrame: DataFrame con los datos de las pruebas EBAU.
00354 """
00355 assert n == len(alumnos_ids), "El número de registros debe ser igual al número de alumnos"
00356 data = {
00357     'indice': range(1, n + 1),
00358     'conv': [fake.bothify(text='C##') for _ in range(n)],
00359     'curso_aca': np.random.choice(cursos_academicos, n),
00360     'especialidad': [fake.word() for _ in range(n)],
00361     'id': alumnos_ids,
00362     'nota_bach': np.round(np.random.uniform(5, 10, n), 2),
00363     'nota_def': np.round(np.random.uniform(5, 10, n), 2),
00364     'nota_prue': np.round(np.random.uniform(5, 10, n), 2),
00365     'universidad': np.random.choice(list(universidades.keys()), n),
00366 }
00367 data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]
00368
00369 return pd.DataFrame(data)
00370
00371

```

4.120.1.7. generate_egresados()

```
def data.generate_synthetic_data.generate_egresados (
    alumnos_df,
    matricula_df,
    lineas_actas_df )
```

Genera datos sintéticos para la tabla 'egresados'.

Args:

alumnos_df (pd.DataFrame): DataFrame con los datos de los alumnos.
matricula_df (pd.DataFrame): DataFrame con los datos de matrícula.
lineas_actas_df (pd.DataFrame): DataFrame con los datos de actas.

Returns:

pd.DataFrame: DataFrame con los datos de los egresados.

Definición en la línea 372 del archivo [generate_synthetic_data.py](#).

```
00372 def generate_egresados(alumnos_df, matricula_df, lineas_actas_df):
00373     """
00374     Genera datos sintéticos para la tabla 'egresados'.
00375
00376     Args:
00377         alumnos_df (pd.DataFrame): DataFrame con los datos de los alumnos.
00378         matricula_df (pd.DataFrame): DataFrame con los datos de matrícula.
00379         lineas_actas_df (pd.DataFrame): DataFrame con los datos de actas.
00380
00381     Returns:
00382     pd.DataFrame: DataFrame con los datos de los egresados.
00383     """
00384     aprobados = lineas_actas_df[lineas_actas_df['calif_numerica'] >= 5].groupby('id').size()
00385     egresados_ids = aprobados[aprobados >= 38].index.tolist()
00386     alumnos_no_abandona = alumnos_df[alumnos_df['abandona'] == 'no']['id'].tolist()
00387     egresados_ids = [id for id in egresados_ids if id in alumnos_no_abandona]
00388
00389     egresados_data = []
00390     for i, id_alumno in tqdm(enumerate(egresados_ids), total=len(egresados_ids), desc="egresados"):
00391         universidad, cod_universidad = alumnos_df.loc[alumnos_df['id'] == id_alumno, ['universidad',
00392             'cod_universidad']].values[0]
00393         cod_plan = matricula_df.loc[matricula_df['id'] == id_alumno, 'cod_plan'].values[0]
00394         ultimo_curso_aca = matricula_df.loc[matricula_df['id'] == id_alumno, 'curso_aca'].max()
00395         egresados_data.append({
00396             'indice': i + 1,
00397             'cod_plan': cod_plan,
00398             'curso_aca': ultimo_curso_aca,
00399             'id': id_alumno,
00400             'nota_media': round(random.uniform(5, 10), 2),
00401             'plan': fake.word(),
00402             'universidad': universidad,
00403             'cod_universidad': cod_universidad
00404         })
00405     return pd.DataFrame(egresados_data)
00406
```

4.120.1.8. generate_gestores()

```
def data.generate_synthetic_data.generate_gestores (
    n,
    universidades )
```

Genera datos sintéticos para la tabla 'gestores'.

Args:

n (int): Número de gestores a generar.
universidades (dict): Diccionario con universidades y sus códigos.

Returns:

pd.DataFrame: DataFrame con los datos de los gestores.

Definición en la línea 407 del archivo `generate_synthetic_data.py`.

```
00407 def generate_gestores(n, universidades):
00408     """
00409     Genera datos sintéticos para la tabla 'gestores'.
00410
00411     Args:
00412         n (int): Número de gestores a generar.
00413         universidades (dict): Diccionario con universidades y sus códigos.
00414
00415     Returns:
00416         pd.DataFrame: DataFrame con los datos de los gestores.
00417     """
00418 data = {
00419     'gestor_id': range(1, n + 1),
00420     'universidad': np.random.choice(list(universidades.keys()), n)
00421 }
00422 data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]
00423 return pd.DataFrame(data)
00424
00425
00426 print("Generando datos....")
```

4.120.1.9. `generate_lineas_actas()`

```
def data.generate_synthetic_data.generate_lineas_actas (
    asignaturas_matriculadas_df )
```

Genera datos sintéticos para la tabla 'lineas_actas'.

Args:

`asignaturas_matriculadas_df` (pd.DataFrame): DataFrame con los datos de asignaturas matriculadas.

Returns:

`pd.DataFrame`: DataFrame con los datos de actas.

Definición en la línea 264 del archivo `generate_synthetic_data.py`.

```
00264 def generate_lineas_actas(asignaturas_matriculadas_df):
00265     """
00266     Genera datos sintéticos para la tabla 'lineas_actas'.
00267
00268     Args:
00269         asignaturas_matriculadas_df (pd.DataFrame): DataFrame con los datos de asignaturas
00270             matriculadas.
00271
00272     Returns:
00273         pd.DataFrame: DataFrame con los datos de actas.
00274     """
00275 actas_data = []
00276 for i, row in tqdm(asignaturas_matriculadas_df.iterrows(), total=len(asignaturas_matriculadas_df),
desc="lineas_actas"):
00277     calif_numerica = round(random.uniform(0, 10), 2)
00278     if calif_numerica < 5:
00279         calif = 'Suspensos'
00280     elif calif_numerica < 7:
00281         calif = 'Aprobado'
00282     elif calif_numerica < 9:
00283         calif = 'Notable'
00284     else:
00285         calif = 'Sobresaliente'
00286
00287     actas_data.append({
00288         'indice': len(actas_data) + 1,
00289         'asignatura': row['asignatura'],
00290         'calif_numerica': calif_numerica,
00291         'calif': calif,
00292         'cod_asig': row['cod_asignatura'],
00293         'cod_plan': row['cod_plan'],
00294         'conv': fake.bothify(text='C##'),
00295         'curso_aca': row['curso_aca'],
00296         'id': row['id'],
00297         'plan': row['plan'],
00298         'universidad': row['universidad'],
00299         'cod_universidad': row['cod_universidad']
00300     })
00301
00302 return pd.DataFrame(actas_data)
```

4.120.1.10. generate_matricula()

```
def data.generate_synthetic_data.generate_matricula (
    alumnos_df )

Genera datos sintéticos para la tabla 'matricula'.
```

Args:

```
    alumnos_df (pd.DataFrame): DataFrame con los datos de los alumnos.
```

Returns:

```
    pd.DataFrame: DataFrame con los datos de matrícula.
```

Definición en la línea 171 del archivo [generate_synthetic_data.py](#).

```
00171 def generate_matricula(alumnos_df):
00172     """
00173     Genera datos sintéticos para la tabla 'matricula'.
00174
00175     Args:
00176         alumnos_df (pd.DataFrame): DataFrame con los datos de los alumnos.
00177
00178     Returns:
00179         pd.DataFrame: DataFrame con los datos de matrícula.
00180     """
00181 n = len(alumnos_df)
00182 num_matriculas = np.random.randint(1, 6, n)
00183 matricula_data = []
00184 for i, row in tqdm(alumnos_df.iterrows(), total=n, desc="matrícula"):
00185     titulacion = random.choice(list(asignaturas_dict.keys()))
00186     # Encontrar la rama correspondiente a la titulación
00187     rama = next(rama for rama, titulaciones in educación.items() if titulacion in titulaciones)
00188
00189     common_data = {
00190         'municipio': fake.city(),
00191         'nacionalidad': fake.country(),
00192         'provincia': fake.state(),
00193         'rama': rama,
00194         'sexo': random.choice(['Masculino', 'Femenino']),
00195         'titulacion': titulacion,
00196         'nombre_plan_propio': fake.word(),
00197         'universidad': row['universidad'],
00198         'cod_universidad': row['cod_universidad'],
00199         'cod_tipo_matricula': fake.bothify(text='T##'),
00200         'tipo_matricula': fake.word(),
00201         'cod_plan': cod_plan_dict[titulacion],
00202     }
00203     cursos_aca_usados = set()
00204     for j in range(num_matriculas[i]):
00205         curso_aca = random.choice([ca for ca in cursos_academicos if ca not in cursos_aca_usados])
00206         cursos_aca_usados.add(curso_aca)
00207         nuevo_ingreso = 'si' if j == 0 else 'no'
00208         matricula_data.append({
00209             'indice': len(matricula_data) + 1,
00210             'ambito_isced': fake.bothify(text='IS##'),
00211             'cod_plan': common_data['cod_plan'],
00212             'cod_mec': fake.bothify(text='MEC##'),
00213             'curso_aca': curso_aca,
00214             'municipio': common_data['municipio'],
00215             'nacionalidad': common_data['nacionalidad'],
00216             'nuevo_ingreso': nuevo_ingreso,
00217             'provincia': common_data['provincia'],
00218             'rama': common_data['rama'],
00219             'sexo': common_data['sexo'],
00220             'titulacion': common_data['titulacion'],
00221             'nombre_plan_propio': common_data['nombre_plan_propio'],
00222             'universidad': common_data['universidad'],
00223             'cod_universidad': common_data['cod_universidad'],
00224             'cod_tipo_matricula': common_data['cod_tipo_matricula'],
00225             'tipo_matricula': common_data['tipo_matricula'],
00226             'id': row['id']
00227         })
00228     return pd.DataFrame(matricula_data)
00229
00230
```

4.120.1.11. `generate_unique_cod_asignaturas()`

```
def data.generate_synthetic_data.generate_unique_cod_asignaturas (
    asignaturas_dict )
```

Genera un diccionario con códigos únicos para cada asignatura.

Args:

asignaturas_dict (dict): Diccionario con asignaturas para cada titulación.

Returns:

dict: Diccionario con códigos únicos para cada asignatura.

Definición en la línea 107 del archivo [generate_synthetic_data.py](#).

```
00107 def generate_unique_cod_asignaturas(asignaturas_dict):
00108     """
00109     Genera un diccionario con códigos únicos para cada asignatura.
00110
00111     Args:
00112         asignaturas_dict (dict): Diccionario con asignaturas para cada titulación.
00113
00114     Returns:
00115         dict: Diccionario con códigos únicos para cada asignatura.
00116     """
00117     cod_asignaturas = {}
00118     for titulacion, asignaturas in asignaturas_dict.items():
00119         for asignatura in asignaturas:
00120             cod_asignatura = asignatura[:4].upper() + fake.bothify(text='#####')
00121             cod_asignaturas[asignatura] = cod_asignatura
00122     return cod_asignaturas
00123
00124
```

4.120.2. Documentación de las variables

4.120.2.1. `alumnos_df`

```
def data.generate_synthetic_data.alumnos_df = generate_alumnos(num_alumnos)
```

Definición en la línea 436 del archivo [generate_synthetic_data.py](#).

4.120.2.2. `asignaturas_dict`

```
def data.generate_synthetic_data.asignaturas_dict = generate_asignaturas(educacion)
```

Definición en la línea 103 del archivo [generate_synthetic_data.py](#).

4.120.2.3. `asignaturas_matriculadas_df`

```
def data.generate_synthetic_data.asignaturas_matriculadas_df = generate_asignaturas_matriculadas(matricula_df,
cod_asignaturas_dict)
```

Definición en la línea 438 del archivo [generate_synthetic_data.py](#).

4.120.2.4. cod_asignaturas_dict

```
def data.generate_synthetic_data.cod_asignaturas_dict = generate_unique_cod_asignaturas(asignaturas_dict)
```

Definición en la línea 125 del archivo [generate_synthetic_data.py](#).

4.120.2.5. cod_plan_dict

```
dictionary data.generate_synthetic_data.cod_plan_dict = {titulacion: fake.bothify(text='PLAN##??')}
```

```
for titulacion in asignaturas_dict.keys() }
```

Definición en la línea 126 del archivo [generate_synthetic_data.py](#).

4.120.2.6. cursos_academicos

```
def data.generate_synthetic_data.cursos_academicos = generate_curso_aca(2017, 2024)
```

Definición en la línea 168 del archivo [generate_synthetic_data.py](#).

4.120.2.7. docentes_df

```
def data.generate_synthetic_data.docentes_df = generate_docentes(num_docentes, universidades_dict,
```

```
cod_asignaturas_dict, asignaturas_dict)
```

Definición en la línea 440 del archivo [generate_synthetic_data.py](#).

4.120.2.8. ebau_prueba_df

```
def data.generate_synthetic_data.ebau_prueba_df = generate_ebau_prueba(num_alumnos, alumnos_df['id'].tolist(),
```

```
universidades_dict)
```

Definición en la línea 441 del archivo [generate_synthetic_data.py](#).

4.120.2.9. educacion

```
dictionary data.generate_synthetic_data.educacion
```

Definición en la línea 22 del archivo [generate_synthetic_data.py](#).

4.120.2.10. `egresados_df`

```
def data.generate_synthetic_data.egresados_df = generate_egresados(alumnos_df, matricula_df,  
lineas_actas_df)
```

Definición en la línea 442 del archivo [generate_synthetic_data.py](#).

4.120.2.11. `fake`

```
data.generate_synthetic_data.fake = Faker('es_ES')
```

Definición en la línea 19 del archivo [generate_synthetic_data.py](#).

4.120.2.12. `gestores_df`

```
def data.generate_synthetic_data.gestores_df = generate_gestores(num_gestores, universidades_dict)
```

Definición en la línea 443 del archivo [generate_synthetic_data.py](#).

4.120.2.13. `index`

```
data.generate_synthetic_data.index
```

Definición en la línea 446 del archivo [generate_synthetic_data.py](#).

4.120.2.14. `lineas_actas_df`

```
def data.generate_synthetic_data.lineas_actas_df = generate_lineas_actas(asignaturas_matriculadas_df)
```

Definición en la línea 439 del archivo [generate_synthetic_data.py](#).

4.120.2.15. `matricula_df`

```
def data.generate_synthetic_data.matricula_df = generate_matricula(alumnos_df)
```

Definición en la línea 437 del archivo [generate_synthetic_data.py](#).

4.120.2.16. num_alumnos

```
int data.generate_synthetic_data.num_alumnos = 5000
```

Definición en la línea 427 del archivo [generate_synthetic_data.py](#).

4.120.2.17. num_docentes

```
int data.generate_synthetic_data.num_docentes = 100
```

Definición en la línea 428 del archivo [generate_synthetic_data.py](#).

4.120.2.18. num_gestores

```
int data.generate_synthetic_data.num_gestores = 50
```

Definición en la línea 429 del archivo [generate_synthetic_data.py](#).

4.120.2.19. universidades_dict

```
dictionary data.generate_synthetic_data.universidades_dict
```

Valor inicial:

```
00001 = {
00002     'Universidad de Madrid': 'UDM01',
00003     'Universidad de Barcelona': 'UDB02',
00004     'Universidad de La Laguna': 'ULL03',
00005 }
```

Definición en la línea 77 del archivo [generate_synthetic_data.py](#).

4.121. Referencia del Namespace data.queries

Funciones

- def [check_data](#) (query, params)
- def [cache_query](#) (func)
- def [alumnos_all](#) ()
- def [resumen_alumno](#) (alumno_id, titulacion)
- def [resumen_gestor](#) (gestor_id)
- def [nota_media_alumno_titulacion](#) (alumno_id, titulacion)
- def [curso_academico_alumnado](#) (alumno_id, titulacion)
- def [asignaturas_matriculadas](#) (alumno_id, curso_academico, titulacion)
- def [titulacion_alumnado](#) (alumno_id)
- def [asignaturas_superadas](#) (alumno_id, curso_academico, titulacion)

- def `calif_cualitativa_asignatura` (alumno_id, curso_academico, titulacion)
- def `calif_numerica_asignatura` (alumno_id, curso_academico, titulacion)
- def `asignaturas_matriculadas_y_superadas` (alumno_id, curso_academico, titulacion)
- def `asignaturas_superadas_media_abandono` (curso_academico, `asignaturas_matriculadas`, titulacion, cod_universidad)
- def `calif_cualitativa_comparativa` (curso_academico, `asignaturas_matriculadas`, titulacion, cod_universidad)
- def `calif_cualitativa_alumno_asignaturas` (alumno_id, curso_academico, `asignaturas_matriculadas`, titulacion)
- def `nota_media_general_mi_nota` (curso_academico, `asignaturas_matriculadas`, alumno_id, titulacion, cod_universidad)
- def `alumnos_repetidores_nuevos` (docente_id, curso_academico, asignaturas)
- def `asignaturas_docente` (id_docente, titulacion)
- def `curso_academico_docente` (id_docente, asignatura)
- def `titulacion_docente` (id_docente)
- def `resumen_docente` (id_docente, titulacion)
- def `docentes_all` ()
- def `alumnos_genero_docente` (id_docente, asignaturas, curso_academico)
- def `alumnos_nota_media_docente` (asignaturas, curso_academico)
- def `alumnos_nota_cualitativa_docente` (asignaturas, curso_academico)
- def `curso_academico_actas_titulacion` (titulacion)
- def `asignaturas_actas_titulacion` (titulacion, curso_academico)
- def `calif_all_cualitativa_asignaturas` (titulacion, curso_academico, asignaturas)
- def `calif_media_asignaturas` (titulacion, curso_academico, asignaturas)
- def `gestores_all` ()
- def `numero_alumnos_matriculados_universidad` (universidad)
- def `universidades_gestor` (gestor_id)
- def `curso_academico_universidad` (cod_universidad)
- def `titulaciones_universidad_gestor` (cod_universidad, curso_academico)
- def `alumnos_nuevo_ingreso_genero_titulacion` (curso_academico, titulaciones, cod_universidad)
- def `alumnos_egresados_genero_titulacion` (cod_universidad, curso_academico, titulaciones)
- def `alumnos_egresados_nacionalidad_titulacion` (cod_universidad, curso_academico, titulaciones)
- def `alumnos_nuevo_ingreso_nacionalidad_titulacion` (cod_universidad, curso_academico, titulaciones)
- def `nota_media_acceso_titulacion` (cod_universidad)
- def `duracion_media_estudios_nota_gestor` (cod_universidad)
- def `cueros_academicos_egresados` (cod_universidad)
- def `tasa_abandono_titulacion_gestor` (cod_universidad, curso_academico)
- def `tasa_graduacion_titulacion_gestor` (cod_universidad, curso_academico)
- def `universidad_alumno` (alumno_id)
- def `universidades_docente` (id_docente)
- def `data_for_model` ()

4.121.1. Documentación de las funciones

4.121.1.1. alumnos_all()

```
def data.queries.alumnos_all ( )
```

Definición en la línea 53 del archivo [queries.py](#).

```
00053 def alumnos_all():
00054     query = queries["alumnado"]["common"]["alumnos_all"]
00055     return check_data(query, {})
00056
00057
00058 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.2. alumnos_egresados_genero_titulacion()

```
def data.queries.alumnos_egresados_genero_titulacion (
    cod_universidad,
    curso_academico,
    titulaciones )
```

Definición en la línea 405 del archivo [queries.py](#).

```
00405 def alumnos_egresados_genero_titulacion(cod_universidad, curso_academico, titulaciones):
00406     query = queries["gestor"]["graphs"]["indicadores"][
00407         "alumnos_egresados_genero_titulacion"
00408     ]
00409     params = {
00410         "cod_universidad": cod_universidad,
00411         "curso_academico": curso_academico,
00412         "titulaciones": titulaciones,
00413     }
00414
00415     return check_data(query, params)
00416
00417
00418 @cache_query
```

Gráfico de llamadas para esta función:

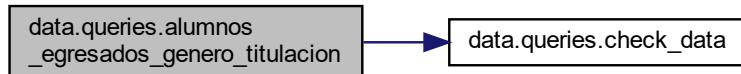
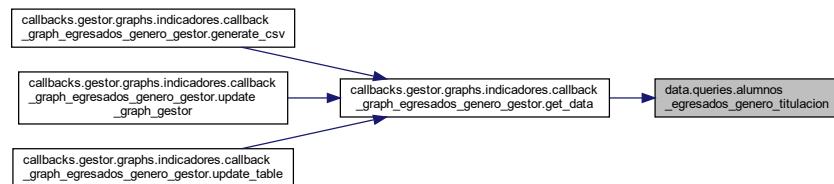


Gráfico de llamadas a esta función:



4.121.1.3. `alumnos_egresados_nacionalidad_titulacion()`

```
def data.queries.alumnos_egresados_nacionalidad_titulacion (  
    cod_universidad,  
    curso_academico,  
    titulaciones )
```

Definición en la línea 419 del archivo [queries.py](#).

```
00421 ):  
00422     query = queries["gestor"]["graphs"]["indicadores"] [  
00423         "alumnos_egresados_nacionalidad_titulacion"  
00424     ]  
00425     params = {  
00426         "cod_universidad": cod_universidad,  
00427         "curso_academico": curso_academico,  
00428         "titulaciones": titulaciones,  
00429     }  
00430  
00431     return check_data(query, params)  
00432  
00433  
00434 @cache_query
```

Gráfico de llamadas para esta función:

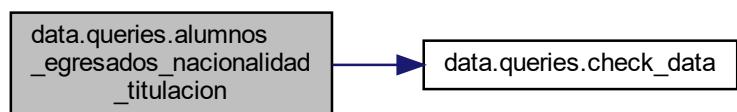
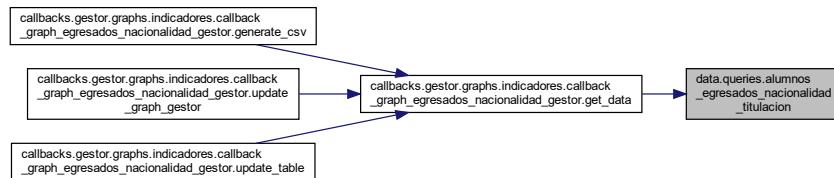


Gráfico de llamadas a esta función:



4.121.1.4. alumnos_genero_docente()

```
def data.queries.alumnos_genero_docente (
    id_docente,
    asignaturas,
    curso_academico )
```

Definición en la línea 283 del archivo [queries.py](#).

```
00283 def alumnos_genero_docente(id_docente, asignaturas, curso_academico):
00284     query = queries["docente"]["graphs"]["personal"]["alumnos_genero_docente"]
00285     params = {
00286         "id_docente": id_docente,
00287         "asignaturas": asignaturas,
00288         "curso_academico": curso_academico,
00289     }
00290
00291     return check_data(query, params)
00292
00293
00294 @cache_query
```

Gráfico de llamadas para esta función:

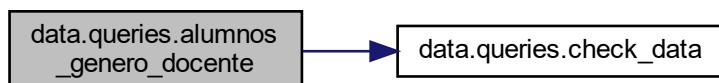
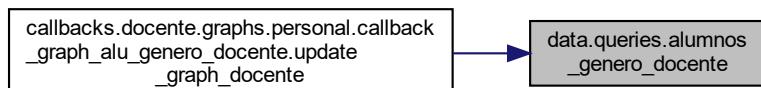


Gráfico de llamadas a esta función:



4.121.1.5. `alumnos_nota_cualitativa_docente()`

```
def data.queries.alumnos_nota_cualitativa_docente (
    asignaturas,
    curso_academico )
```

Definición en la línea 303 del archivo `queries.py`.

```
00303 def alumnos_nota_cualitativa_docente(asignaturas, curso_academico):
00304     query = queries["docente"]["graphs"]["personal"]["alumnos_nota_cualitativa_docente"]
00305     params = {"asignaturas": asignaturas, "curso_academico": curso_academico}
00306
00307     return check_data(query, params)
00308
00309
00310 @cache_query
```

Gráfico de llamadas para esta función:

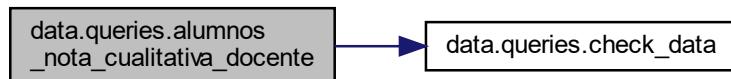
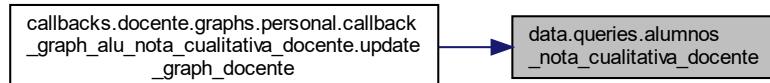


Gráfico de llamadas a esta función:



4.121.1.6. `alumnos_nota_media_docente()`

```
def data.queries.alumnos_nota_media_docente (
    asignaturas,
    curso_academico )
```

Definición en la línea 295 del archivo `queries.py`.

```
00295 def alumnos_nota_media_docente(asignaturas, curso_academico):
00296     query = queries["docente"]["graphs"]["personal"]["alumnos_nota_media_docente"]
00297     params = {"asignaturas": asignaturas, "curso_academico": curso_academico}
00298
00299     return check_data(query, params)
00300
00301
00302 @cache_query
```

Gráfico de llamadas para esta función:

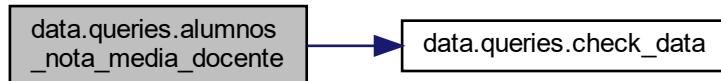
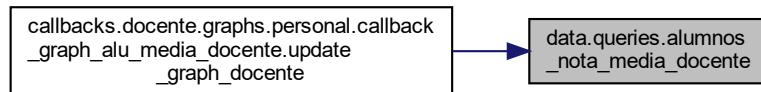


Gráfico de llamadas a esta función:



4.121.1.7. alumnos_nuevo_ingreso_genero_titulacion()

```
def data.queries.alumnos_nuevo_ingreso_genero_titulacion (
    curso_academico,
    titulaciones,
    cod_universidad )
```

Definición en la línea 389 del archivo [queries.py](#).

```
00391 ):
00392     query = queries["gestor"]["graphs"]["indicadores"][
00393         "alumnos_nuevo_ingreso_genero_titulacion"
00394     ]
00395     params = {
00396         "curso_academico": curso_academico,
00397         "titulaciones": titulaciones,
00398         "cod_universidad": cod_universidad,
00399     }
00400
00401     return check_data(query, params)
00402
00403
00404 @cache_query
```

Gráfico de llamadas para esta función:

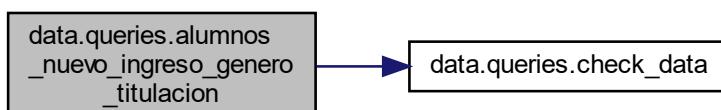
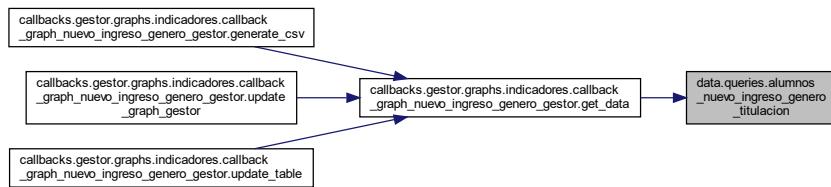


Gráfico de llamadas a esta función:



4.121.1.8. `alumnos_nuevo_ingreso_nacionalidad_titulacion()`

```
def data.queries.alumnos_nuevo_ingreso_nacionalidad_titulacion (
    cod_universidad,
    curso_academico,
    titulaciones )
```

Definición en la línea 435 del archivo [queries.py](#).

```
00437 ):
00438     query = queries["gestor"]["graphs"]["indicadores"][
00439         "alumnos_nuevo_ingreso_nacionalidad_titulacion"
00440     ]
00441     params = {
00442         "cod_universidad": cod_universidad,
00443         "curso_academico": curso_academico,
00444         "titulaciones": titulaciones,
00445     }
00446
00447     return check_data(query, params)
00448
00449
00450 @cache_query
```

Gráfico de llamadas para esta función:

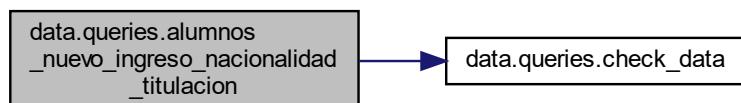
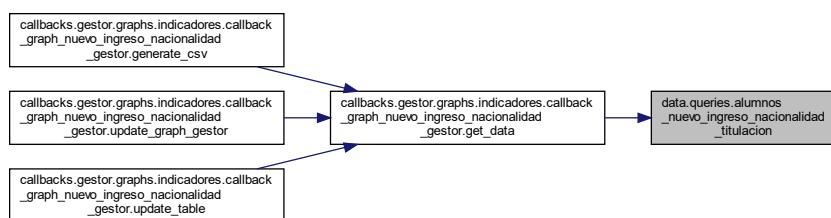


Gráfico de llamadas a esta función:



4.121.1.9. alumnos_repetidores_nuevos()

```
def data.queries.alumnos_repetidores_nuevos (
    docente_id,
    curso_academico,
    asignaturas )
```

Definición en la línea 233 del archivo [queries.py](#).

```
00233 def alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas):
00234     query = queries["docente"]["graphs"]["personal"]["alumnos_repetidores_nuevos"]
00235     params = {
00236         "docente_id": docente_id,
00237         "curso_academico": curso_academico,
00238         "asignaturas": asignaturas,
00239     }
00240     return check_data(query, params)
00242
00243
00244 @cache_query
```

Gráfico de llamadas para esta función:

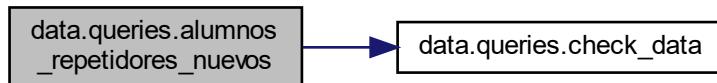
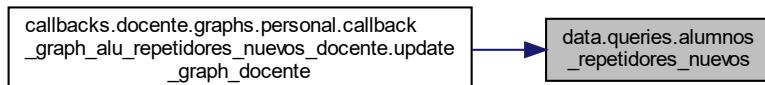


Gráfico de llamadas a esta función:



4.121.1.10. asignaturas_actas_titulacion()

```
def data.queries.asignaturas_actas_titulacion (
    titulacion,
    curso_academico )
```

Definición en la línea 319 del archivo [queries.py](#).

```
00319 def asignaturas_actas_titulacion(titulacion, curso_academico):
00320     query = queries["docente"]["filters"]["asignaturas_actas_titulacion"]
00321     params = {"titulacion": titulacion, "curso_academico": curso_academico}
00322
00323     return check_data(query, params)
00324
00325
00326 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.11. `asignaturas_docente()`

```
def data.queries.asignaturas_docente (
    id_docente,
    titulacion )
```

Definición en la línea 245 del archivo `queries.py`.

```
00245 def asignaturas_docente(id_docente, titulacion):
00246     query = queries["docente"]["filters"]["asignaturas_docente"]
00247     params = {"id_docente": id_docente, "titulacion": titulacion}
00248
00249     return check_data(query, params)
00250
00251
00252 @cache_query
```

Gráfico de llamadas para esta función:

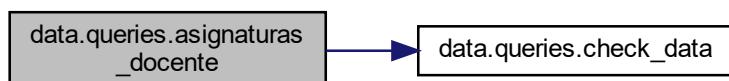
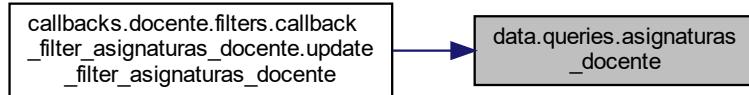


Gráfico de llamadas a esta función:



4.121.1.12. asignaturas_matriculadas()

```
def data.querries.asignaturas_matriculadas (
    alumno_id,
    curso_academico,
    titulacion )
```

Definición en la línea 96 del archivo `queries.py`.

```
00096 def asignaturas_matriculadas(alumno_id, curso_academico, titulacion):
00097     query = queries["alumnado"]["filters"]["asignaturas_matriculadas"]
00098     params = {
00099         "alumno_id": alumno_id,
00100         "curso_academico": curso_academico,
00101         "titulacion": titulacion,
00102     }
00103
00104     return check_data(query, params)
00105
00106
00107 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.13. `asignaturas_matriculadas_y_superadas()`

```
def data.queries.asignaturas_matriculadas_y_superadas (
    alumno_id,
    curso_academico,
    titulacion )
```

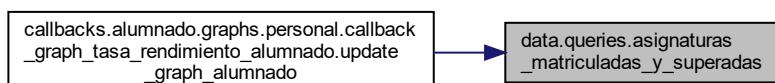
Definición en la línea 154 del archivo `queries.py`.

```
00154 def asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion):
00155     query = queries["alumnado"]["graphs"]["personal"][
00156         "asignaturas_matriculadas_y_superadas"
00157     ]
00158     params = {
00159         "alumno_id": alumno_id,
00160         "curso_academico": curso_academico,
00161         "titulacion": titulacion,
00162     }
00163
00164     return check_data(query, params)
00165
00166
00167 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.14. `asignaturas_superadas()`

```
def data.queries.asignaturas_superadas (
    alumno_id,
    curso_academico,
    titulacion )
```

Definición en la línea 116 del archivo `queries.py`.

```
00116 def asignaturas_superadas(alumno_id, curso_academico, titulacion):
00117     query = queries["alumnado"]["graphs"]["personal"][
00118         "curso_academico_asignaturas_superadas"
```

```

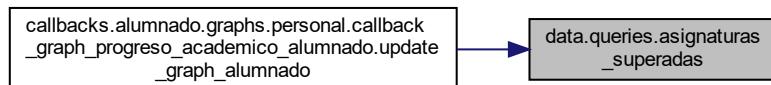
00119     ]
00120     params = {
00121         "alumno_id": alumno_id,
00122         "curso_academico": curso_academico,
00123         "titulacion": titulacion,
00124     }
00125
00126     return check_data(query, params)
00127
00128
00129 @cache_query

```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.15. asignaturas_superadas_media_abandono()

```

def data.queries.asignaturas_superadas_media_abandono (
    curso_academico,
    asignaturas_matriculadas,
    titulacion,
    cod_universidad )

```

Definición en la línea 168 del archivo [queries.py](#).

```

00170 ):
00171     query = queries["alumnado"]["graphs"]["general"][
00172         "asignaturas_superadas_media_abandono"
00173     ]
00174     params = {
00175         "curso_academico": curso_academico,
00176         "asignaturas_matriculadas": asignaturas_matriculadas,
00177         "titulacion": titulacion,
00178         "cod_universidad": cod_universidad,
00179     }
00180
00181     return check_data(query, params)
00182
00183
00184 @cache_query

```

Gráfico de llamadas para esta función:

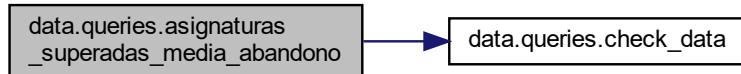
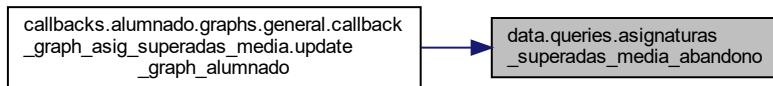


Gráfico de llamadas a esta función:



4.121.1.16. `cache_query()`

```
def data.queries.cache_query (\n    func )
```

Decorador para cachear los resultados de las consultas.

Args:
func: Función a decorar

Returns:
wrapper: Función decorada

Definición en la línea 35 del archivo [queries.py](#).

```
00035 def cache_query(func):\n00036     """\n00037     Decorador para cachear los resultados de las consultas.\n00038 \n00039     Args:\n00040         func: Función a decorar\n00041 \n00042     Returns:\n00043         wrapper: Función decorada\n00044     """\n00045     @lru_cache(maxsize=32)\n00046     def wrapper(*args, **kwargs):\n00047         return func(*args, **kwargs)\n00048     return wrapper\n00049 \n00050 \n00051 \n00052 @cache_query
```

4.121.1.17. calif_all_cualitativa_asignaturas()

```
def data.queries.calif_all_cualitativa_asignaturas (
    titulacion,
    curso_academico,
    asignaturas )
```

Definición en la línea 327 del archivo [queries.py](#).

```
00327 def calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignaturas):
00328     query = queries["docente"]["graphs"]["general"]["calif_all_cualitativa_asignaturas"]
00329     params = {
00330         "titulacion": titulacion,
00331         "curso_academico": curso_academico,
00332         "asignaturas": asignaturas,
00333     }
00334
00335     return check_data(query, params)
00336
00337
00338 @cache_query
```

Gráfico de llamadas para esta función:

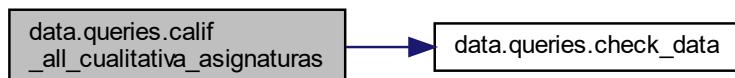
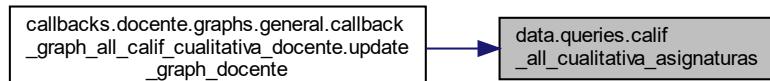


Gráfico de llamadas a esta función:



4.121.1.18. calif_cualitativa_alumno_asignaturas()

```
def data.queries.calif_cualitativa_alumno_asignaturas (
    alumno_id,
    curso_academico,
    asignaturas_matriculadas,
    titulacion )
```

Definición en la línea 200 del archivo [queries.py](#).

```
00202 ):
00203     query = queries["alumnado"]["graphs"]["general"][
00204         "calif_cualitativa_alumno_asignaturas"
```

```

00205     ]
00206     params = {
00207         "alumno_id": alumno_id,
00208         "curso_academico": curso_academico,
00209         "asignaturas_matriculadas": asignaturas_matriculadas,
00210         "titulacion": titulacion,
00211     }
00212
00213     return check_data(query, params)
00214
00215
00216 @cache_query

```

Gráfico de llamadas para esta función:

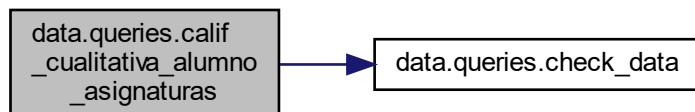
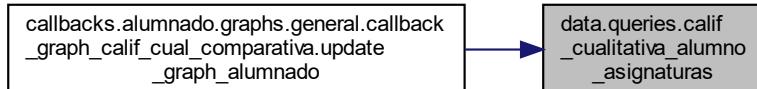


Gráfico de llamadas a esta función:



4.121.1.19. `calif_cualitativa_asignatura()`

```

def data.queries.calif_cualitativa_asignatura (
    alumno_id,
    curso_academico,
    titulacion )

```

Definición en la línea 130 del archivo `queries.py`.

```

00130 def calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion):
00131     query = queries["alumnado"]["graphs"]["personal"]["calif_cualitativa_asignatura"]
00132     params = {
00133         "alumno_id": alumno_id,
00134         "curso_academico": curso_academico,
00135         "titulacion": titulacion,
00136     }
00137
00138     return check_data(query, params)
00139
00140
00141 @cache_query

```

Gráfico de llamadas para esta función:

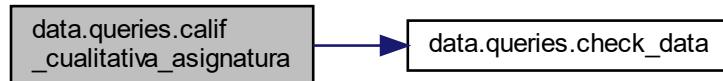
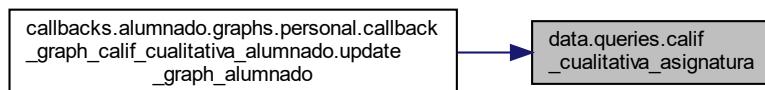


Gráfico de llamadas a esta función:



4.121.1.20. calif_cualitativa_comparativa()

```
def data.queries.calif_cualitativa_comparativa (
    curso_academico,
    asignaturas_matriculadas,
    titulacion,
    cod_universidad )
```

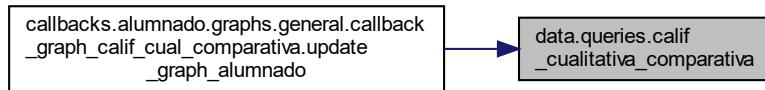
Definición en la línea 185 del archivo [queries.py](#).

```
00187 ):
00188     query = queries["alumnado"]["graphs"]["general"]["calif_cualitativa_comparativa"]
00189     params = {
00190         "curso_academico": curso_academico,
00191         "asignaturas_matriculadas": asignaturas_matriculadas,
00192         "titulacion": titulacion,
00193         "cod_universidad": cod_universidad,
00194     }
00195
00196     return check_data(query, params)
00197
00198
00199 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.21. `calif_media_asignaturas()`

```
def data.queries.calif_media_asignaturas (\n    titulacion,\n    curso_academico,\n    asignaturas )
```

Definición en la línea 339 del archivo [queries.py](#).

```
00339 def calif_media_asignaturas(titulacion, curso_academico, asignaturas):\n00340     query = queries["docente"]["graphs"]["general"]["calif_media_asignaturas"]\n00341     params = {\n00342         "titulacion": titulacion,\n00343         "curso_academico": curso_academico,\n00344         "asignaturas": asignaturas,\n00345     }\n00346\n00347     return check_data(query, params)\n00348\n00349\n00350 @cache_query
```

Gráfico de llamadas para esta función:

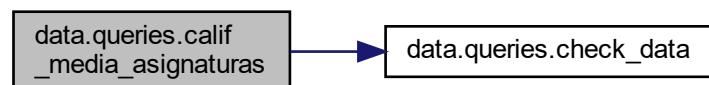
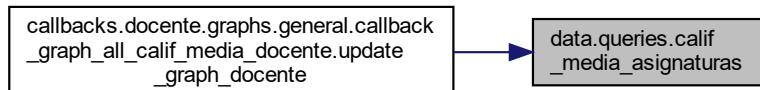


Gráfico de llamadas a esta función:



4.121.1.22. calif_numerica_asignatura()

```
def data.queries.calif_numerica_asignatura (
    alumno_id,
    curso_academico,
    titulacion )
```

Definición en la línea 142 del archivo [queries.py](#).

```
00142 def calif_numerica_asignatura(alumno_id, curso_academico, titulacion):
00143     query = queries["alumnado"]["graphs"]["personal"]["calif_numerica_asignatura"]
00144     params = {
00145         "alumno_id": alumno_id,
00146         "curso_academico": curso_academico,
00147         "titulacion": titulacion,
00148     }
00149     return check_data(query, params)
00151
00152
00153 @cache_query
```

Gráfico de llamadas para esta función:

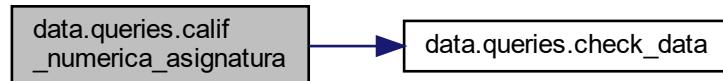
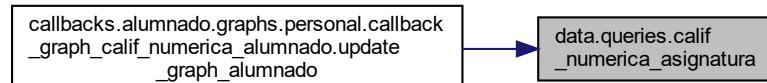


Gráfico de llamadas a esta función:



4.121.1.23. check_data()

```
def data.queries.check_data (
    query,
    params )
```

Ejecuta una consulta en la base de datos y maneja los errores.

Args:

query (str): Consulta SQL
params (dict): Parámetros de la consulta

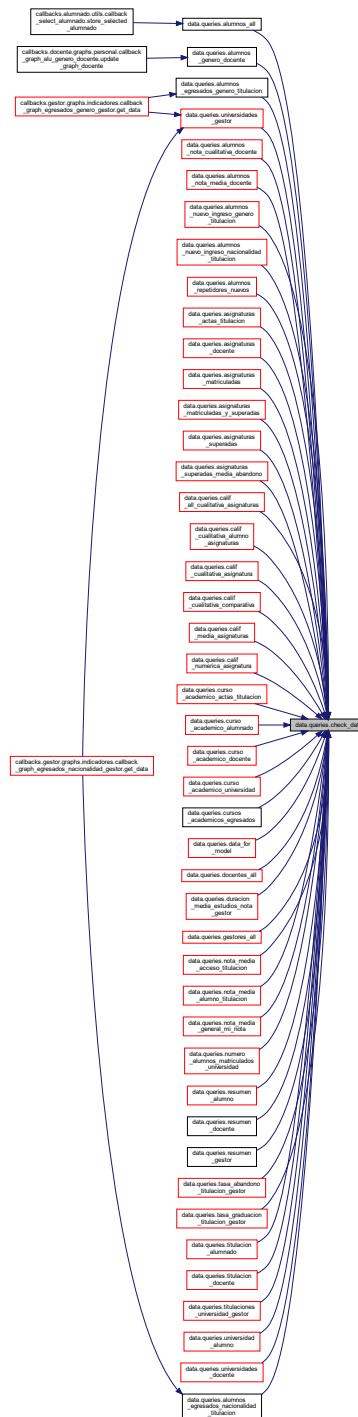
Returns:

list: Resultado de la consulta

Definición en la línea 16 del archivo [queries.py](#).

```
00016 def check_data(query, params):
00017     """
00018 Ejecuta una consulta en la base de datos y maneja los errores.
00019
00020 Args:
00021 query (str): Consulta SQL
00022 params (dict): Parámetros de la consulta
00023
00024 Returns:
00025 list: Resultado de la consulta
00026 """
00027 try:
00028     data = db.execute_query(query, params)
00029 except Exception as e:
00030     print("Query execution failed: ", e)
00031     return []
00032
00033 return data
00034
```

Gráfico de llamadas a esta función:



4.121.1.24. curso_academico_actas_titulacion()

```
def data.queries.curso_academico_actas_titulacion (
    titulacion )
```

Definición en la línea 311 del archivo `queries.py`.

```
00311 def curso_academico_actas_titulacion(titulacion):  
00312     query = queries["docente"]["filters"]["curso_academico_actas_titulacion"]  
00313     params = {"titulacion": titulacion}  
00314  
00315     return check_data(query, params)  
00316  
00317  
00318 @cache_query
```

Gráfico de llamadas para esta función:

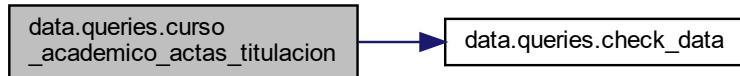


Gráfico de llamadas a esta función:



4.121.1.25. `curso_academico_alumnado()`

```
def data.queries.curso_academico_alumnado (  
    alumno_id,  
    titulacion )
```

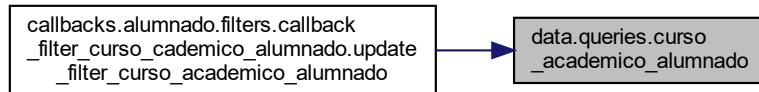
Definición en la línea 88 del archivo `queries.py`.

```
00088 def curso_academico_alumnado(alumno_id, titulacion):  
00089     query = queries["alumnado"]["filters"]["curso_academico_alumnado"]  
00090     params = {"alumno_id": alumno_id, "titulacion": titulacion}  
00091  
00092     return check_data(query, params)  
00093  
00094  
00095 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.26. curso_academico_docente()

```
def data.queries.curso_academico_docente (
    id_docente,
    asignatura )
```

Definición en la línea 253 del archivo [queries.py](#).

```
00253 def curso_academico_docente(id_docente, asignatura):
00254     query = queries["docente"]["filters"]["curso_academico_docente"]
00255     params = {"id_docente": id_docente, "asignatura": asignatura}
00256
00257     return check_data(query, params)
00258
00259
00260 @cache_query
```

Gráfico de llamadas para esta función:

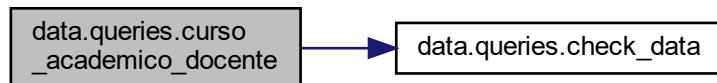
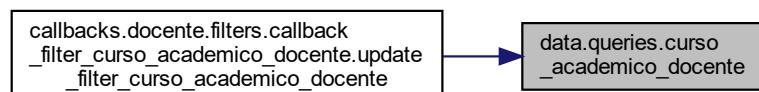


Gráfico de llamadas a esta función:



4.121.1.27. curso_academico_universidad()

```
def data.queries.curso_academico_universidad (
    cod_universidad )
```

Definición en la línea 373 del archivo [queries.py](#).

```
00373 def curso_academico_universidad(cod_universidad):
00374     query = queries["gestor"]["filters"]["curso_academico_universidad"]
00375     params = {"cod_universidad": cod_universidad}
00376
00377     return check_data(query, params)
00378
00379
00380 @cache_query
```

Gráfico de llamadas para esta función:

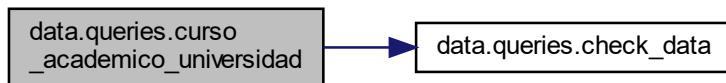
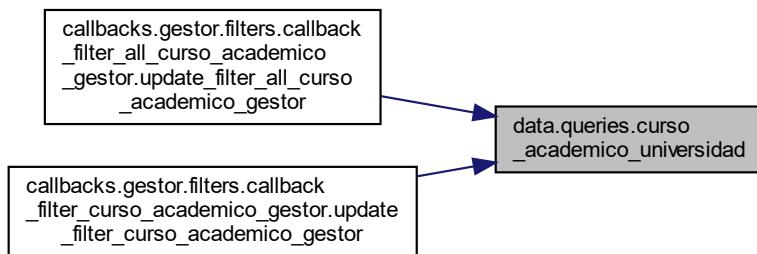


Gráfico de llamadas a esta función:



4.121.1.28. cursos_academicos_egresados()

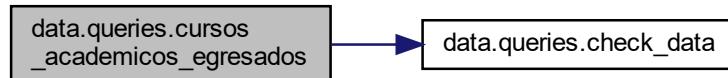
```
def data.queries.cursos_academicos_egresados (
    cod_universidad )
```

Definición en la línea 469 del archivo [queries.py](#).

```
00469 def cursos_academicos_egresados(cod_universidad):
00470     query = queries["gestor"]["common"]["cursos_academicos_de_egresados"]
00471     params = {"cod_universidad": cod_universidad}
00472
00473     return check_data(query, params)
00474
```

```
00475
00476 @cache_query
```

Gráfico de llamadas para esta función:



4.121.1.29. data_for_model()

```
def data.queries.data_for_model ( )
```

Definición en la línea 512 del archivo [queries.py](#).

```
00512 def data_for_model():
00513     query = queries["alumnado"]["common"]["data_for_model"]
00514     data = check_data(query, {})
00515
00516     if not data:
00517         return []
00518
00519     df = pd.DataFrame(data)
00520
00521     # Ajustar los tipos de datos de las columnas
00522     df = df.astype({
00523         'id': str,
00524         'anio_nac': int,
00525         'nacionalidad': str,
00526         'sexo': str,
00527         'titulacion': str,
00528         'nota_def_acceso': float,
00529         'nota_media': float,
00530         'abandona': str
00531     })
00532     return df
00533
00534 @cache_query
```

Gráfico de llamadas para esta función:

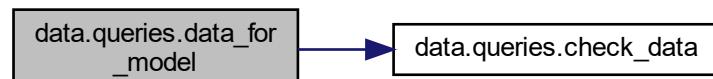
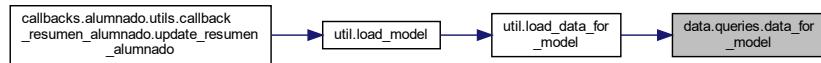


Gráfico de llamadas a esta función:



4.121.1.30. `docentes_all()`

```
def data.queries.docentes_all ( )
```

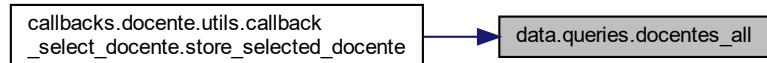
Definición en la línea 277 del archivo `queries.py`.

```
00277 def docentes_all():
00278     query = queries["docente"]["common"]["docentes_all"]
00279     return check_data(query, {})
00280
00281
00282 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.31. duracion_media_estudios_nota_gestor()

```
def data.queries.duracion_media_estudios_nota_gestor (
    cod_universidad )
```

Definición en la línea 459 del archivo [queries.py](#).

```
00459 def duracion_media_estudios_nota_gestor(cod_universidad):
00460     query = queries["gestor"]["graphs"]["resultados"][
00461         "duracion_media_estudios_nota_gestor"
00462     ]
00463     params = {"cod_universidad": cod_universidad}
00464
00465     return check_data(query, params)
00466
00467
00468 @cache_query
```

Gráfico de llamadas para esta función:

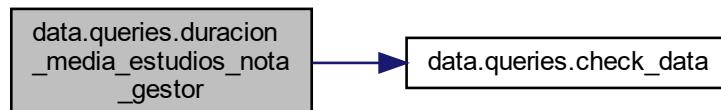
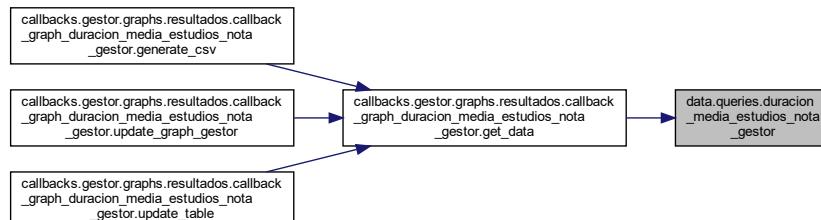


Gráfico de llamadas a esta función:



4.121.1.32. gestores_all()

```
def data.queries.gestores_all( )
```

Definición en la línea 351 del archivo [queries.py](#).

```
00351 def gestores_all():
00352     query = queries["gestor"]["common"]["gestores_all"]
00353     return check_data(query, {})
00354
00355
00356 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.33. `nota_media_acceso_titulacion()`

```
def data.queries.nota_media_acceso_titulacion (cod_universidad )
```

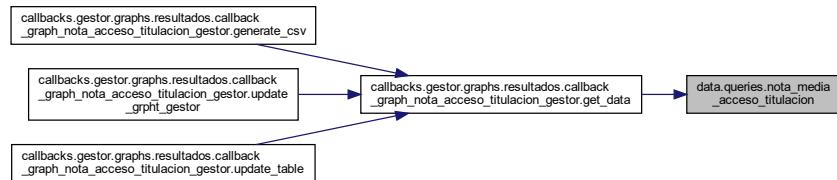
Definición en la línea 451 del archivo `queries.py`.

```
00451 def nota_media_acceso_titulacion(cod_universidad):
00452     query = queries["gestor"]["graphs"]["resultados"]["nota_media_acceso_titulacion"]
00453     params = {"cod_universidad": cod_universidad}
00454
00455     return check_data(query, params)
00456
00457
00458 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.34. nota_media_alumno_titulacion()

```
def data.queries.nota_media_alumno_titulacion (
    alumno_id,
    titulacion )
```

Definición en la línea 75 del archivo `queries.py`.

```
00075 def nota_media_alumno_titulacion(alumno_id, titulacion):
00076     query = queries["alumnado"]["common"]["nota_media_alumno_titulacion"]
00077     params = {"alumno_id": alumno_id, "titulacion": titulacion}
00078
00079     data = check_data(query, params)
00080
00081     if not data:
00082         return "No disponible"
00083
00084     return round(data[0][0], 2)
00085
00086
00087 @cache_query
```

Gráfico de llamadas para esta función:

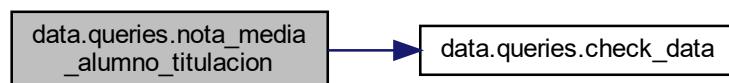


Gráfico de llamadas a esta función:



4.121.1.35. nota_media_general_mi_nota()

```
def data.queries.nota_media_general_mi_nota (
    curso_academico,
    asignaturas_matriculadas,
    alumno_id,
    titulacion,
    cod_universidad )
```

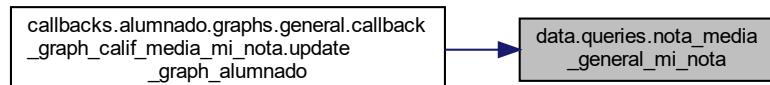
Definición en la línea 217 del archivo [queries.py](#).

```
00219 ):
00220     query = queries["alumnado"]["graphs"]["general"]["nota_media_general_mi_nota"]
00221     params = {
00222         "curso_academico": curso_academico,
00223         "asignaturas_matriculadas": asignaturas_matriculadas,
00224         "alumno_id": alumno_id,
00225         "titulacion": titulacion,
00226         "cod_universidad": cod_universidad,
00227     }
00228
00229     return check_data(query, params)
00230
00231
00232 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.36. numero_alumnos_matriculados_universidad()

```
def data.queries.numero_alumnos_matriculados_universidad (
    universidad )
```

Definición en la línea 357 del archivo [queries.py](#).

```
00357 def numero_alumnos_matriculados_universidad(universidad):
00358     query = queries["gestor"]["common"]["numero_alumnos_matriculados_universidad"]
```

```

00359     params = {"universidad": universidad}
00360
00361     return check_data(query, params)
00362
00363
00364 @cache_query

```

Gráfico de llamadas para esta función:

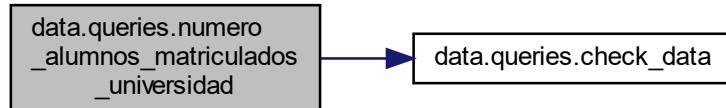
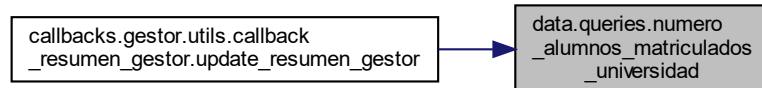


Gráfico de llamadas a esta función:



4.121.1.37. resumen_alumno()

```

def data.queries.resumen_alumno (
    alumno_id,
    titulacion )

```

Definición en la línea 59 del archivo [queries.py](#).

```

00059 def resumen_alumno(alumno_id, titulacion):
00060     query = queries["alumnado"]["common"]["resumen_alumno"]
00061     params = {"alumno_id": alumno_id, "titulacion": titulacion}
00062
00063     return check_data(query, params)
00064
00065
00066 @cache_query

```

Gráfico de llamadas para esta función:

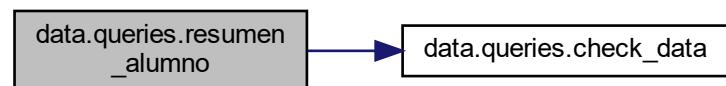
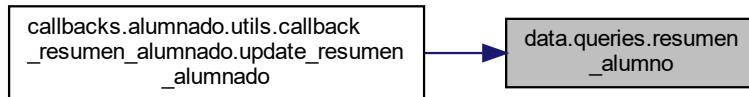


Gráfico de llamadas a esta función:



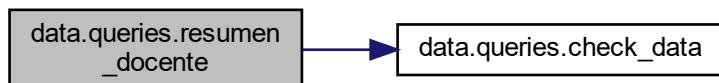
4.121.1.38. `resumen_docente()`

```
def data.queries.resumen_docente (
    id_docente,
    titulacion )
```

Definición en la línea 269 del archivo `queries.py`.

```
00269 def resumen_docente(id_docente, titulacion):
00270     query = queries["docente"]["common"]["resumen_docente"]
00271     params = {"id_docente": id_docente, "titulacion": titulacion}
00272
00273     return check_data(query, params)
00274
00275
00276 @cache_query
```

Gráfico de llamadas para esta función:



4.121.1.39. `resumen_gestor()`

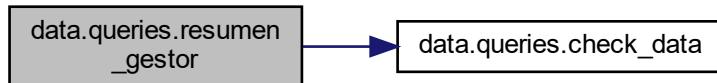
```
def data.queries.resumen_gestor (
    gestor_id )
```

Definición en la línea 67 del archivo `queries.py`.

```
00067 def resumen_gestor(gestor_id):
00068     query = queries["gestor"]["common"]["resumen_gestor"]
00069     params = {"gestor_id": gestor_id}
00070
00071     return check_data(query, params)
00072
```

```
00073
00074 @cache_query
```

Gráfico de llamadas para esta función:



4.121.1.40. tasa_abandono_titulacion_gestor()

```
def data.queries.tasa_abandono_titulacion_gestor (
    cod_universidad,
    curso_academico )
```

Definición en la línea 477 del archivo [queries.py](#).

```
00477 def tasa_abandono_titulacion_gestor(cod_universidad, curso_academico):
00478     query = queries["gestor"]["graphs"]["riesgo_abandono"][
00479         "tasa_abandono_titulacion_gestor"
00480     ]
00481     params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00482
00483     return check_data(query, params)
00484
00485
00486 @cache_query
```

Gráfico de llamadas para esta función:

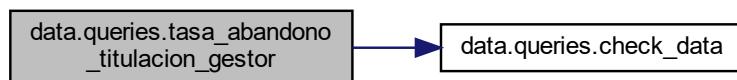
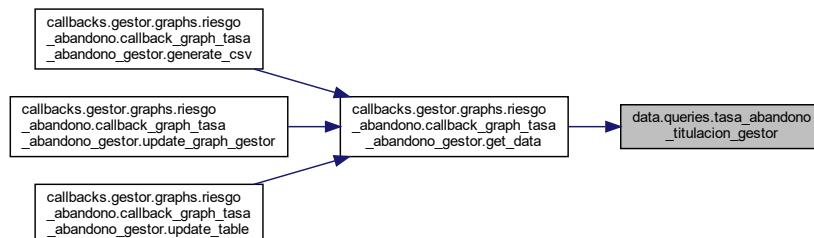


Gráfico de llamadas a esta función:



4.121.1.41. `tasa_graduacion_titulacion_gestor()`

```
def data.queries.tasa_graduacion_titulacion_gestor (
    cod_universidad,
    curso_academico )
```

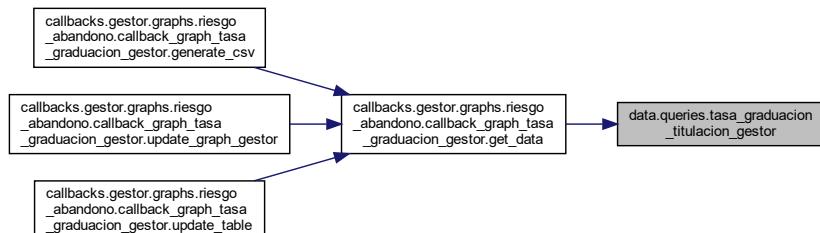
Definición en la línea 487 del archivo `queries.py`.

```
00487 def tasa_graduacion_titulacion_gestor(cod_universidad, curso_academico):
00488     query = queries["gestor"]["graphs"]["riesgo_abandono"][
00489         "tasa_graduacion_titulacion_gestor"
00490     ]
00491     params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00492
00493     return check_data(query, params)
00494
00495
00496 @cache_query
```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.121.1.42. `titulacion_alumnado()`

```
def data.queries.titulacion_alumnado (
    alumno_id )
```

Definición en la línea 108 del archivo `queries.py`.

```
00108 def titulacion_alumnado(alumno_id):
00109     query = queries["alumnado"]["filters"]["titulacion_alumnado"]
```

```

00110     params = {"alumno_id": alumno_id}
00111
00112     return check_data(query, params)
00113
00114
00115 @cache_query

```

Gráfico de llamadas para esta función:

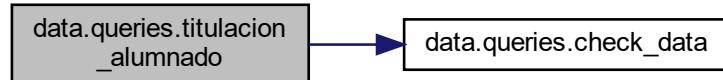
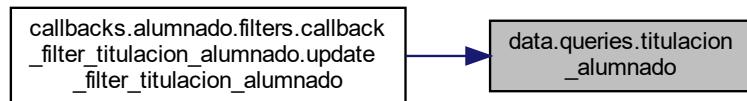


Gráfico de llamadas a esta función:



4.121.1.43. titulacion_docente()

```

def data.queries.titulacion_docente (
    id_docente )

```

Definición en la línea 261 del archivo [queries.py](#).

```

00261 def titulacion_docente(id_docente):
00262     query = queries["docente"]["filters"]["titulacion_docente"]
00263     params = {"id_docente": id_docente}
00264
00265     return check_data(query, params)
00266
00267
00268 @cache_query

```

Gráfico de llamadas para esta función:

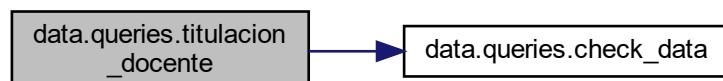
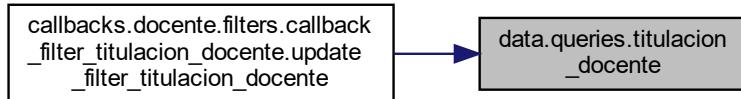


Gráfico de llamadas a esta función:



4.121.1.44. `titulaciones_universidad_gestor()`

```
def data.queries.titulaciones_universidad_gestor (\n    cod_universidad,\n    curso_academico )
```

Definición en la línea 381 del archivo `queries.py`.

```
00381 def titulaciones_universidad_gestor(cod_universidad, curso_academico):\n00382     query = queries["gestor"]["filters"]["titulaciones_universidad_gestor"]\n00383     params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}\n00384\n00385     return check_data(query, params)\n00386\n00387\n00388 @cache_query
```

Gráfico de llamadas para esta función:

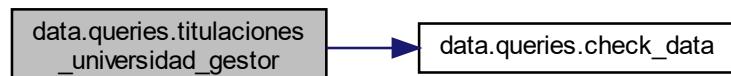
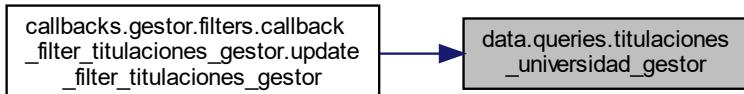


Gráfico de llamadas a esta función:



4.121.1.45. `universidad_alumno()`

```
def data.queries.universidad_alumno (
    alumno_id )
```

Definición en la línea 497 del archivo `queries.py`.

```
00497 def universidad_alumno(alumno_id):
00498     query = queries["alumnado"]["common"]["universidad_alumno"]
00499     params = {"alumno_id": alumno_id}
00500
00501     return check_data(query, params)
00502
00503
00504 @cache_query
```

Gráfico de llamadas para esta función:

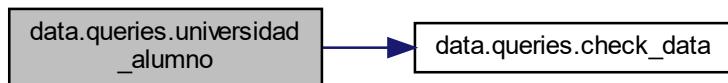
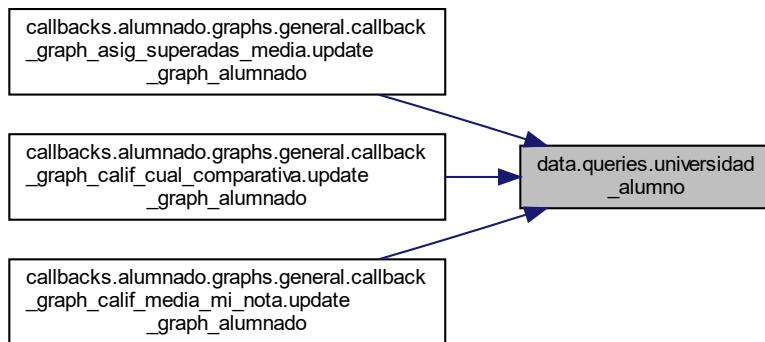


Gráfico de llamadas a esta función:



4.121.1.46. `universidades_docente()`

```
def data.queries.universidades_docente (
    id_docente )
```

Definición en la línea 505 del archivo `queries.py`.

```
00505 def universidades_docente(id_docente):
00506     query = queries["docente"]["common"]["universidades_docente"]
```

```
00507     params = {"id_docente": id_docente}
00508
00509     return check_data(query, params)
00510
00511 @cache_query
```

Gráfico de llamadas para esta función:

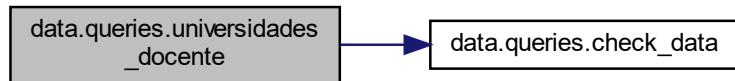
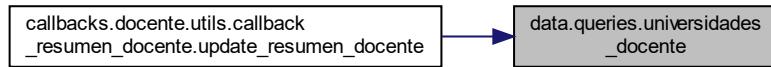


Gráfico de llamadas a esta función:



4.121.1.47. `universidades_gestor()`

```
def data.queries.universidades_gestor (
    gestor_id )
```

Definición en la línea 365 del archivo `queries.py`.

```
00365 def universidades_gestor(gestor_id):
00366     query = queries["gestor"]["common"]["universidades_gestor"]
00367     params = {"gestor_id": gestor_id}
00368
00369     return check_data(query, params)
00370
00371
00372 @cache_query
```

Gráfico de llamadas para esta función:

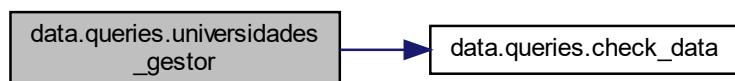
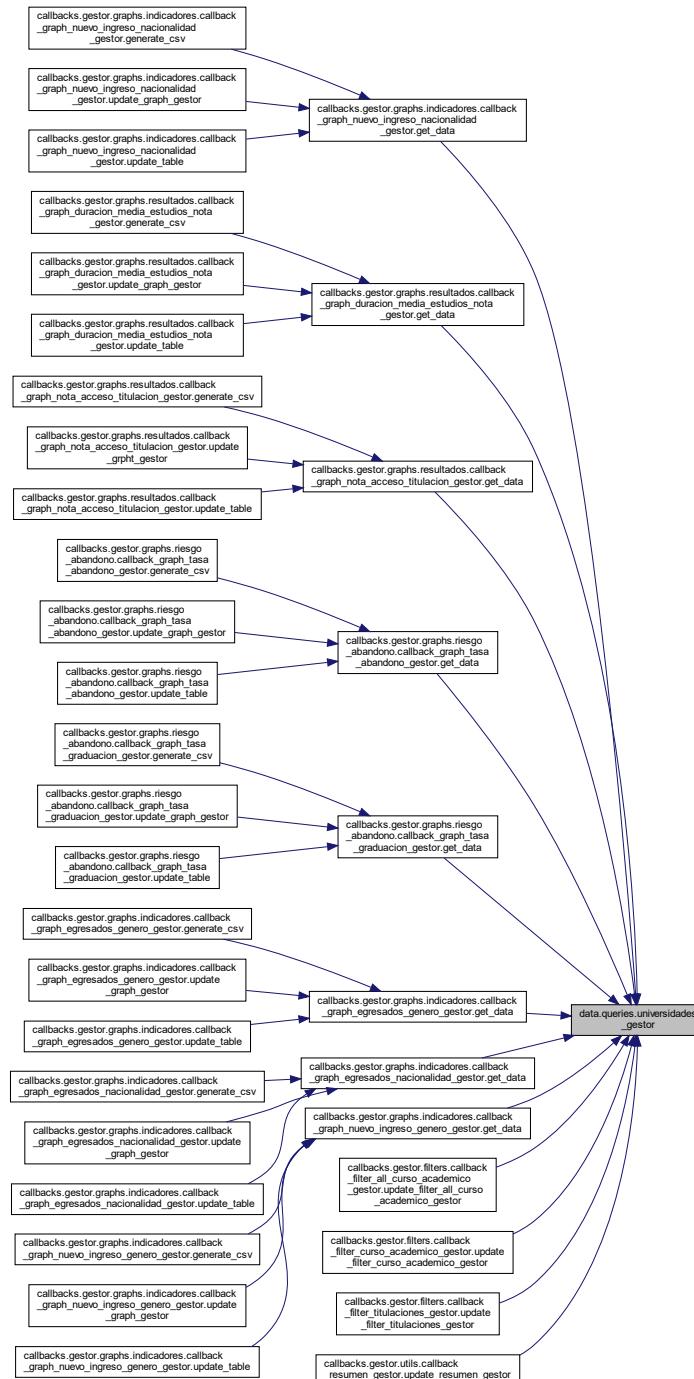


Gráfico de llamadas a esta función:



4.122. Referencia del Namespace `data.queries_dictionary`

Variables

- dictionary `queries`

4.122.1. Documentación de las variables

4.122.1.1. queries

```
dictionary data.queries_dictionary.queries
```

Definición en la línea 12 del archivo [queries_dictionary.py](#).

4.123. Referencia del Namespace layouts

Namespaces

- namespace [alumno_layout](#)
- namespace [docente_layout](#)
- namespace [gestor_layout](#)

4.124. Referencia del Namespace layouts.alumno_layout

Funciones

- def [alumno_layout\(\)](#)

4.124.1. Documentación de las funciones

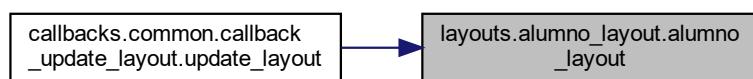
4.124.1.1. alumno_layout()

```
def layouts.alumno_layout.alumno_layout ( )
```

Definición en la línea 15 del archivo [alumno_layout.py](#).

```
00015 def alumno_layout():
00016     #
00017     # @brief Retorna el layout del perfil "Alumno"
00018     # @return Layout del alumno
00019     #
00020     return html.Div([tabs_alumnado.tabs_alumnado() ])
```

Gráfico de llamadas a esta función:



4.125. Referencia del Namespace layouts.docente_layout

Funciones

- def docente_layout ()

4.125.1. Documentación de las funciones

4.125.1.1. docente_layout()

```
def layouts.docente_layout.docente_layout ( )
```

Definición en la línea 15 del archivo [docente_layout.py](#).

```
00015 def docente_layout():
00016     #
00017     # @brief Retorna el layout del perfil "Docente"
00018     # @return Layout del docente
00019     #
00020     return html.Div([tabs_docente.tabs_docente()])
```

Gráfico de llamadas a esta función:



4.126. Referencia del Namespace layouts.gestor_layout

Funciones

- def gestor_layout ()

4.126.1. Documentación de las funciones

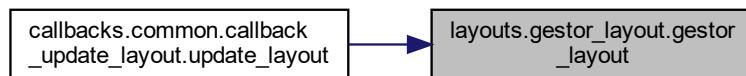
4.126.1.1. gestor_layout()

```
def layouts.gestor_layout.gestor_layout ( )
```

Definición en la línea 15 del archivo `gestor_layout.py`.

```
00015 def gestor_layout():
00016     #
00017     # @brief Retorna el layout del perfil "Gestor"
00018     # @return Layout del gestor
00019     #
00020     return html.Div([tabs_gestor.tabs_gestor()])
```

Gráfico de llamadas a esta función:



4.127. Referencia del Namespace model

Variables

- `data` = `load_data_for_model()`
- `current_year` = `datetime.now().year`
- list `columnas_categoricas` = ['nacionalidad', 'sexo', 'titulacion']
- list `columnas_numericas` = ['nota_def_acceso', 'nota_media', 'edad_actual']
- `preprocessor`
- `X` = `data[columnas_categoricas + columnas_numericas]`
- `y` = `data['abandona']`
- `ids` = `data['id']`
- `X_train` = `preprocessor.fit_transform(X_train)`
- `X_test` = `preprocessor.transform(X_test)`
- `y_train`
- `y_test`
- `id_train`
- `id_test`
- `test_size`
- `random_state`
- `stratify`
- `smote` = `SMOTE(random_state=42)`
- `poly` = `PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)`
- dictionary `modelos`
- dictionary `param_grids`
- dictionary `best_estimators` = {}
- dictionary `metrics` = {}
- `total`
- `desc`
- `grid_search` = `GridSearchCV(model, param_grids[key], cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)`

- `y_pred = grid_search.predict(X_test)`
- `mejores_modelos = sorted(metrics.items(), key=lambda item: item[1]['f1_score'], reverse=True)[:3]`
- `list mejores_estimadores = [(key, best_estimators[key]) for key, _ in mejores_modelos]`
- `voting_clf`
- `string joblib_file = "src/trained_model.pkl"`
- `y_pred_voting = voting_clf.predict(X_test)`
- `cross_val_scores_voting = cross_val_score(voting_clf, poly.transform(preprocessor.transform(X)), y, cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)`

4.127.1. Documentación de las variables

4.127.1.1. `best_estimators`

```
dictionary model.best_estimators = {}
```

Definición en la línea 112 del archivo [model.py](#).

4.127.1.2. `columnas_categoricas`

```
list model.columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
```

Definición en la línea 38 del archivo [model.py](#).

4.127.1.3. `columnas_numericas`

```
list model.columnas_numericas = ['nota_def_acceso', 'nota_media', 'edad_actual']
```

Definición en la línea 39 del archivo [model.py](#).

4.127.1.4. `cross_val_scores_voting`

```
model.cross_val_scores_voting = cross_val_score(voting_clf, poly.transform(preprocessor.transform(X)), y, cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)
```

Definición en la línea 176 del archivo [model.py](#).

4.127.1.5. current_year

```
model.current_year = datetime.now().year
```

Definición en la línea 31 del archivo [model.py](#).

4.127.1.6. data

```
model.data = load_data_for_model()
```

Definición en la línea 28 del archivo [model.py](#).

4.127.1.7. desc

```
model.desc
```

Definición en la línea 115 del archivo [model.py](#).

4.127.1.8. grid_search

```
model.grid_search = GridSearchCV(model, param_grids[key], cv=StratifiedKFold(n_splits=5),  
scoring='accuracy', n_jobs=-1)
```

Definición en la línea 117 del archivo [model.py](#).

4.127.1.9. id_test

```
model.id_test
```

Definición en la línea 55 del archivo [model.py](#).

4.127.1.10. id_train

```
model.id_train
```

Definición en la línea 55 del archivo [model.py](#).

4.127.1.11. ids

```
model.ids = data['id']
```

Definición en la línea 52 del archivo [model.py](#).

4.127.1.12. joblib_file

```
string model.joblib_file = "src/trained_model.pkl"
```

Definición en la línea 159 del archivo [model.py](#).

4.127.1.13. mejores_estimadores

```
list model.mejores_estimadores = [(key, best_estimators[key]) for key, _ in mejores_modelos]
```

Definición en la línea 146 del archivo [model.py](#).

4.127.1.14. mejores_modelos

```
model.mejores_modelos = sorted(metrics.items(), key=lambda item: item[1]['f1_score'], reverse=True) [← :3]
```

Definición en la línea 145 del archivo [model.py](#).

4.127.1.15. metrics

```
dictionary model.metrics = {}
```

Definición en la línea 113 del archivo [model.py](#).

4.127.1.16. modelos

```
dictionary model.modelos
```

Valor inicial:

```
00001 = {
00002     'RandomForest': RandomForestClassifier(random_state=42, n_jobs=-1),
00003     'LogisticRegression': LogisticRegression(max_iter=1000, random_state=42),
00004     'KNeighbors': KNeighborsClassifier(n_jobs=-1),
00005     'AdaBoost': AdaBoostClassifier(random_state=42),
00006     'XGBoost': xgb.XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss'),
00007     'LightGBM': lgb.LGBMClassifier(random_state=42, verbose=-1, force_row_wise=True)
00008 }
```

Definición en la línea 70 del archivo [model.py](#).

4.127.1.17. param_grids

```
dictionary model.param_grids
```

Valor inicial:

```
00001 = {
00002     'RandomForest': {
00003         'n_estimators': [100, 200],
00004         'max_depth': [10, 20],
00005         'min_samples_split': [2, 5],
00006         'min_samples_leaf': [1, 2]
00007     },
00008     'LogisticRegression': {
00009         'C': [0.01, 0.1, 1.0],
00010         'solver': ['liblinear', 'lbfgs']
00011     },
00012     'KNeighbors': {
00013         'n_neighbors': [3, 5, 7],
00014         'weights': ['uniform', 'distance']
00015     },
00016     'AdaBoost': {
00017         'n_estimators': [50, 100],
00018         'learning_rate': [0.01, 0.1]
00019     },
00020     'XGBoost': {
00021         'n_estimators': [100, 200],
00022         'learning_rate': [0.01, 0.1],
00023         'max_depth': [3, 5]
00024     },
00025     'LightGBM': {
00026         'n_estimators': [100, 200],
00027         'learning_rate': [0.01, 0.1],
00028         'max_depth': [3, 5]
00029     }
00030 }
```

Definición en la línea 80 del archivo [model.py](#).

4.127.1.18. poly

```
model.poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
```

Definición en la línea 65 del archivo [model.py](#).

4.127.1.19. preprocessor

```
model.preprocessor
```

Valor inicial:

```
00001 = ColumnTransformer(
00002     transformers=[
00003         ('num', Pipeline([('imputer', SimpleImputer(strategy='mean')), ('scaler', StandardScaler())]),
00004             columns_numericas),
00005         ('cat', Pipeline([('imputer', SimpleImputer(strategy='constant', fill_value='Desconocido')),
00006                         ('encoder', OneHotEncoder(handle_unknown='ignore'))]), columns_categoricas)
00007     ])
```

Definición en la línea 42 del archivo [model.py](#).

4.127.1.20. random_state

```
model.random_state
```

Definición en la línea 55 del archivo [model.py](#).

4.127.1.21. smote

```
model.smote = SMOTE(random_state=42)
```

Definición en la línea 61 del archivo [model.py](#).

4.127.1.22. stratify

```
model.stratify
```

Definición en la línea 55 del archivo [model.py](#).

4.127.1.23. test_size

```
model.test_size
```

Definición en la línea 55 del archivo [model.py](#).

4.127.1.24. total

```
model.total
```

Definición en la línea 115 del archivo [model.py](#).

4.127.1.25. voting_clf

```
model.voting_clf
```

Valor inicial:

```
00001 = VotingClassifier(  
00002     estimators=mejores_estimadores,  
00003     voting='soft',  
00004     n_jobs=-1  
00005 )
```

Definición en la línea 149 del archivo [model.py](#).

4.127.1.26. X

```
model.X = data[columnas_categoricas + columnas_numericas]
```

Definición en la línea 50 del archivo [model.py](#).

4.127.1.27. X_test

```
model.X_test = preprocessor.transform(X_test)
```

Definición en la línea 55 del archivo [model.py](#).

4.127.1.28. X_train

```
model.X_train = preprocessor.fit_transform(X_train)
```

Definición en la línea 55 del archivo [model.py](#).

4.127.1.29. y

```
model.y = data['abandona']
```

Definición en la línea 51 del archivo [model.py](#).

4.127.1.30. y_pred

```
model.y_pred = grid_search.predict(X_test)
```

Definición en la línea 122 del archivo [model.py](#).

4.127.1.31. y_pred_voting

```
model.y_pred_voting = voting_clf.predict(X_test)
```

Definición en la línea 164 del archivo [model.py](#).

4.127.1.32. `y_test`

`model.y_test`

Definición en la línea 55 del archivo [model.py](#).

4.127.1.33. `y_train`

`model.y_train`

Definición en la línea 55 del archivo [model.py](#).

4.128. Referencia del Namespace `util`

Funciones

- def [list_to_tuple](#) (value)
- def [random_color](#) (size)
- def [load_data_for_model](#) ()
- def [load_model](#) ()

Variables

- dictionary [config_mode_bar_buttons_gestor](#)

4.128.1. Documentación de las funciones

4.128.1.1. `list_to_tuple()`

```
def util.list_to_tuple ( value )
```

Convierte una lista en una tupla.

Args:

 value: Lista a convertir.

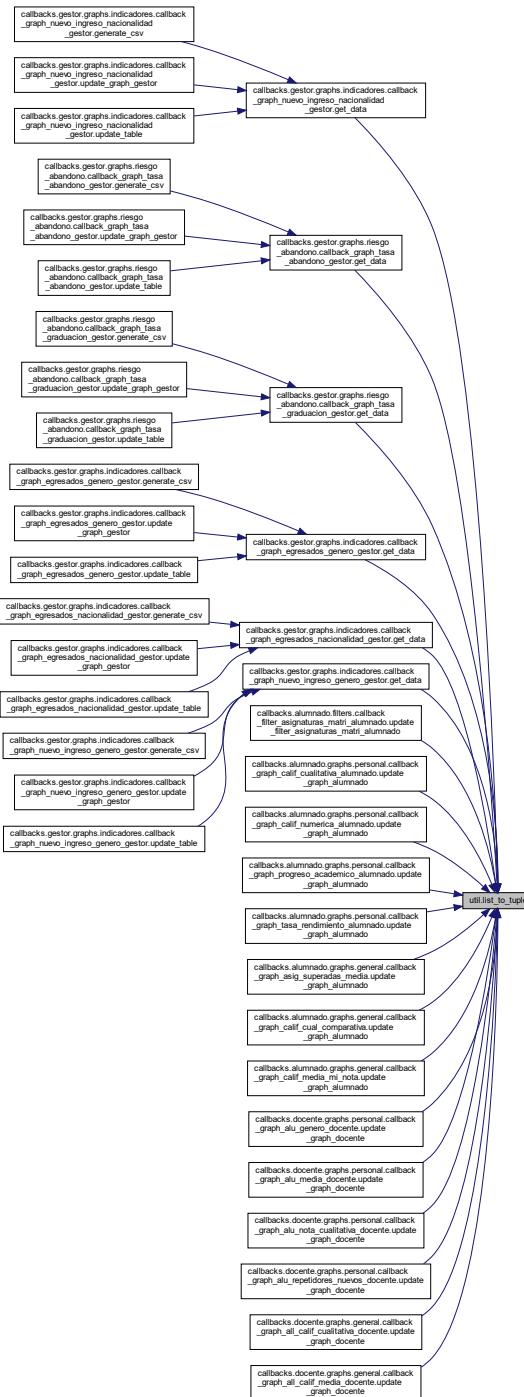
Returns:

 Tupla con los elementos de la lista.

Definición en la línea 22 del archivo [util.py](#).

```
00022 def list_to_tuple(value):
00023     """
00024     Convierte una lista en una tupla.
00025     Args:
00026         value: Lista a convertir.
00027     Returns:
00028     Tupla con los elementos de la lista.
00029     """
00030     if isinstance(value, str):
00031         return (value,)
00032     elif isinstance(value, list):
00033         return tuple(value)
00034     else:
00035         return ValueError(
00036             "Tipo de dato no soportado, se esperaba un list o str. {}".format(
00037                 type(value)
00038             )
00039         )
00040
00041
```

Gráfico de llamadas a esta función:



4.128.1.2. load_data_for_model()

```
def util.load_data_for_model( )
```

Carga los datos necesarios para entrenar el modelo.

Returns:
`pd.DataFrame`: Datos para entrenar el modelo.

Definición en la línea 70 del archivo [util.py](#).

```
00070 def load_data_for_model():
00071     """
00072 Carga los datos necesarios para entrenar el modelo.
00073
00074 Returns:
00075 pd.DataFrame: Datos para entrenar el modelo.
00076 """
00077 data = data_for_model()
00078 df = pd.DataFrame(data)
00079
00080 # Ajustar los tipos de datos de las columnas
00081     df = df.astype({
00082         'id': str,
00083         'anio_nac': int,
00084         'nacionalidad': str,
00085         'sexo': str,
00086         'titulacion': str,
00087         'nota_def_acceso': float,
00088         'nota_media': float,
00089         'abandona': str
00090     })
00091
00092     return df
00093
00094
```

Gráfico de llamadas para esta función:

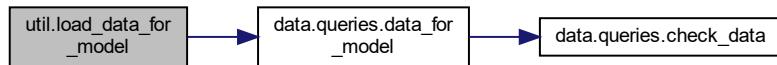
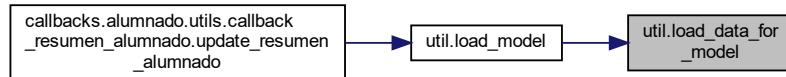


Gráfico de llamadas a esta función:



4.128.1.3. `load_model()`

```
def util.load_model( )
```

Carga el modelo entrenado.

Returns:
`Modelo` entrenado.

Definición en la línea 95 del archivo `util.py`:

```

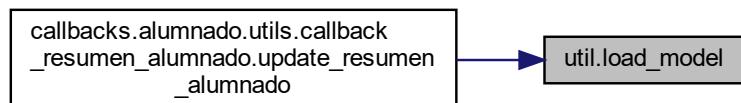
00095 def load_model():
00096     """
00097 Carga el modelo entrenado.
00098
00099 Returns:
00100 Modelo entrenado.
00101 """
00102 model_file = "src/trained_model.pkl"
00103 voting_clf = joblib.load(model_file)
00104 data = load_data_for_model()
00105
00106 current_year = datetime.now().year
00107 data['edad_actual'] = current_year - data['anio_nac']
00108
00109 # Codificar variable objetivo
00110 data['abandona'] = data['abandona'].map({'si': 1, 'no': 0})
00111
00112 # Definir columnas
00113 columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
00114 columnas_numericas = ['nota_def_acceso', 'nota_media', 'edad_actual']
00115
00116 # Imputación de datos
00117 num_imputer = SimpleImputer(strategy='mean')
00118 cat_imputer = SimpleImputer(strategy='constant', fill_value='Desconocido')
00119
00120 # Preprocesamiento
00121 preprocessor = ColumnTransformer(
00122     transformers=[
00123         ('num', Pipeline([('imputer', num_imputer), ('scaler', StandardScaler())]),
00124         columnas_numericas),
00125         ('cat', Pipeline([('imputer', cat_imputer), ('encoder',
00126             OneHotEncoder(handle_unknown='ignore')])]), columnas_categoricas)
00127     ]
00128 )
00129
00130 # Dividir datos en características y variable objetivo
00131 X = data[columnas_categoricas + columnas_numericas]
00132 y = data['abandona']
00133 ids = data['id']
00134
00135 X_non_abandon = X[y == 0]
00136 data_non_abandon = data[y == 0].copy()
00137 X_non_abandon_preprocessed = preprocessor.fit_transform(X_non_abandon)
00138 poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
00139 X_non_abandon_poly = poly.fit_transform(X_non_abandon_preprocessed)
00140
00141 # Predecir probabilidades de abandono
00142 probabilidades_abandono = voting_clf.predict_proba(X_non_abandon_poly)[:, 1]
00143 data_non_abandon['probabilidad_abandono'] = probabilidades_abandono
00144
00145 return data_non_abandon

```

Gráfico de llamadas para esta función:



Gráfico de llamadas a esta función:



4.128.1.4. random_color()

```
def util.random_color (
    size )
```

Genera una lista de colores aleatorios en formato hexadecimal.

Args:

size: Número de colores a generar.

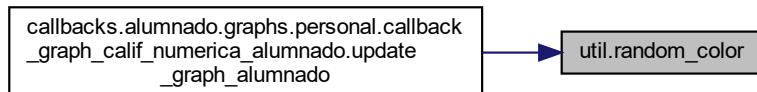
Returns:

Lista de colores aleatorios.

Definición en la línea 42 del archivo [util.py](#).

```
00042 def random_color(size):
00043     """
00044     Genera una lista de colores aleatorios en formato hexadecimal.
00045
00046 Args:
00047     size: Número de colores a generar.
00048 Returns:
00049     Lista de colores aleatorios.
00050 """
00051     return ["#%06X" % random.randint(0, 0xFFFFFF) for _ in range(size)]
00052
00053
00054 # Configuración de los botones de la barra de herramientas de Plotly
```

Gráfico de llamadas a esta función:



4.128.2. Documentación de las variables

4.128.2.1. config_mode_bar_buttons_gestor

```
dictionary util.config_mode_bar_buttons_gestor
```

Valor inicial:

```
00001 = {
00002     "displayModeBar": True,
00003     "displaylogo": False,
00004     "scrollZoom": True,
00005     "modeBarButtonsToRemove": ["zoom2d", "lasso2d", "resetScale2d"],
00006     "modeBarButtonsToAdd": [
00007         "drawline",
00008         "drawcircle",
00009         "drawrect",
00010         "eraseshape",
00011         "toggleSpikelines",
00012     ],
00013 }
```

Definición en la línea 55 del archivo [util.py](#).

Capítulo 5

Documentación de las clases

5.1. Referencia de la Clase `data.db_connector.DatabaseConnector`

Métodos públicos

- def `__init__` (self, dbname, user, password, host, port, pool_size=5, max_overflow=10)
- def `execute_query` (self, query, params=None)
- def `close` (self)

Atributos públicos

- `db_url`
- `engine`

5.1.1. Descripción detallada

Clase para conectarse a la base de datos

Attributes:

```
db_url (str): URL de la base de datos
engine (Engine): Motor de la base de datos
```

Methods:

```
execute_query: Ejecuta una consulta en la base de datos
close: Cierra la conexión a la base de datos
```

Definición en la línea 17 del archivo [db_connector.py](#).

5.1.2. Documentación del constructor y destructor

5.1.2.1. `__init__()`

```
def data.db_connector.DatabaseConnector.__init__ (
    self,
    dbname,
    user,
    password,
    host,
    port,
    pool_size = 5,
    max_overflow = 10 )
```

Definición en la línea 30 del archivo [db_connector.py](#).

```
00032     ):
00033         self.db_url = f"postgresql+psycopg2://{user}:{password}@{host}:{port}/{dbname}"
00034         self.engine = create_engine(
00035             self.db_url,
00036             poolclass=QueuePool,
00037             pool_size=pool_size,
00038             max_overflow=max_overflow,
00039         )
00040
```

5.1.3. Documentación de las funciones miembro

5.1.3.1. `close()`

```
def data.db_connector.DatabaseConnector.close (
    self )
```

Cierra la conexión a la base de datos

Definición en la línea 56 del archivo [db_connector.py](#).

```
00056     def close(self):
00057         """
00058 Cierra la conexión a la base de datos
00059
00060 """
00061     self.engine.dispose()
00062
00063
```

5.1.3.2. `execute_query()`

```
def data.db_connector.DatabaseConnector.execute_query (
    self,
    query,
    params = None )
```

Ejecuta una consulta en la base de datos

Args:

```
    query (str): Consulta SQL
    params (dict): Parámetros de la consulta
```

Returns:

```
    list: Resultado de la consulta
```

Definición en la línea 41 del archivo [db_connector.py](#).

```
00041     def execute_query(self, query, params=None):
00042         """
00043     Ejecuta una consulta en la base de datos
00044
00045     Args:
00046         query (str): Consulta SQL
00047         params (dict): Parámetros de la consulta
00048
00049     Returns:
00050     list: Resultado de la consulta
00051     """
00052     with self.engine.connect() as connection:
00053         result = connection.execute(text(query), params or {})
00054         return result.fetchall()
00055
```

5.1.4. Documentación de los datos miembro

5.1.4.1. db_url

```
data.db_connector.DatabaseConnector.db_url
```

Definición en la línea 33 del archivo [db_connector.py](#).

5.1.4.2. engine

```
data.db_connector.DatabaseConnector.engine
```

Definición en la línea 34 del archivo [db_connector.py](#).

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/[db_connector.py](#)

Capítulo 6

Documentación de archivos

6.1. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/app.py

Namespaces

- namespace app

Variables

- app.app
- app.title
- app._favicon
- app.server = app.server
- app.layout
- app.debug

6.2. app.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file app.py
00003 # @brief Este archivo contiene el código principal de la aplicación.
00004 # @details Se define la estructura de la aplicación y se inicia el servidor.
00005 # @version 1.0
00006 # @date 06/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import dash, html
00013 import atexit
00014 import dash_bootstrap_components as dbc
00015 import components.common.header as header
00016 import components.common.footer as footer
00017 from layouts.alumno_layout import alumno_layout
00018 from layouts.docente_layout import docente_layout
00019 from layouts.gestor_layout import gestor_layout
00020 from callbacks.common.callback_update_layout import update_layout
00021 from data.db_connector import db
00022
00023
```

```
00024 app = dash.Dash(  
00025     __name__,  
00026     external_stylesheets=[  
00027         'src/assets/css/styles.css',  
00028         dbc.themes.BOOTSTRAP  
00029     ],  
00030     suppress_callback_exceptions=True  
00031 )  
00032  
00033 app.title = 'Dashboard académico'  
00034 app._favicon = 'src/assets/images/favicon.ico'  
00035 server = app.server  
00036  
00037 app.layout = html.Div([  
00038     header.header('store-role'),  
00039     html.Div(id='page-content'),  
00040     footer.footer(),  
00041 ])  
00042  
00043 #Cierre de la conexión a la base de datos  
00044 atexit.register(db.close)  
00045  
00046 # Iniciar el servidor  
00047 if __name__ == '__main__':  
00048     app.run_server(debug=False)
```

6.3. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/__init__.py

6.4. __init__.py

[Ir a la documentación de este archivo.](#)

6.5. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/__init__.py

6.6. __init__.py

[Ir a la documentación de este archivo.](#)

6.7. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/__init__.py

6.8. __init__.py

[Ir a la documentación de este archivo.](#)

6.9. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_ ↵
Universities_TFG/src/callbacks/alumnado/graphs/__init__.py

6.10. __init__.py

[Ir a la documentación de este archivo.](#)

6.11. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_ ↵ TFG/src/callbacks/alumnado/graphs/general/__init__.py

6.12. __init__.py

[Ir a la documentación de este archivo.](#)

6.13. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_ ↵ TFG/src/callbacks/alumnado/graphs/personal/__init__.py

6.14. __init__.py

[Ir a la documentación de este archivo.](#)

6.15. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_ ↵
Universities_TFG/src/callbacks/alumnado/utils/__init__.py

6.16. __init__.py

[Ir a la documentación de este archivo.](#)

6.17. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_ ↵
Universities_TFG/src/callbacks/common/__init__.py

6.18. __init__.py

[Ir a la documentación de este archivo.](#)

6.19. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_ Universities_TFG/src/callbacks/docente/__init__.py

6.20. __init__.py

[Ir a la documentación de este archivo.](#)

6.21. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_ Universities_TFG/src/callbacks/docente/filters/__init__.py

6.22. __init__.py

[Ir a la documentación de este archivo.](#)

6.23. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_ Universities_TFG/src/callbacks/docente/graphs/__init__.py

6.24. __init__.py

[Ir a la documentación de este archivo.](#)

6.25. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_ Universities_TFG/src/callbacks/docente/graphs/general/__init__.py

6.26. __init__.py

[Ir a la documentación de este archivo.](#)

6.27. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/ Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/__init__.py**6.28. __init__.py**

[Ir a la documentación de este archivo.](#)

6.29. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_↔
Universities_TFG/src/callbacks/docente/utils/__init__.py

6.30. __init__.py

[Ir a la documentación de este archivo.](#)

6.31. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_↔
Universities_TFG/src/callbacks/gestor/__init__.py

6.32. __init__.py

[Ir a la documentación de este archivo.](#)

6.33. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_↔
Universities_TFG/src/callbacks/gestor/filters/__init__.py

6.34. __init__.py

[Ir a la documentación de este archivo.](#)

6.35. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_↔
Universities_TFG/src/callbacks/gestor/graphs/__init__.py

6.36. __init__.py

[Ir a la documentación de este archivo.](#)

6.37. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↔ Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/indicadores/__init__.py

6.38. __init__.py

[Ir a la documentación de este archivo.](#)

6.39. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/__init__.py

6.40. __init__.py

[Ir a la documentación de este archivo.](#)

6.41. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo_abandono/__init__.py

6.42. __init__.py

[Ir a la documentación de este archivo.](#)

**6.43. Referencia del Archivo
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/__init__.py**

6.44. __init__.py

[Ir a la documentación de este archivo.](#)

**6.45. Referencia del Archivo
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/__init__.py**

6.46. __init__.py

[Ir a la documentación de este archivo.](#)

**6.47. Referencia del Archivo
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/__init__.py**

6.48. __init__.py

[Ir a la documentación de este archivo.](#)

6.49. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/alumnado/filters/__init__.py

6.50. __init__.py

[Ir a la documentación de este archivo.](#)

6.51. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/alumnado/graphs/__init__.py

6.52. __init__.py

[Ir a la documentación de este archivo.](#)

6.53. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/alumnado/utils/__init__.py

6.54. __init__.py

[Ir a la documentación de este archivo.](#)

6.55. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/common/__init__.py

6.56. __init__.py

[Ir a la documentación de este archivo.](#)

6.57. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/docente/__init__.py

6.58. __init__.py

[Ir a la documentación de este archivo.](#)

6.59. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/docente/filters/__init__.py**

6.60. __init__.py

[Ir a la documentación de este archivo.](#)

6.61. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/docente/graphs/__init__.py**

6.62. __init__.py

[Ir a la documentación de este archivo.](#)

6.63. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/docente/utils/__init__.py**

6.64. __init__.py

[Ir a la documentación de este archivo.](#)

6.65. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/gestor/__init__.py**

6.66. __init__.py

[Ir a la documentación de este archivo.](#)

6.67. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/gestor/filters/__init__.py**

6.68. __init__.py

[Ir a la documentación de este archivo.](#)

6.69. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/__init__.py

6.70. __init__.py

[Ir a la documentación de este archivo.](#)

6.71. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/__init__.py

6.72. __init__.py

[Ir a la documentación de este archivo.](#)

6.73. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/data/__init__.py

6.74. __init__.py

[Ir a la documentación de este archivo.](#)

6.75. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/layouts/__init__.py

6.76. __init__.py

[Ir a la documentación de este archivo.](#)

6.77. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/callback_filter_asignaturas_matri_alumnado.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.alumnado](#)
- namespace [callbacks.alumnado.filters](#)
- namespace [callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado](#)

Funciones

- def callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado.update_filter_asignaturas_matri_alumnado(alumno_id, curso_academico, titulacion, n_clicks, existing_options)

6.78. callback_filter_asignaturas_matri_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_filter_asignaturas_matri_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar las asignaturas matriculadas por el alumno
00004 # @version 1.0
00005 # @date 07/05/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martin
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import Input, Output, State, callback, callback_context
00012 from data.queries import asignaturas_matriculadas
00013 from util import list_to_tuple
00014
00015 @callback(
00016     Output('asignaturas-matriculadas', 'options'),
00017     Output('asignaturas-matriculadas', 'value'),
00018     Input('selected-alumnado-store', 'data'),
00019     Input('curso-academico', 'value'),
00020     Input('titulacion-alumnado', 'value'),
00021     Input('select-all-button', 'n_clicks'),
00022     State('asignaturas-matriculadas', 'options')
00023 )
00024 def update_filter_asignaturas_matri_alumnado(alumno_id, curso_academico, titulacion, n_clicks,
existing_options):
00025 """
00026 Actualiza las opciones del dropdown de asignaturas matriculadas por el alumno y
00027 gestiona el evento del botón "Seleccionar todo".
00028
00029     Args:
00030         alumno_id (str): Identificador del alumno.
00031         curso_academico (list): Lista con los cursos académicos
00032         titulacion (str): Titulación seleccionada
00033         n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00034         existing_options (list): Opciones actuales del dropdown
00035
00036     Returns:
00037         list: Opciones del dropdown
00038         list: Valor seleccionado
00039 """
00040
00041 ctx = callback_context
00042 trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None
00043
00044     # Evento de selección de todas las asignaturas matriculadas
00045 if trigger_id == 'select-all-button' and n_clicks > 0:
00046     return existing_options, [option['value'] for option in existing_options]
00047
00048 if not (alumno_id and curso_academico and titulacion):
00049     return [], None
00050
00051 try:
00052     curso_academico = list_to_tuple(curso_academico)
00053 except Exception as e:
00054     return [], None
00055
00056 data = asignaturas_matriculadas(alumno_id, curso_academico, titulacion)
00057
00058 if not data:
00059     return [], None
00060
00061 opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
00062 value = [option['value'] for option in opciones_dropdown]
00063
00064 return opciones_dropdown, value

```

6.79. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_↔
Universities_TFG/src/callbacks/alumnado/filters/callback_filter_↔
curso_cademico_alumnado.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.filters
- namespace callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado

Funciones

- def callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado.update_filter_curso_academico_alumnado(alumno_id, titulacion, n_clicks, existing_options)

6.80. callback_filter_curso_cademico_alumnado.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file callback_filter_curso_cademico_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar el filtro de curso académico del alumnado
00004 # @version 1.0
00005 # @date 05/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import Input, Output, State, callback, callback_context
00012 from callbacks.alumnado.utils.callback_select_alumnado import store_selected_alumnado
00013 from data.queries import curso_academico_alumnado
00014
00015 @callback(
00016     Output('curso-academico', 'options'),
00017     Output('curso-academico', 'value'),
00018     Input('selected-alumnado-store', 'data'),
00019     Input('titulacion-alumnado', 'value'),
00020     Input('select-all-cursos-academicos', 'n_clicks'),
00021     State('curso-academico', 'options')
00022 )
00023 def update_filter_curso_academico_alumnado(alumno_id, titulacion, n_clicks, existing_options):
00024 """
00025 Actualiza las opciones del dropdown de cursos académicos y
00026 gestiona el evento del botón "Seleccionar todo".
00027
00028     Args:
00029         alumno_id (str): Identificador del alumno.
00030         titulacion (str): Titulación seleccionada
00031         n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00032         existing_options (list): Opciones actuales del dropdown
00033
00034     Returns:
00035         list: Opciones del dropdown
00036         list: Valor seleccionado
00037 """
00038
00039 ctx = callback_context
00040 trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None
00041
00042     # Evento de selección de todos los cursos académicos
00043     if trigger_id == 'select-all-cursos-academicos' and n_clicks > 0:
00044         return existing_options, [option['value'] for option in existing_options]
00045
00046     if not (alumno_id and titulacion):
00047         return [], None
00048
```

```

00049 # Obtener los cursos académicos desde la base de datos
00050 data = curso_academico_alumnado(alumno_id, titulacion)
00051
00052 if not data:
00053     return [], None
00054
00055 opciones_dropdown = [{'label': curso[0], 'value': curso[0]} for curso in data]
00056 value = [option['value'] for option in opciones_dropdown]
00057
00058 return opciones_dropdown, value
00059

```

6.81. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/alumnado/filters/callback_filter_←
titulacion_alumnado.py**

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.filters
- namespace callbacks.alumnado.filters.callback_filter_titulacion_alumnado

Funciones

- def callbacks.alumnado.filters.callback_filter_titulacion_alumnado.update_filter_titulacion_alumnado(alumno_id, selected_value, stored_titulacion)

6.82. callback_filter_titulacion_alumnado.py

Ir a la documentación de este archivo.

```

00001 #
00002 # @file callback_filter_titulacion_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar las titulaciones del alumnado
00004 # @version 1.0
00005 # @date 18/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import Input, Output, State, callback
00012 from data.queries import titulacion_alumnado
00013
00014 @callback(
00015     Output('titulacion-alumnado', 'options'),
00016     Output('titulacion-alumnado', 'value'),
00017     Output('selected-titulacion-alumnado-store', 'data'),
00018     Input('selected-alumnado-store', 'data'),
00019     Input('titulacion-alumnado', 'value'),
00020     State('selected-titulacion-alumnado-store', 'data')
00021 )
00022 def update_filter_titulacion_alumnado(alumno_id, selected_value, stored_titulacion):
00023     """
00024     Actualiza las opciones del dropdown de titulaciones del perfil "Alumno".
00025
00026     Args:
00027         alumno_id (str): Identificador del alumno.
00028         selected_value (str): Valor seleccionado en el dropdown
00029         stored_titulacion (str): Titulación almacenada
00030
00031     Returns:
00032         list: Opciones del dropdown

```

```
00033     str: Valor seleccionado
00034 """
00035 if not alumno_id:
00036     return [], None, None
00037
00038     data = titulacion_alumnado(alumno_id)
00039
00040     if not data:
00041         return [], None, None
00042
00043     opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
00044
00045 # Si no hay valor seleccionado pero hay un valor almacenado, lo usamos si está en las opciones
00046 if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
00047     return opciones_dropdown, stored_titulacion, stored_titulacion
00048
00049 return opciones_dropdown, selected_value, selected_value
```

6.83. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/graphs/general/callback_graph_asig_superadas_media.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.graphs
- namespace callbacks.alumnado.graphs.general
- namespace callbacks.alumnado.graphs.general.callback_graph_asig_superadas_media

Funciones

- def callbacks.alumnado.graphs.general.callback_graph_asig_superadas_media.update_graph_alumnado(curso_academico, alumno_id, asignaturas_matriculadas, titulacion)

6.84. callback_graph_asig_superadas_media.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_graph_asig_superadas_media.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 # de relación entre la nota media y el número de asignaturas superadas
00005 # por alumno del perfil "Alumno" de la pestaña "Rendimiento académico general".
00006 # @version 1.0
00007 # @date 09/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import callback, Input, Output
00014 import plotly.graph_objs as go
00015 import pandas as pd
00016 from data.queries import asignaturas_superadas_media_abandono, universidad_alumno
00017 from util import list_to_tuple
00018
00019
00020 @callback(
00021     Output("asignaturas-superadas-general-mi-nota", "figure"),
00022     Input("curso-academico", "value"),
00023     Input("selected-alumnado-store", "data"),
00024     Input("asignaturas-matriculadas", "value"),
```

```

00025     Input("titulacion-alumnado", "value"),
00026 )
00027 def update_graph_alumnado(curso_academico, alumno_id, asignaturas_matriculadas, titulacion):
00028 """
00029 Actualiza el gráfico de relación entre la nota media y el número de asignaturas superadas
00030 por alumno del perfil "Alumno" de la pestaña "Rendimiento académico general".
00031
00032 Args:
00033     curso_academico (list): Lista con los cursos académicos
00034     alumno_id (str): Identificador del alumno
00035     asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00036     titulacion (str): Titulación seleccionada
00037
00038 Returns:
00039     go.Figure: Figura con el gráfico
00040
00041 """
00042 fig = go.Figure()
00043
00044 fig.update_layout(
00045 title={
00046 "text": "Relación nota media y número de asignaturas superadas por alumno",
00047         "x": 0.5,
00048 },
00049     xaxis_title="Nota Media",
00050     yaxis_title="Nº Asignaturas superadas",
00051     legend_title="Estado del alumno",
00052     showlegend=True,
00053     xaxis=dict(range=[0, 10]),
00054     yaxis=dict(range=[0, 40]),
00055 )
00056
00057 if not (curso_academico and alumno_id and asignaturas_matriculadas and titulacion):
00058     return fig
00059
00060 try:
00061     curso_academico = list_to_tuple(curso_academico)
00062     asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00063 except Exception as e:
00064     print("Error:", e)
00065     return fig
00066
00067 data_universidad = universidad_alumno(alumno_id)
00068
00069 if not data_universidad:
00070     return fig
00071
00072 data = asignaturas_superadas_media_abandono(
00073     curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
00074 )
00075
00076 if not data:
00077     return fig
00078
00079 df = pd.DataFrame(
00080     data, columns=["Alumno_id", "Abandono", "Nota_Media", "Asignaturas_Superadas"]
00081 )
00082
00083 df["Abandono"] = (
00084     df["Abandono"]
00085     .str.strip()
00086     .str.lower()
00087     .replace({"si": "Abandona", "no": "No abandona"})
00088 )
00089 df["Personal"] = df["Alumno_id"].apply(lambda x: " (Yo)" if x == alumno_id else "")
00090 df["Key"] = df["Abandono"] + df["Personal"]
00091
00092 colors = {
00093     "Abandona": "red",
00094     "No abandona": "blue",
00095     "Abandona (Yo)": "yellow",
00096     "No abandona (Yo)": "yellow",
00097 }
00098
00099 for key, group in df.groupby("Key"):
00100     fig.add_trace(
00101         go.Scatter(
00102             x=group["Nota_Media"],
00103             y=group["Asignaturas_Superadas"],
00104             mode="markers",
00105             name=key,
00106             marker=dict(
00107                 size=12,
00108                 line=dict(width=2 if " (Yo)" in key else 1),
00109                 color=colors[key],
00110             ),
00111             opacity=1.0 if " (Yo)" in key else 0.8,

```

```
00112         )
00113     )
00114
00115     return fig
```

6.85. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/alumnado/graphs/general/callback_graph_calif_cual_comparativa.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.graphs
- namespace callbacks.alumnado.graphs.general
- namespace callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa

Funciones

- def callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa.update_graph_alumnado(curso_academico, asignaturas_matriculadas, alumno_id, titulacion)

6.86. callback_graph_calif_cual_comparativa.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_graph_calif_cual_comparativa.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de calificaciones
00004 #       cualitativas general por curso académico del perfil "Alumno" de la pestaña
00005 #       "Rendimiento académico general".
00006 # @version 1.0
00007 # @date 07/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import callback, Input, Output
00014 import plotly.graph_objs as go
00015 import pandas as pd
00016 from data.queries import (
00017     calif_cualitativa_comparativa,
00018     calif_cualitativa_alumno_asignaturas,
00019     universidad_alumno,
00020 )
00021 from util import list_to_tuple
00022
00023
00024 @callback(
00025     Output("nota-cualitativa-general-mi-nota", "figure"),
00026     Input("curso-academico", "value"),
00027     Input("asignaturas-matriculadas", "value"),
00028     Input("selected-alumnado-store", "data"),
00029     Input("titulacion-alumnado", "value"),
00030 )
00031 def update_graph_alumnado(curso_academico, asignaturas_matriculadas, alumno_id, titulacion):
00032     """
00033     Actualiza el gráfico de calificaciones cualitativas general por curso académico
00034     del perfil "Alumno" de la pestaña "Rendimiento académico general".
00035
00036     Args:
00037         curso_academico (list): Lista con los cursos académicos
```

```

00038     asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00039     alumno_id (str): Identificador del alumno
00040     titulacion (str): Titulación seleccionada
00041
00042     Returns:
00043         go.Figure: Figura con el gráfico
00044     """
00045 fig = go.Figure()
00046
00047 fig.update_layout(
00048     title={
00049         "text": "Calificaciones cualitativas general por curso académico",
00050         "x": 0.5,
00051     },
00052     barmode="stack",
00053     xaxis={"title": "Asignatura", "tickangle": 45},
00054     yaxis={"title": "Nº Alumnos matriculados"},
00055     showlegend=True,
00056     legend={"title": "Calificación"},
00057     height=600,
00058 )
00059
00060 if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
00061     return fig
00062
00063 try:
00064     curso_academico = list_to_tuple(curso_academico)
00065     asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00066 except Exception as e:
00067     print("Error:", e)
00068     return fig
00069
00070 data_universidad = universidad_alumno(alumno_id)
00071
00072 if not data_universidad:
00073     return fig
00074
00075 general_data = calif_cualitativa_comparativa(
00076     curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
00077 )
00078 student_data = calif_cualitativa_alumno_asignaturas(
00079     alumno_id, curso_academico, asignaturas_matriculadas, titulacion
00080 )
00081
00082 if not general_data:
00083     return fig
00084
00085 general_df = pd.DataFrame(
00086     general_data, columns=["Asignatura", "Calificacion", "Numero"]
00087 )
00088 student_df = pd.DataFrame(
00089     student_data, columns=["Asignatura", "Calificacion", "Numero"]
00090 )
00091
00092 categories = ["Suspensos", "No presentado", "Aprobado", "Notable", "Sobresaliente"]
00093
00094 color_mapping = {
00095     "Sobresaliente": "blue",
00096     "Notable": "green",
00097     "Aprobado": "orange",
00098     "Suspensos": "red",
00099     "No presentado": "gray",
00100 }
00101
00102 general_pivot = general_df.pivot_table(
00103     index="Asignatura", columns="Calificacion", values="Numero", fill_value=0
00104 )
00105 general_pivot = general_pivot.reindex(columns=categories, fill_value=0)
00106
00107 for category in categories:
00108     fig.add_trace(
00109         go.Bar(
00110             x=general_pivot.index,
00111             y=general_pivot[category],
00112             name=category,
00113             marker=dict(color=color_mapping[category]),
00114             opacity=0.7,
00115         )
00116     )
00117
00118 for category in categories:
00119     y = [
00120         1 if (
00121             (subject in student_df['Asignatura'].values) and
00122             (student_df[student_df['Asignatura'] == subject]['Calificacion'].values[0] ==
00123             category)
00124         ) else 0 for subject in general_pivot.index

```

```
00124         ]
00125     fig.add_trace(
00126         go.Bar(
00127             x=general_pivot.index,
00128             y=y,
00129             name=f"{category} (Yo)",
00130             marker=dict(color=color_mapping[category], line=dict(color='black', width=2)),
00131             opacity=1
00132         )
00133     )
00134
00135 return fig
```

6.87. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/graphs/general/callback_graph_calif_media_mi_nota.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.graphs
- namespace callbacks.alumnado.graphs.general
- namespace callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_nota

Funciones

- def callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_nota.update_graph_alumnado(curso_academico, asignaturas_matriculadas, alumno_id, titulacion)

6.88. callback_graph_calif_media_mi_nota.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_graph_calif_media_mi_nota.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #       de relación calificaciones del alumno con nota media general
00005 #       del perfil "Alumno" de la pestaña "Rendimiento académico general".
00006 # @version 1.0
00007 # @date 08/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu01138589@ull.edu.es
00011 #
00012
00013 from dash import callback, Input, Output
00014 import pandas as pd
00015 import plotly.graph_objs as go
00016 from data.queries import nota_media_general_mi_nota, universidad_alumno
00017 from util import list_to_tuple
00018
00019
00020 @callback(
00021     Output("nota-media-general-mi-nota", "figure"),
00022     Input("curso-academico", "value"),
00023     Input("asignaturas-matriculadas", "value"),
00024     Input("selected-alumnado-store", "data"),
00025     Input("titulacion-alumnado", "value"),
00026 )
00027 def update_graph_alumnado(curso_academico, asignaturas_matriculadas, alumno_id, titulacion):
00028     """
00029 Actualiza el gráfico de relación calificaciones del alumno con nota media general
```

```

00030 del perfil "Alumno" de la pestaña "Rendimiento académico general".
00031
00032     Args:
00033         curso_academico (list): Lista con los cursos académicos
00034         asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00035         alumno_id (str): Identificador del alumno
00036         titulacion (str): Titulación seleccionada
00037
00038     Returns:
00039         go.Figure: Figura con el gráfico
00040
00041     """
00042 fig = go.Figure()
00043
00044 fig.update_layout(
00045     title={
00046         "text": "Relación calificaciones del alumno con nota media general",
00047             "x": 0.5,
00048         },
00049         xaxis={"title": "Asignatura", "tickangle": 45},
00050         yaxis={"title": "Nota"},
00051         barmode="group",
00052         height=600,
00053     )
00054
00055     if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
00056         return fig
00057
00058     try:
00059         curso_academico = list_to_tuple(curso_academico)
00060         asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00061     except Exception as e:
00062         print("Error:", e)
00063         return fig
00064
00065     data_universidad = universidad_alumno(alumno_id)
00066
00067     if not data_universidad:
00068         return fig
00069
00070     data = nota_media_general_mi_nota(
00071         curso_academico,
00072         asignaturas_matriculadas,
00073         alumno_id,
00074         titulacion,
00075         data_universidad[0][0],
00076     )
00077
00078     if not data:
00079         return fig
00080
00081 df = pd.DataFrame(data, columns=["Asignatura", "NotaMediaGeneral", "MiNota"])
00082
00083 fig.add_trace(
00084     go.Bar(
00085         x=df["Asignatura"],
00086         y=df["MiNota"],
00087         name="Mi nota",
00088         marker_color="blue",
00089         opacity=0.7,
00090     )
00091 )
00092
00093 fig.add_trace(
00094     go.Bar(
00095         x=df["Asignatura"],
00096         y=df["NotaMediaGeneral"],
00097         name="Nota media general",
00098         marker_color="grey",
00099         opacity=0.7,
00100     )
00101 )
00102
00103 return fig

```

6.89. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/graphs/personal/callback_graph_calif_cualitativa_alumnado.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.graphs
- namespace callbacks.alumnado.graphs.personal
- namespace callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado

Funciones

- def callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado.update_graph_alumnado(alumno_id, curso_academico, titulacion)

6.90. callback_graph_calif_cualitativa_alumnado.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_graph_calif_cualitativa_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de calificaciones
00004 # cualitativas de las asignaturas matriculadas del perfil "Alumno" de la pestaña
00005 # "Expediente académico personal".
00006 # @version 1.0
00007 # @date 05/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, callback
00014 import plotly.graph_objs as go
00015 import pandas as pd
00016 from data.queries import calif_cualitativa_asignatura
00017 from util import list_to_tuple
00018
00019
00020 @callback(
00021     Output("graph-bar-evolucion-asignaturas-matriculadas", "figure"),
00022     Input("selected-alumnado-store", "data"),
00023     Input("curso-academico", "value"),
00024     Input("titulacion-alumnado", "value"),
00025 )
00026 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00027 """
00028 Actualiza el gráfico de calificaciones cualitativas de las asignaturas matriculadas
00029 del perfil "Alumno" de la pestaña "Expediente académico personal".
00030
00031     Args:
00032         alumno_id (str): Identificador del alumno.
00033         curso_academico (list): Lista con los cursos académicos
00034         titulacion (str): Titulación seleccionada
00035
00036     Returns:
00037         go.Figure: Figura con el gráfico
00038 """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     title={"text": "Calificaciones cualitativas", "x": 0.5},
00044     barmode="stack",
00045     xaxis={"title": "Curso académico"},
00046     yaxis={"title": "Nº Asignaturas matriculadas"},
```

```

00047     showlegend=True,
00048     legend={"title": "Calificación"},
00049 )
00050
00051 if not (alumno_id and curso_academico and titulacion):
00052     return fig
00053
00054 try:
00055     curso_academico = list_to_tuple(curso_academico)
00056 except Exception as e:
00057     print("Error:", e)
00058     return fig
00059
00060 data = calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion)
00061
00062 if not data:
00063     return fig
00064
00065 df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "cantidad"])
00066 df_pivot = df.pivot_table(
00067     index="curso_academico", columns="calificacion", values="cantidad", fill_value=0
00068 )
00069
00070 categories = ["No presentado", "Suspensos", "Aprobados", "Notable", "Sobresaliente"]
00071 for category in categories:
00072     if category not in df_pivot.columns:
00073         df_pivot[category] = 0
00074
00075 df_pivot = df_pivot[categories]
00076 df_pivot = df_pivot.sort_index()
00077
00078 color_mapping = {
00079     "Sobresaliente": "blue",
00080     "Notable": "green",
00081     "Aprobados": "orange",
00082     "Suspensos": "red",
00083     "No presentado": "gray",
00084 }
00085
00086 for category in categories:
00087     fig.add_trace(
00088         go.Bar(
00089             x=df_pivot.index,
00090             y=df_pivot[category],
00091             name=category,
00092             marker_color=color_mapping[category],
00093             opacity=0.7,
00094         )
00095     )
00096
00097 return fig

```

6.91. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_graph_calif_numerica_alumnado.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.graphs
- namespace callbacks.alumnado.graphs.personal
- namespace callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_alumnado

Funciones

- def callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_alumnado.update_graph_alumnado(alumno_id, curso_academico, titulacion)

6.92. callback_graph_calif_numerica_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_calif_numerica_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de calificaciones
00004 #      cuantitativas de las asignaturas matriculadas del perfil "Alumno" de la pestaña
00005 #      "Expediente académico personal".
00006 # @version 1.0
00007 # @date 06/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martin
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, callback
00014 import plotly.graph_objs as go
00015 import pandas as pd
00016 from data.queries import calif_numerica_asignatura
00017 from util import list_to_tuple, random_color
00018
00019
00020 @callback(
00021     Output("graph-bar-calificaciones-por-asignatura", "figure"),
00022     Input("selected-alumnado-store", "data"),
00023     Input("curso-academico", "value"),
00024     Input("titulacion-alumnado", "value"),
00025 )
00026 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00027 """
00028 Actualiza el gráfico de calificaciones cuantitativas de las asignaturas matriculadas
00029 del perfil "Alumno" de la pestaña "Expediente académico personal".
00030
00031     Args:
00032         alumno_id (str): Identificador del alumno.
00033         curso_academico (list): Lista con los cursos académicos
00034         titulacion (str): Titulación seleccionada
00035
00036     Returns:
00037         go.Figure: Figura con el gráfico
00038 """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     title={"text": "Calificaciones cuantitativas", "x": 0.5},
00044     xaxis={"title": "Asignatura", "tickangle": 45},
00045     yaxis={"title": "Calificación"},
00046     height=600,
00047     showlegend=True,
00048     legend=dict(
00049         x=1,
00050         y=1,
00051         traceorder="normal",
00052         font=dict(
00053             size=10,
00054         ),
00055         title="Asignaturas",
00056     ),
00057 )
00058
00059 if not (alumno_id and curso_academico and titulacion):
00060     return fig
00061
00062 try:
00063     curso_academico = list_to_tuple(curso_academico)
00064 except Exception as e:
00065     print("Error:", e)
00066     return fig
00067
00068 data = calif_numerica_asignatura(alumno_id, curso_academico, titulacion)
00069
00070 if not data:
00071     return fig
00072
00073 df = pd.DataFrame(data, columns=["Asignatura", "Calificacion"])
00074 colors = random_color(len(df))
00075
00076 for index, row in df.iterrows():
00077     fig.add_trace(
00078         go.Bar(
00079             x=[row["Asignatura"]],
00080             y=[row["Calificacion"]],
00081             name=row["Asignatura"],
00082             marker_color=colors[index],

```

```

00083         opacity=0.7,
00084     )
00085   )
00086 return fig

```

6.93. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_graph_progreso_academico_alumnado.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.graphs
- namespace callbacks.alumnado.graphs.personal
- namespace callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado

Funciones

- def callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado.update_graph_alumnado(alumno_id, curso_academico, titulacion)

6.94. callback_graph_progreso_academico_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_progreso_academico_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #       de evolución del progreso académico del perfil "Alumno" de la
00005 #       pestaña "Expediente académico personal".
00006 # @version 1.0
00007 # @date 06/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, callback
00014 import plotly.graph_objs as go
00015 import pandas as pd
00016 from data.queries import asignaturas_superadas
00017 from util import list_to_tuple
00018
00019
00020 @callback(
00021     Output("graph-evolucion-progreso-academico", "figure"),
00022     Input("selected-alumnado-store", "data"),
00023     Input("curso-academico", "value"),
00024     Input("titulacion-alumnado", "value"),
00025 )
00026 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00027 """
00028 Actualiza el gráfico de evolución del progreso académico del perfil "Alumno"
00029 de la pestaña "Expediente académico personal".
00030
00031     Args:
00032         alumno_id (str): Identificador del alumno.
00033         curso_academico (list): Lista con los cursos académicos
00034         titulacion (str): Titulación seleccionada
00035
00036     Returns:

```

```
00037     go.Figure: Figura con el gráfico
00038
00039     """
00040
00041 fig = go.Figure()
00042
00043 fig.update_layout(
00044     title={"text": "Evolución del progreso académico", "x": 0.5},
00045     xaxis={"title": "Curso académico"},
00046     yaxis={"title": "Nº Asignaturas de superadas"},
00047     showlegend=False,
00048 )
00049
00050     if not (alumno_id and curso_academico and titulacion):
00051         return fig
00052
00053     try:
00054         curso_academico = list_to_tuple(curso_academico)
00055     except Exception as e:
00056         print("Error:", e)
00057         return fig
00058
00059 data = asignaturas_superadas(alumno_id, curso_academico, titulacion)
00060
00061     if not data:
00062         return fig
00063
00064 data = pd.DataFrame(data)
00065 academic_years = data["curso_academico"]
00066 subjects_passed = data["n_asig_superadas"]
00067
00068 cumulative_passed = []
00069 cumulative_total = 0
00070 for count in subjects_passed:
00071     cumulative_total += count
00072     cumulative_passed.append(cumulative_total)
00073
00074 fig.add_trace(
00075     go.Bar(x=academic_years, y=cumulative_passed, marker_color="blue", opacity=0.7)
00076 )
00077
00078 return fig
```

6.95. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/↔ Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/graphs/personal/callback_graph_↔ tasa_rendimiento_alumnado.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.alumnado](#)
- namespace [callbacks.alumnado.graphs](#)
- namespace [callbacks.alumnado.graphs.personal](#)
- namespace [callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado](#)

Funciones

- def [callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado.update_graph_alumnado](#)(alumno_id, curso_academico, titulacion)

6.96. callback_graph_tasa_rendimiento_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_tasa_rendimiento_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de tasa de rendimiento
00004 #       del perfil "Alumno" de la pestaña "Expediente académico personal".
00005 # @version 1.0
00006 # @date 06/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu01138589@ull.edu.es
00010 #
00011
00012 from dash import Input, Output, callback
00013 import plotly.graph_objs as go
00014 import pandas as pd
00015 from data.queries import asignaturas_matriculadas_y_superadas
00016 from util import list_to_tuple
00017
00018
00019 @callback(
00020     Output("graph-bar-tasa-rendimiento", "figure"),
00021     Input("selected-alumnado-store", "data"),
00022     Input("curso-academico", "value"),
00023     Input("titulacion-alumnado", "value"),
00024 )
00025 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00026 """
00027 Actualiza el gráfico de tasa de rendimiento del perfil "Alumno"
00028 de la pestaña "Expediente académico personal".
00029
00030     Args:
00031         alumno_id (str): Identificador del alumno.
00032         curso_academico (list): Lista con los cursos académicos
00033             titulacion (str): Titulación seleccionada
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037
00038 """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     title={"text": "Tasa de rendimiento por curso académico", "x": 0.5},
00044     xaxis={"title": "Tasa de rendimiento (%)"},
00045     yaxis={"title": "Curso académico"},
00046     showlegend=False,
00047 )
00048
00049 if not (alumno_id and curso_academico and titulacion):
00050     return fig
00051
00052 try:
00053     curso_academico = list_to_tuple(curso_academico)
00054 except Exception as e:
00055     print("Error:", e)
00056     return fig
00057
00058 data = asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion)
00059
00060 if not data:
00061     return fig
00062
00063 df = pd.DataFrame(data, columns=["curso_academico", "matriculadas", "superadas"])
00064 df["tasa_rendimiento"] = (df["superadas"] / df["matriculadas"]) * 100
00065
00066 fig.add_trace(
00067     go.Bar(
00068         x=df["tasa_rendimiento"],
00069         y=df["curso_academico"],
00070         orientation="h",
00071         marker_color="blue",
00072         opacity=0.7,
00073         width=0.7,
00074     )
00075 )
00076
00077 return fig

```

6.97. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/utils/callback_resumen_alumnado.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.utils
- namespace callbacks.alumnado.utils.callback_resumen_alumnado

Funciones

- def callbacks.alumnado.utils.callback_resumen_alumnado.update_resumen_alumnado (alumno_id, titulacion)
- def callbacks.alumnado.utils.callback_resumen_alumnado.not_data ()

6.98. callback_resumen_alumnado.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_resumen_alumnado.py
00003 # @brief Este fichero contiene el callback para actualizar el resumen del perfil "Alumno"
00004 # @version 1.0
00005 # @date 05/05/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, callback, Output, Input
00012 import pandas as pd
00013 from data.queries import nota_media_alumno_titulacion, resumen_alumno
00014 from util import load_model
00015
00016 @callback(
00017     Output("resumen-alumnado", "children"),
00018     Input("selected-alumnado-store", "data"),
00019     Input("titulacion-alumnado", "value"),
00020 )
00021 def update_resumen_alumnado(alumno_id, titulacion):
00022     """
00023 Actualiza el resumen del perfil "Alumno".
00024
00025     Args:
00026         alumno_id (str): Identificador del alumno.
00027         titulacion (str): Titulación seleccionada
00028
00029     Returns:
00030         list: Componentes con el resumen del perfil "Alumno"
00031     """
00032 if not (alumno_id and titulacion):
00033     return not_data()
00034
00035     data = resumen_alumno(alumno_id, titulacion)
00036
00037     data = pd.DataFrame(data, columns=["id_alumno", "universidad", "titulacion", "abandona"])
00038
00039     if data.empty:
00040         return not_data()
00041
00042     componentes = [
00043         html.H2("Resumen"),
00044         html.P("Universidad:", className="resumen-label"),
00045         html.P(data["universidad"].iloc[0]),
00046         html.P("Titulación:", className="resumen-label"),
00047         html.P(data["titulacion"].iloc[0]),
00048         html.P("Alumno:", className="resumen-label"),
```

```

00049     html.P(alumno_id),
00050     html.P("Nota Media:", className="resumen-label"),
00051     html.P(not_a_media_alumno_titulacion(alumno_id, titulacion))
00052 ]
00053
00054 if data["abandona"].iloc[0] == 'no':
00055     componentes.append(html.P("Estado:", className="resumen-label"))
00056     componentes.append(html.P("Activo"))
00057     componentes.append(html.P("Probabilidad de abandono:", className="resumen-label"))
00058     data_model = load_model()
00059
00060     probabilidad_abandono = data_model[data_model['id'] ==
00061         alumno_id]['probabilidad_abandono'].values[0]
00062     componentes.append(html.P(f"{probabilidad_abandono:.2%}"))
00063 else:
00064     componentes.append(html.P("Estado:", className="resumen-label"))
00065     componentes.append(html.P("Abandona"))
00066
00067 componentes.append(html.Hr())
00068
00069 return html.Div(componentes)
00070
00071
00072
00073 def not_data():
00074     return html.Div(
00075     [
00076         html.H2("Resumen"),
00077         html.P("Universidad:", className="resumen-label"),
00078         html.P("No disponible"),
00079         html.P("Titulación:", className="resumen-label"),
00080         html.P("No disponible"),
00081         html.P("Alumno:", className="resumen-label"),
00082         html.P("No disponible"),
00083         html.P("Nota Media:", className="resumen-label"),
00084         html.P("No disponible"),
00085         html.Hr(),
00086     ],
00087 )

```

6.99. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/callback_select_alumnado.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.utils
- namespace callbacks.alumnado.utils.callback_select_alumnado

Funciones

- def callbacks.alumnado.utils.callback_select_alumnado.store_selected_alumnado(selected_value, stored_value)

6.100. callback_select_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_select_alumnado.py
00003 # @brief Este fichero contiene el callback para almacenar el valor
00004 # seleccionado en el dropdown de alumnos
00005 # @version 1.0

```

```
00006 # @date 05/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import Input, Output, State, callback
00013 from data.queries import alumnos_all
00014
00015
00016 @callback(
00017     Output("alumnado-dropdown", "value"),
00018     Output("alumnado-dropdown", "options"),
00019     Output("selected-alumnado-store", "data"),
00020     Input("alumnado-dropdown", "value"),
00021     State("selected-alumnado-store", "data"),
00022 )
00023 def store_selected_alumnado(selected_value, stored_value):
00024     """
00025 Almacena el valor seleccionado en el dropdown de alumnos.
00026
00027 Args:
00028 selected_value (str): Valor seleccionado en el dropdown
00029 stored_value (str): Valor almacenado
00030
00031 Returns:
00032 str: Valor seleccionado
00033 list: Opciones del dropdown
00034 str: Valor almacenado
00035 """
00036 data = alumnos_all()
00037
00038 if not data:
00039     return None, [], None
00040
00041 opciones_dropdown = [{"label": alumno[0], "value": alumno[0]} for alumno in data]
00042
00043 if selected_value is None and stored_value in [
00044     op["value"] for op in opciones_dropdown
00045 ]:
00046     return stored_value, opciones_dropdown, stored_value
00047
00048 return selected_value, opciones_dropdown, selected_value
```

6.101. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/↔ Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/alumnado/utils/callback_tabs_alumnado.py

Namespaces

- namespace callbacks
- namespace callbacks.alumnado
- namespace callbacks.alumnado.utils
- namespace callbacks.alumnado.utils.callback_tabs_alumnado

Funciones

- def callbacks.alumnado.utils.callback_tabs_alumnado.render_content(tab, selected_alumnado)

6.102. callback_tabs_alumnado.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_tabs_alumnado.py
00003 # @brief Este fichero contiene el callback para renderizar el contenido
00004 #       de las pestañas del dashboard del alumnado
```

```
00005 # @version 1.0
00006 # @date 29/04/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html, callback, Output, Input, State
00013 from components.common.sidebar import sidebar
00014 from components.common.filters import filters
00015 from components.alumnado.utils.select_alumnado import select_alumnado
00016 from components.alumnado.utils.recomendador_alumnado import recomendador_alumnado
00017 from components.alumnado.utils.resumen_alumnado import resumen_alumnado
00018 from components.alumnado.filters.filter_curso_academico_alumnado import
    filter_curso_academico_alumnado
00019 from components.alumnado.filters.filter_asignaturas_matri_alumnado import
    filter_asignaturas_matri_alumnado
00020 from components.alumnado.filters.filter_titulacion_alumnado import filter_titulacion_alumnado
00021 from components.alumnado.graphs.graphs_general_alumnado import graphs_general_alumnado
00022 from components.alumnado.graphs.graphs_personal_alumnado import graphs_personal_alumnado
00023
00024
00025 @callback(
00026     Output('tabs-alumnado-content', 'children'),
00027     Input('tabs-alumnado', 'value'),
00028     State('selected-alumnado-store', 'data')
00029 )
00030 def render_content(tab, selected_alumnado):
00031     """
00032 Renderiza el contenido de las pestañas del dashboard del alumnado.
00033
00034 Args:
00035 tab (str): Pestaña seleccionada
00036 selected_alumnado (str): Alumno seleccionado
00037
00038 Returns:
00039 list: Componentes de la pestaña seleccionada
00040
00041 """
00042 if tab == 'expediente-personal-tab':
00043     return html.Div([
00044         select_alumnado(),
00045         html.H2("Dashboard Alumnado", style={'text-align': 'center'}),
00046         html.Div([
00047             sidebar([
00048                 resumen_alumnado(),
00049                 filters([
00050                     filter_titulacion_alumnado(),
00051                     filter_curso_academico_alumnado()
00052                 ])
00053             ]),
00054             graphs_personal_alumnado()
00055         ], className='content-layout-dashboard')
00056     ])
00057 elif tab == 'rendimiento-academico-tab':
00058     return html.Div([
00059         html.H2("Dashboard Alumnado", style={'text-align': 'center'}),
00060         html.Div([
00061             sidebar([
00062                 resumen_alumnado(),
00063                 filters([
00064                     filter_titulacion_alumnado(),
00065                     filter_curso_academico_alumnado(),
00066                     filter_asignaturas_matri_alumnado()
00067                 ])
00068             ]),
00069             graphs_general_alumnado()
00070         ], className='content-layout-dashboard')
00071     ])
00072 elif tab == 'recomendador-tab':
00073     return html.Div([
00074         recomendador_alumnado()
00075     ])
00076
```

6.103. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/common/callback_sidebarCollapse.py

Namespaces

- namespace callbacks
- namespace callbacks.common
- namespace callbacks.common.callback_sidebarCollapse

Funciones

- def callbacks.common.callback_sidebarCollapse.sidebarCollapse (n, is_open)

6.104. callback_sidebarCollapse.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_sidebarCollapse.py
00003 # @brief Este fichero contiene el callback para colapsar y descolapsar el sidebar
00004 # @version 1.0
00005 # @date 29/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martin
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import Output, Input, State, callback
00012
00013
00014 @callback(
00015     Output("collapse", "is_open"),
00016     Input("sidebar-toggle", "n_clicks"),
00017     State("collapse", "is_open"),
00018 )
00019 def sidebarCollapse(n, is_open):
00020     """
00021 Callback que permite colapsar y descolapsar el sidebar.
00022
00023 Args:
00024 n (int): Número de clicks
00025 is_open (bool): Estado del sidebar
00026
00027 Returns:
00028 bool: Estado del sidebar
00029 """
00030 if n:
00031     return not is_open
00032     return is_open
```

6.105. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/common/callback_update_layout.py

Namespaces

- namespace callbacks
- namespace callbacks.common
- namespace callbacks.common.callback_update_layout

Funciones

- def `callbacks.common.callback_update_layout.update_layout` (`role`)

6.106. `callback_update_layout.py`

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_update_layout.py
00003 # @brief Este fichero contiene el callback para actualizar el layout de
00004 #           la aplicación según el rol del usuario.
00005 # @version 1.0
00006 # @date 29/04/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import Output, Input, callback
00013 from layouts.alumno_layout import alumno_layout
00014 from layouts.docente_layout import docente_layout
00015 from layouts.gestor_layout import gestor_layout
00016
00017 @callback(
00018     Output('page-content', 'children'),
00019     Input('store-role', 'data')
00020 )
00021 def update_layout(role):
00022     """
00023 Actualiza el layout de la aplicación según el rol del usuario.
00024
00025 Args:
00026 role (str): Rol del usuario
00027
00028 Returns:
00029 str: Layout correspondiente al rol del usuario
00030
00031 """
00032 if role == 'Alumno':
00033     return alumno_layout()
00034 elif role == 'Docente':
00035     return docente_layout()
00036 elif role == 'Gestor':
00037     return gestor_layout()
00038 return ""

```

6.107. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/callback_user_role.py

Namespaces

- namespace `callbacks`
- namespace `callbacks.common`
- namespace `callbacks.common.callback_user_role`

Funciones

- def `callbacks.common.callback_user_role.initialize_dropdown` (`ts, stored_role`)
- def `callbacks.common.callback_user_role.update_role` (`new_role, current_role`)

Variables

- bool callbacks.common.callback_user_role.prevent_initial_call = True,

6.108. callback_user_role.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_user_role.py
00003 # @brief Este fichero contiene los callbacks para almacenar y actualizar el rol del usuario
00004 # @version 1.0
00005 # @date 14/06/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martin
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import Output, Input, State, callback
00012
00013
00014 @callback(
00015     Output("dropdown-role", "value"),
00016     Input("store-role", "modified_timestamp"),
00017     State("store-role", "data"),
00018     prevent_initial_call=True,
00019 )
00020 def initialize_dropdown(ts, stored_role):
00021     """
00022     Inicializa el dropdown con el rol almacenado.
00023
00024 Args:
00025 ts (float): Timestamp de la última modificación del store
00026 stored_role (str): Rol almacenado
00027 """
00028 return stored_role if ts is not None else "Alumno"
00029
00030
00031 @callback(
00032     Output("store-role", "data"),
00033     Input("dropdown-role", "value"),
00034     State("store-role", "data"),
00035     prevent_initial_call=True,
00036 )
00037 def update_role(new_role, current_role):
00038     """
00039     Actualiza el rol almacenado en el store.
00040
00041 Args:
00042 new_role (str): Nuevo rol seleccionado
00043 current_role (str): Rol actual almacenado
00044
00045 Returns:
00046 str: Rol almacenado
00047 """
00048 if new_role != current_role:
00049     return new_role
00050 return current_role

```

6.109. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/docente/filters/callback_filter_all←
_asignaturas_titulacion_docente.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.filters
- namespace callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente

Funciones

- def callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente.update_filter_asignaturas_docente(titulacion, curso_academico, n_clicks, existing_options)

6.110. callback_filter_all_asignaturas_titulacion_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_filter_all_asignaturas_titulacion_docente.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del dropdown con todas las asignaturas de la titulación seleccionada
00005 #       y gestionar el evento del botón que selecciona todas las asignaturas.
00006 # @version 1.0
00007 # @date 20/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martin
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, State, callback, callback_context
00014 from data.queries import asignaturas_actas_titulacion
00015
00016
00017 @callback(
00018     Output("all-asignaturas-titulacion-docente", "options"),
00019     Output("all-asignaturas-titulacion-docente", "value"),
00020     Input("titulacion-docente", "value"),
00021     Input("all-cursos-academicos-docente", "value"),
00022     Input("select-all-asignaturas-titulacion-docente", "n_clicks"),
00023     State("all-asignaturas-titulacion-docente", "options"),
00024 )
00025 def update_filter_asignaturas_docente(titulacion, curso_academico, n_clicks, existing_options):
00026 """
00027 Actualiza las opciones del dropdown con todas las asignaturas de la titulación seleccionada y
00028 gestiona el evento del botón que selecciona todas las asignaturas.
00029
00030 Args:
00031     titulacion (str): Titulación seleccionada
00032     curso_academico (list): Cursos académicos seleccionados
00033     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00034     existing_options (list): Opciones actuales del dropdown
00035
00036 Returns:
00037     list: Opciones del dropdown
00038     list: Valor seleccionado
00039 """
00040 ctx = callback_context
00041 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00042
00043 if trigger_id == "select-all-asignaturas-titulacion-docente" and n_clicks > 0:
00044     return existing_options, [option["value"] for option in existing_options]
00045
00046 if not (titulacion and curso_academico):
00047     return [], None
00048
00049 data = asignaturas_actas_titulacion(titulacion, curso_academico)
00050
00051 if not data:
00052     return [], None
00053
00054 opciones_dropdown = [
00055     {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
00056 ]
00057 value = (
00058     [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
00059 )
00060
00061 return opciones_dropdown, value

```

6.111. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/docente/filters/callback_filter_all_←
_curso_academico.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.filters
- namespace callbacks.docente.filters.callback_filter_all_curso_academico

Funciones

- def callbacks.docente.filters.callback_filter_all_curso_academico.update_filter_all_cursos_academicos_docente (titulacion)

6.112. callback_filter_all_curso_academico.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_filter_all_curso_academico.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del dropdown con todos los cursos académicos de la titulación seleccionada.
00005 # @version 1.0
00006 # @date 20/05/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martin
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import Input, Output, callback
00013 from data.queries import curso_academico_actas_titulacion
00014
00015
00016 @callback(
00017     Output("all-cursos-academicos-docente", "options"),
00018     Output("all-cursos-academicos-docente", "value"),
00019     Input("titulacion-docente", "value"),
00020 )
00021 def update_filter_all_cursos_academicos_docente(titulacion):
00022 """
00023 Actualiza las opciones del dropdown con todos los cursos académicos de la titulación seleccionada.
00024
00025 Args:
00026 titulacion (str): Titulación seleccionada
00027
00028 Returns:
00029 list: Opciones del dropdown
00030 list: Valor seleccionado
00031 """
00032
00033 if not titulacion:
00034     return [], None
00035
00036     data = curso_academico_actas_titulacion(titulacion)
00037
00038     if not data:
00039         return [], None
00040
00041     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00042     value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00043
00044     return opciones_dropdown, value
```

6.113. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/docente/filters/callback_filter_←
asignaturas_docente.py**

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.filters
- namespace callbacks.docente.filters.callback_filter_asignaturas_docente

Funciones

- def callbacks.docente.filters.callback_filter_asignaturas_docente.update_filter_asignaturas_docente(docente←_id, titulacion)

6.114. callback_filter_asignaturas_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file callback_filter_asignaturas_docente.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del dropdown con las asignaturas del docente seleccionado.
00005 # @version 1.0
00006 # @date 15/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import Input, Output, callback
00013 from data.queries import asignaturas_docente
00014
00015
00016 @callback(
00017     Output("asignaturas-docente", "options"),
00018     Output("asignaturas-docente", "value"),
00019     Input("selected-docente-store", "data"),
00020     Input("titulacion-docente", "value"),
00021 )
00022 def update_filter_asignaturas_docente(docente_id, titulacion):
00023 """
00024 Actualiza las opciones del dropdown con las asignaturas del docente seleccionado.
00025
00026 Args:
00027 docente_id (str): ID del docente
00028 titulacion (str): Titulación seleccionada
00029
00030 Returns:
00031 list: Opciones del dropdown
00032 str: Valor seleccionado
00033
00034 """
00035 if not (docente_id and titulacion):
00036     return [], None
00037
00038     data = asignaturas_docente(docente_id, titulacion)
00039
00040     if not data:
00041         return [], None
00042
00043     opciones_dropdown = [
00044         {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
00045     ]
00046     value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00047
00048     return opciones_dropdown, value
```

6.115. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/docente/filters/callback_filter_←
curso_academico_docente.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.filters
- namespace callbacks.docente.filters.callback_filter_curso_academico_docente

Funciones

- def callbacks.docente.filters.callback_filter_curso_academico_docente.update_filter_curso_academico_docente(docente_id, asignatura, n_clicks, existing_options)

6.116. callback_filter_curso_academico_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file callback_filter_curso_academico_docente.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del dropdown con los cursos académicos del docente seleccionado y
00005 #       gestionar el evento del botón que selecciona todos los cursos académicos.
00006 # @version 1.0
00007 # @date 15/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, callback, State, callback_context
00014 from data.queries import curso_academico_docente
00015 from callbacks.docente.utils.callback_select_docente import store_selected_docente
00016
00017
00018 @callback(
00019     Output("curso-academico-docente", "options"),
00020     Output("curso-academico-docente", "value"),
00021     Input("selected-docente-store", "data"),
00022     Input("asignaturas-docente", "value"),
00023     Input("select-all-cursos-academicos-docente", "n_clicks"),
00024     State("curso-academico-docente", "options"),
00025 )
00026 def update_filter_curso_academico_docente(docente_id, asignatura, n_clicks, existing_options):
00027 """
00028 Actualiza las opciones del dropdown con los cursos académicos del docente seleccionado y
00029 gestiona el evento del botón que selecciona todos los cursos académicos.
00030
00031 Args:
00032     docente_id (str): ID del docente
00033     asignatura (str): Asignatura seleccionada
00034     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00035         existing_options (list): Opciones actuales del dropdown
00036
00037     Returns:
00038         list: Opciones del dropdown
00039         list: Valor seleccionado
00040
00041 """
00042 ctx = callback_context
00043 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00044
00045 if trigger_id == "select-all-cursos-academicos" and n_clicks > 0:
00046     return existing_options, [option["value"] for option in existing_options]
00047
00048 if not (docente_id and asignatura):
```

```

00049     return [], None
00050
00051     data = curso_academico_docente(docente_id, asignatura)
00052
00053     if not data:
00054         return [], None
00055
00056     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00057     value = [option["value"] for option in opciones_dropdown]
00058
00059     return opciones_dropdown, value

```

6.117. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/docente/filters/callback_filter_←
titulacion_docente.py**

Namespaces

- namespace `callbacks`
- namespace `callbacks.docente`
- namespace `callbacks.docente.filters`
- namespace `callbacks.docente.filters.callback_filter_titulacion_docente`

Funciones

- def `callbacks.docente.filters.callback_filter_titulacion_docente.update_filter_titulacion_docente` (docente_id, selected_value, stored_titulacion)

6.118. callback_filter_titulacion_docente.py

Ir a la documentación de este archivo.

```

00001 #
00002 # @file callback_filter_titulacion_docente.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del dropdown con las titulaciones del docente seleccionado.
00005 # @version 1.0
00006 # @date 19/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import Input, Output, State, callback
00013 from data.queries import titulacion_docente
00014
00015 @callback(
00016     Output('titulacion-docente', 'options'),
00017     Output('titulacion-docente', 'value'),
00018     Output('selected-titulacion-docente-store', 'data'),
00019     Input('selected-docente-store', 'data'),
00020     Input('titulacion-docente', 'value'),
00021     State('selected-titulacion-docente-store', 'data')
00022 )
00023 def update_filter_titulacion_docente(docente_id, selected_value, stored_titulacion):
00024     """
00025 Actualiza las opciones del dropdown con las titulaciones del docente seleccionado.
00026
00027 Args:
00028     docente_id (str): ID del docente
00029     selected_value (str): Valor seleccionado
00030     stored_titulacion (str): Titulación seleccionada
00031
00032 Returns:

```

```
00033 list: Opciones del dropdown
00034 str: Valor seleccionado
00035
00036 """
00037 if not docente_id:
00038     return [], None, None
00039
00040 data = titulacion_docente(docente_id)
00041
00042 if not data:
00043     return [], None, None
00044
00045 opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
00046
00047 if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
00048     return opciones_dropdown, stored_titulacion, stored_titulacion
00049
00050 return opciones_dropdown, selected_value, selected_value
00051
00052
```

6.119. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/general/callback_graph_all_calif_cualitativa_docente.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.graphs
- namespace callbacks.docente.graphs.general
- namespace callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente

Funciones

- def callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente.update_graph_docente(titulacion, curso_academico, asignatura)

6.120. callback_graph_all_calif_cualitativa_docente.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_graph_all_calif_cualitativa_docente.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de calificaciones cualitativas
00004 #       de la titulación seleccionada del perfil "Docente" de la pestaña "Rendimiento académico
00005 #       general".
00006 # @version 1.0
00007 # @date 20/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012 from dash import Output, Input, callback
00013 import plotly.graph_objs as go
00014 import pandas as pd
00015 from data.queries import calif_all_cualitativa_asignaturas
00016 from util import list_to_tuple
00017
00018
00019 @callback(
00020     Output("calificaciones-cuali-all-asig-docente", "figure"),
```

```

00021     Input("titulacion-docente", "value"),
00022     Input("all-cursos-academicos-docente", "value"),
00023     Input("all-asignaturas-titulacion-docente", "value"),
00024 )
00025 def update_graph_docente(titulacion, curso_academico, asignatura):
00026     """
00027 Actualiza el gráfico de calificaciones cualitativas de la titulación
00028 seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00029
00030     Args:
00031         titulacion (str): Titulación seleccionada
00032         curso_academico (list): Lista con los cursos académicos
00033         asignatura (list): Lista con las asignaturas seleccionadas
00034
00035     Returns:
00036         go Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     barmode="stack",
00043     title={"text": "Calificaciones cualitativas de la titulación", "x": 0.5},
00044     xaxis={"title": "Asignaturas", "tickangle": 45},
00045     yaxis={"title": "Nº Alumnos matriculados"},
00046     legend=dict(
00047         x=1,
00048         y=1,
00049         traceorder="normal",
00050         font=dict(
00051             size=10
00052         ),
00053         title="Calificaciones"
00054     ),
00055 )
00056
00057 if not (curso_academico and asignatura):
00058     return fig
00059
00060 try:
00061     asignatura = list_to_tuple(asignatura)
00062 except Exception as e:
00063     return fig
00064
00065 data = calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignatura)
00066
00067 if not data:
00068     return fig
00069
00070 df = pd.DataFrame(
00071     data, columns=["titulacion", "asignatura", "calificación", "n_alumnos"]
00072 )
00073
00074 colors_mapping = {
00075     "Sobresaliente": "blue",
00076     "Notable": "green",
00077     "Aprobado": "orange",
00078     "Suspensos": "red",
00079     "No presentado": "gray",
00080 }
00081
00082 for calif, color in colors_mapping.items():
00083     df_calif = df[df["calificación"] == calif]
00084     fig.add_trace(
00085         go.Bar(
00086             x=df_calif["asignatura"],
00087             y=df_calif["n_alumnos"],
00088             name=calif,
00089             marker_color=color,
00090             opacity=0.7,
00091         )
00092     )
00093
00094 fig.update_xaxes(categoryorder="array", categoryarray=asignatura)
00095
00096 return fig

```

6.121. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/docente/graphs/general/callback←
_graph_all_calif_media_docente.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.graphs
- namespace callbacks.docente.graphs.general
- namespace callbacks.docente.graphs.general.callback_graph_all_calif_media_docente

Funciones

- def callbacks.docente.graphs.general.callback_graph_all_calif_media_docente.update_graph_docente (titulacion, curso_academico, asignatura)

6.122. callback_graph_all_calif_media_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file callback_graph_all_calif_media_docente.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de nota media por asignatura de
00004 # la titulación
00005 # seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00006 # @version 1.0
00007 # @date 05/06/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012 from dash import callback, Output, Input
00013 import plotly.graph_objs as go
00014 import pandas as pd
00015 from data.queries import calif_media_asignaturas
00016 from util import list_to_tuple
00017
00018
00019 @callback(
00020     Output("calificaciones-media-all-asig-docente", "figure"),
00021     Input("titulacion-docente", "value"),
00022     Input("all-cursos-academicos-docente", "value"),
00023     Input("all-asignaturas-titulacion-docente", "value"),
00024 )
00025 def update_graph_docente(titulacion, curso_academico, asignatura):
00026 """
00027 Actualiza el gráfico de nota media por asignatura de la titulación
00028 seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00029
00030     Args:
00031         titulacion (str): Titulación seleccionada
00032         curso_academico (list): Lista con los cursos académicos
00033         asignatura (list): Lista con las asignaturas seleccionadas
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037 """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     title={"text": "Nota media por asignatura de la titulación", "x": 0.5},
00043     xaxis_title="Asignaturas",
00044     yaxis_title="Nota media",
00045 )
```

```

00046
00047     if not (curso_academico and asignatura):
00048         return fig
00049
00050     try:
00051         asignatura = list_to_tuple(asignatura)
00052     except Exception as e:
00053         return fig
00054
00055     data = calif_media_asignaturas(titulacion, curso_academico, asignatura)
00056
00057     if not data:
00058         return fig
00059
00060     df = pd.DataFrame(data, columns=["asignatura", "media_calif"])
00061
00062     fig.add_trace(
00063         go.Bar(
00064             x=df["asignatura"], y=df["media_calif"], marker_color="blue", opacity=0.7
00065         )
00066     )
00067
00068     return fig

```

6.123. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_genero_docente.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.graphs
- namespace callbacks.docente.graphs.personal
- namespace callbacks.docente.graphs.personal.callback_graph_alu_genero_docente

Funciones

- def callbacks.docente.graphs.personal.callback_graph_alu_genero_docente.update_graph_docente (asignaturas, curso_academico, docente_id)

6.124. callback_graph_alu_genero_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_alu_genero_docente.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de evolución de alumnos
00004 # matriculados por género
00005 # @version 1.0
00006 # @date 15/05/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import Input, Output, callback
00013 import plotly.graph_objs as go
00014 import pandas as pd
00015 from data.queries import alumnos_genero_docente
00016 from util import list_to_tuple
00017

```

```
00018
00019 @callback(
00020     Output("graph-alumnos-matri-genero", "figure"),
00021     Input("asignaturas-docente", "value"),
00022     Input("curso-academico-docente", "value"),
00023     Input("selected-docente-store", "data"),
00024 )
00025 def update_graph_docente(asignaturas, curso_academico, docente_id):
00026     """
00027     Actualiza el gráfico de evolución de alumnos matriculados por género
00028     del perfil "Docente" de la pestaña "Rendimiento académico personal".
00029
00030     Args:
00031         asignaturas (list): Lista con las asignaturas seleccionadas
00032         curso_academico (list): Lista con los cursos académicos
00033         docente_id (str): Identificador del docente
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     barmode="stack",
00043     title={"text": "Evolución alumnos matriculados por género", "x": 0.5},
00044     xaxis={"title": "Curso académico"},
00045     yaxis={"title": "Nº Alumnos matriculados"},
00046     legend={"title": "Género"},
00047 )
00048
00049 if not (asignaturas and curso_academico and docente_id):
00050     return fig
00051
00052 try:
00053     curso_academico = list_to_tuple(curso_academico)
00054 except Exception as e:
00055     return fig
00056
00057 data = alumnos_genero_docente(docente_id, asignaturas, curso_academico)
00058
00059 if not data:
00060     return fig
00061
00062 df = pd.DataFrame(data, columns=["curso_academico", "genero", "cantidad"])
00063
00064 df_pivot = df.pivot(
00065     index="curso_academico", columns="genero", values="cantidad"
00066 ).fillna(0)
00067
00068 colores = {"Femenino": "red", "Masculino": "blue"}
00069
00070 for genero in df_pivot.columns:
00071     fig.add_trace(
00072         go.Bar(
00073             x=df_pivot.index,
00074             y=df_pivot[genero],
00075             name="Mujeres" if genero == "Femenino" else "Hombres",
00076             marker_color=colores[genero],
00077             opacity=0.7,
00078         )
00079     )
00080
00081 return fig
```

6.125. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_media_docente.py

Namespaces

- namespace `callbacks`
- namespace `callbacks.docente`
- namespace `callbacks.docente.graphs`
- namespace `callbacks.docente.graphs.personal`
- namespace `callbacks.docente.graphs.personal.callback_graph_alu_media_docente`

Funciones

- def callbacks.docente.graphs.personal.callback_graph_alu_media_docente.update_graph_docente (asignaturas, curso_academico, docente_id)

6.126. callback_graph_alu_media_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_alu_media_docente.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de evolución de la nota media
00004 # por asignatura
00005 #     del perfil "Docente" de la pestaña "Rendimiento académico personal".
00006 # @version 1.0
00007 # @date 16/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012 from dash import Input, Output, callback
00013 import plotly.graph_objs as go
00014 import pandas as pd
00015 from data.queries import alumnos_nota_media_docente
00016 from util import list_to_tuple
00017
00018
00019 @callback(
00020     Output("graph-alumnos-nota-media", "figure"),
00021     Input("asignaturas-docente", "value"),
00022     Input("curso-academico-docente", "value"),
00023     Input("selected-docente-store", "data"),
00024 )
00025 def update_graph_docente(asignaturas, curso_academico, docente_id):
00026     """
00027 Actualiza el gráfico de evolución de la nota media por asignatura
00028 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00029
00030     Args:
00031         asignaturas (list): Lista con las asignaturas seleccionadas
00032         curso_academico (list): Lista con los cursos académicos
00033         docente_id (str): Identificador del docente
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     title={"text": "Evolución nota media por asignatura", "x": 0.5},
00043         xaxis_title="Curso académico",
00044         yaxis_title="Nota media",
00045         xaxis=dict(type="category"),
00046     )
00047
00048 if not (asignaturas and curso_academico and docente_id):
00049     return fig
00050
00051 try:
00052     curso_academico = list_to_tuple(curso_academico)
00053     asignaturas = list_to_tuple(asignaturas)
00054 except Exception as e:
00055     return fig
00056
00057 data = alumnos_nota_media_docente(asignaturas, curso_academico)
00058
00059 if not data:
00060     return fig
00061
00062 df = pd.DataFrame(data, columns=["curso_academico", "asignatura", "nota_media"])
00063
00064 fig.add_trace(
00065     go.Bar(
00066         x=df["curso_academico"],
00067         y=df["nota_media"],
00068         name="Nota media",
00069         marker=dict(color="blue"),
00070         opacity=0.7,
00071     )

```

```
00072 )
00073
00074     return fig
```

6.127. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_← TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_nota_cualitativa_docente.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.graphs
- namespace callbacks.docente.graphs.personal
- namespace callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente

Funciones

- def callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente.update_graph_docente(asignaturas, curso_academico, docente_id)

6.128. callback_graph_alu_nota_cualitativa_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file callback_graph_alu_nota_cualitativa_docente.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico de evolución de calificaciones
00004 #       cualitativas
00005 #       del perfil "Docente" de la pestaña "Rendimiento académico personal".
00006 # @version 1.0
00007 # @date 16/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012 from dash import Input, Output, callback
00013 import plotly.graph_objs as go
00014 import pandas as pd
00015 from data.queries import alumnos_nota_cualitativa_docente
00016 from util import list_to_tuple
00017
00018
00019 @callback(
00020     Output("graph-alumnos-nota-cualitativa", "figure"),
00021     Input("asignaturas-docente", "value"),
00022     Input("curso-academico-docente", "value"),
00023     Input("selected-docente-store", "data"),
00024 )
00025 def update_graph_docente(asignaturas, curso_academico, docente_id):
00026     """
00027 Actualiza el gráfico de evolución de calificaciones cualitativas
00028 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00029
00030     Args:
00031         asignaturas (list): Lista con las asignaturas seleccionadas
00032         curso_academico (list): Lista con los cursos académicos
00033         docente_id (str): Identificador del docente
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037     """

```

```

00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     barmode="stack",
00043         title={"text": "Evolución calificaciones cualitativas", "x": 0.5},
00044         xaxis={"title": "Curso académico"},
00045         yaxis={"title": "Nº Alumnos matriculados"},
00046         legend_title_text="Calificación",
00047     )
00048
00049 if not (asignaturas and curso_academico and docente_id):
00050     return fig
00051
00052 try:
00053     curso_academico = list_to_tuple(curso_academico)
00054     asignaturas = list_to_tuple(asignaturas)
00055 except Exception as e:
00056     return fig
00057
00058 data = alumnos_nota_cualitativa_docente(asignaturas, curso_academico)
00059 if not data:
00060     return fig
00061
00062 df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "n_alumnos"])
00063
00064 # Pivotear el DataFrame para obtener las cantidades por calificación en columnas separadas
00065 df_pivot = df.pivot(
00066     index="curso_academico", columns="calificacion", values="n_alumnos"
00067 ).fillna(0)
00068 catagories = ["No presentado", "Suspensó", "Aprobado", "Notable", "Sobresaliente"]
00069
00070 for calif in catagories:
00071     if calif not in df_pivot:
00072         df_pivot[calif] = 0
00073
00074 df_pivot = df_pivot[catagories]
00075
00076 colors = {
00077     "Sobresaliente": "blue",
00078     "Notable": "green",
00079     "Aprobado": "orange",
00080     "Suspensó": "red",
00081     "No presentado": "gray",
00082 }
00083
00084 for calif in catagories:
00085     fig.add_trace(
00086         go.Bar(
00087             x=df_pivot.index,
00088             y=df_pivot[calif],
00089             name=calif,
00090             marker=dict(color=colors[calif]),
00091             opacity=0.7,
00092         )
00093     )
00094
00095 return fig

```

6.129. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_repetidores_nuevos_docente.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.docente](#)
- namespace [callbacks.docente.graphs](#)
- namespace [callbacks.docente.graphs.personal](#)
- namespace [callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente](#)

Funciones

- def callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente.update_graph_docente(asignaturas, curso_academico, docente_id)

6.130. callback_graph_alu_repetidores_nuevos_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_alu_repetidores_nuevos_docente.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #     de evolución de alumnos de nuevo ingreso y repetidores
00005 #     del perfil "Docente" de la pestaña "Rendimiento académico personal".
00006 # @version 1.0
00007 # @date 15/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, callback
00014 import plotly.graph_objs as go
00015 import pandas as pd
00016 from data.queries import alumnos_repetidores_nuevos
00017 from util import list_to_tuple
00018
00019
00020 @callback(
00021     Output("graph-alumnos-repetidores-nuevos", "figure"),
00022     Input("asignaturas-docente", "value"),
00023     Input("curso-academico-docente", "value"),
00024     Input("selected-docente-store", "data"),
00025 )
00026 def update_graph_docente(asignaturas, curso_academico, docente_id):
00027 """
00028 Actualiza el gráfico de evolución de alumnos de nuevo ingreso y repetidores
00029 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00030
00031     Args:
00032         asignaturas (list): Lista con las asignaturas seleccionadas
00033         curso_academico (list): Lista con los cursos académicos
00034         docente_id (str): Identificador del docente
00035
00036     Returns:
00037         go.Figure: Figura con el gráfico
00038     """
00039
00040 fig = go.Figure()
00041
00042 fig.update_layout(
00043     barmode="stack",
00044         title={"text": "Evolución alumnos de nuevo ingreso y repetidores", "x": 0.5},
00045         xaxis={"title": "Curso académico"},
00046         yaxis={"title": "Nº Alumnos matriculados"},
00047         showlegend=True,
00048         legend={"orientation": "h", "x": 0, "y": 1.1},
00049     )
00050
00051 if not (asignaturas and curso_academico and docente_id):
00052     return fig
00053
00054 try:
00055     curso_academico = list_to_tuple(curso_academico)
00056 except Exception as e:
00057     return fig
00058
00059 data = alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas)
00060
00061 if not data:
00062     return fig
00063
00064 df = pd.DataFrame(
00065     data,
00066     columns=["curso_academico", "alumnos_repetidores", "alumnos_nuevo_ingreso"],
00067 )
00068
00069 fig.add_trace(
00070     go.Bar(
00071         x=df["curso_academico"],
00072         y=df["alumnos_repetidores"],
00073

```

```

00073         name="Alumnos repetidores",
00074         marker_color="red",
00075         opacity=0.7,
00076     )
00077 )
00078
00079 fig.add_trace(
00080     go.Bar(
00081         x=df["curso_academico"],
00082         y=df["alumnos_nuevo_ingreso"],
00083         name="Alumnos de primera matrícula",
00084         marker_color="green",
00085         opacity=0.7,
00086     )
00087 )
00088
00089 return fig

```

6.131. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/callback_resumen_docente.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.docente](#)
- namespace [callbacks.docente.utils](#)
- namespace [callbacks.docente.utils.callback_resumen_docente](#)

Funciones

- def [callbacks.docente.utils.callback_resumen_docente.update_resumen_docente](#) (docente_id, titulacion)
- def [callbacks.docente.utils.callback_resumen_docente.not_data](#) ()

6.132. callback_resumen_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_resumen_docente.py
00003 # @brief Este fichero contiene el callback para actualizar el resumen del docente
00004 # @version 1.0
00005 # @date 14/05/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martin
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, callback, Output, Input
00012 from data.queries import universidades_docente
00013
00014
00015 @callback(
00016     Output("resumen-docente", "children"),
00017     Input("selected-docente-store", "data"),
00018     Input("titulacion-docente", "value"),
00019 )
00020 def update_resumen_docente(docente_id, titulacion):
00021     """
00022 Actualiza el resumen del docente
00023
00024 Args:
00025 docente_id (str): ID del docente
00026 titulacion (str): Titulación seleccionada
00027
00028 Returns:

```

```
00029 html.Div: Layout del resumen del docente
00030 """
00031 if not (docente_id and titulacion):
00032     return not_data()
00033
00034     data = universidades_docente(docente_id)
00035
00036     if not data:
00037         return not_data()
00038
00039     return html.Div(
00040         [
00041             html.H2("Resumen"),
00042             html.P("Universidad:", className="resumen-label"),
00043             html.P(data[0][1]),
00044             html.P("Titulación:", className="resumen-label"),
00045             html.P(titulacion),
00046             html.P("Docente:", className="resumen-label"),
00047             html.P(docente_id),
00048             html.Hr(),
00049         ]
00050     )
00051
00052
00053 def not_data():
00054     return html.Div(
00055         [
00056             html.H2("Resumen"),
00057             html.P("Universidad:", className="resumen-label"),
00058             html.P("No disponible"),
00059             html.P("Titulación:", className="resumen-label"),
00060             html.P("No disponible"),
00061             html.P("Docente:", className="resumen-label"),
00062             html.P("No disponible"),
00063             html.Hr(),
00064         ]
00065     )
```

6.133. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↔ Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/docente/utils/callback_select_docente.py

Namespaces

- namespace callbacks
- namespace callbacks.docente
- namespace callbacks.docente.utils
- namespace callbacks.docente.utils.callback_select_docente

Funciones

- def callbacks.docente.utils.callback_select_docente(store_selected_docente) (selected_value, stored_value)

6.134. callback_select_docente.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_select_docente.py
00003 # @brief Este fichero contiene el callback para almacenar el docente seleccionado en el store.
00004 # @version 1.0
00005 # @date 14/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
```

```

00011 from dash import Input, Output, callback, State
00012 from data.queries import docentes_all
00013
00014
00015 @callback(
00016     Output("docente-dropdown", "value"),
00017     Output("docente-dropdown", "options"),
00018     Output("selected-docente-store", "data"),
00019     Input("docente-dropdown", "value"),
00020     State("selected-docente-store", "data"),
00021 )
00022 def store_selected_docente(selected_value, stored_value):
00023     """
00024 Almacena el docente seleccionado en el store.
00025
00026 Args:
00027     selected_value (str): Valor seleccionado
00028     stored_value (str): Valor almacenado
00029
00030 Returns:
00031     str: Valor seleccionado
00032     list: Opciones del dropdown
00033     str: Valor almacenado
00034 """
00035
00036 data = docentes_all()
00037
00038 if not data:
00039     return None, [], None
00040
00041 opciones_dropdown = [{"label": docente[0], "value": docente[0]} for docente in data]
00042
00043 if selected_value is None and stored_value in [
00044     op["value"] for op in opciones_dropdown
00045 ]:
00046     return stored_value, opciones_dropdown, stored_value
00047
00048 return selected_value, opciones_dropdown, selected_value

```

6.135. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/callback_tabs_docente.py

Namespaces

- namespace `callbacks`
- namespace `callbacks.docente`
- namespace `callbacks.docente.utils`
- namespace `callbacks.docente.utils.callback_tabs_docente`

Funciones

- def `callbacks.docente.utils.callback_tabs_docente.render_content(tab, selected_docente)`

6.136. callback_tabs_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_tabs_docente.py
00003 # @brief Este fichero contiene el callback para renderizar el contenido
00004 #       de las pestañas del dashboard del docente.
00005 # @version 1.0
00006 # @date 14/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es

```

```
00010 #
00011
00012 from dash import html, callback, Input, Output, State
00013 from components.docente.utils.select_docente import select_docente
00014 from components.common.sidebar import sidebar
00015 from components.common.filters import filters
00016 from components.docente.utils.resumen_docente import resumen_docente
00017 from components.docente.utils.recomendador_docente import recomendador_docente
00018 from components.docente.filters.filter_curso_academico_docente import filter_curso_academico_docente
00019 from components.docente.filters.filter_asignaturas_docente import filter_asignaturas_docente
00020 from components.docente.filters.filter_titulacion_docente import filter_titulacion_docente
00021 from components.docente.filters.filter_all_curso_academico import filter_all_curso_academico
00022 from components.docente.filters.filter_all_asignaturas_titulacion_docente import
    filter_all_asignaturas_titulacion_docente
00023 from components.docente.graphs.graphs_personal_docente import graphs_personal_docente
00024 from components.docente.graphs.graphs_general_docente import graphs_general_docente
00025
00026
00027 @callback(
00028     Output('tabs-docente-content', 'children'),
00029     Input('tabs-docente', 'value'),
00030     State('selected-docente-store', 'data')
00031 )
00032 def render_content(tab, selected_docente):
00033 """
00034 Renderiza el contenido de las pestañas del dashboard del docente.
00035
00036 Args:
00037 tab (str): Pestaña seleccionada
00038 selected_docente (str): Docente seleccionado
00039
00040 Returns:
00041 ist: Componentes de la pestaña seleccionada
00042 """
00043
00044 if tab == 'rendimiento-academico-asignatura-tab':
00045     return html.Div([
00046         select_docente(),
00047         html.H2("Dashboard Docente", style={'text-align': 'center'}),
00048         html.Div([
00049             sidebar([
00050                 resumen_docente(),
00051                 filters([
00052                     filter_titulacion_docente(),
00053                     filter_asignaturas_docente(),
00054                     filter_curso_academico_docente()
00055                 ]),
00056             ]),
00057             graphs_personal_docente()
00058         ], className='content-layout-dashboard')
00059     ])
00060 elif tab == 'rendimiento-academico-tab':
00061     return html.Div([
00062         html.H2("Dashboard Docente", style={'text-align': 'center'}),
00063         html.Div([
00064             sidebar([
00065                 resumen_docente(),
00066                 filters([
00067                     filter_titulacion_docente(),
00068                     filter_all_curso_academico(),
00069                     filter_all_asignaturas_titulacion_docente()
00070                 ]),
00071             ]),
00072             graphs_general_docente()
00073         ], className='content-layout-dashboard')
00074     ])
00075 elif tab == 'recomendaciones-tab':
00076     return html.Div([
00077         recomendador_docente()
00078     ])
```

6.137. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/callback_filter_all_curso_academico_gestor.py

Namespaces

- namespace callbacks

- namespace [callbacks.gestor](#)
- namespace [callbacks.gestor.filters](#)
- namespace [callbacks.gestor.filters.callback_filter_all_curso_academico_gestor](#)

Funciones

- def [callbacks.gestor.filters.callback_filter_all_curso_academico_gestor.update_filter_all_curso_academico_gestor](#)(gestor_id, n_clicks, existing_options)

6.138. `callback_filter_all_curso_academico_gestor.py`

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_filter_all_curso_academico_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del filtro de todos los cursos académicos del perfil "Gestor" en la pestaña "Riesgo
00005 #       académico".
00006 # @version 1.0
00007 # @date 27/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012 from dash import Input, Output, State, callback, callback_context
00013 from data.queries import universidades_gestor, curso_academico_universidad
00014
00015
00016 @callback(
00017     Output("curso-all-academico-gestor", "options"),
00018     Output("curso-all-academico-gestor", "value"),
00019     Input("selected-gestor-store", "data"),
00020     Input("select-all-curso-academico-button", "n_clicks"),
00021     State("curso-all-academico-gestor", "options"),
00022 )
00023 def update_filter_all_curso_academico_gestor(gestor_id, n_clicks, existing_options):
00024 """
00025 Actualiza las opciones del filtro de todos los cursos académicos del perfil "Gestor"
00026 en la pestaña "Riesgo académico".
00027
00028 Args:
00029     gestor_id (str): ID del gestor seleccionado
00030     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00031     existing_options (list): Opciones actuales del filtro de curso académico
00032
00033 Returns:
00034     list: Opciones del dropdown
00035     list: Valores seleccionados
00036 """
00037
00038 ctx = callback_context
00039 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00040
00041 if trigger_id == "select-all-curso-academico-button" and n_clicks > 0:
00042     return existing_options, [option["value"] for option in existing_options]
00043
00044 if not gestor_id:
00045     return [], None
00046
00047 cod_universidad = universidades_gestor(gestor_id)
00048
00049 if not cod_universidad:
00050     return [], None
00051
00052 data = curso_academico_universidad(cod_universidad[0][0])
00053
00054 if not data:
00055     return [], None
00056
00057 opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00058 value = (
00059     [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
00060 )
00061
00062 return opciones_dropdown, value

```

6.139. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/gestor/filters/callback_filter_←
curso_academico_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.filters
- namespace callbacks.gestor.filters.callback_filter_curso_academico_gestor

Funciones

- def callbacks.gestor.filters.callback_filter_curso_academico_gestor.update_filter_curso_academico_gestor(gestor_id)

6.140. callback_filter_curso_academico_gestor.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_filter_curso_academico_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del filtro de curso académico del perfil "Gestor" en la
00005 #       pestaña "Indicadores académicos".
00006 # @version 1.0
00007 # @date 21/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, callback
00014 from data.queries import universidades_gestor, curso_academico_universidad
00015
00016
00017 @callback(
00018     Output("curso-academico-gestor", "options"),
00019     Output("curso-academico-gestor", "value"),
00020     Input("selected-gestor-store", "data"),
00021 )
00022 def update_filter_curso_academico_gestor(gestor_id):
00023     """
00024 Actualiza las opciones del filtro de curso académico del perfil "Gestor"
00025     en la pestaña "Indicadores académicos".
00026
00027     Args:
00028         gestor_id (str): ID del gestor seleccionado
00029
00030     Returns:
00031         list: Opciones del dropdown
00032         str: Valor seleccionado
00033     """
00034
00035 if not gestor_id:
00036     return [], None
00037
00038 cod_universidad = universidades_gestor(gestor_id)
00039
00040 if not cod_universidad:
00041     return [], None
00042
00043 data = curso_academico_universidad(cod_universidad[0][0])
00044
00045 if not data:
00046     return [], None
00047
00048 opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00049 value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00050
00051 return opciones_dropdown, value
```

6.141. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/gestor/filters/callback_filter_←
titulaciones_gestor.py**

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.filters
- namespace callbacks.gestor.filters.callback_filter_titulaciones_gestor

Funciones

- def callbacks.gestor.filters.callback_filter_titulaciones_gestor.update_filter_titulaciones_gestor (docente_id, curso_academico, n_clicks, existing_options)

6.142. callback_filter_titulaciones_gestor.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file callback_filter_titulaciones_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar las opciones
00004 #       del filtro de titulaciones en la pestaña "Indicadores académicos"
00005 #       del perfil "Gestor".
00006 # @version 1.0
00007 # @date 21/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, State, callback, callback_context
00014 from data.queries import universidades_gestor, titulaciones_universidad_gestor
00015
00016
00017 @callback(
00018     Output("titulaciones-gestor", "options"),
00019     Output("titulaciones-gestor", "value"),
00020     Input("selected-gestor-store", "data"),
00021     Input("curso-academico-gestor", "value"),
00022     Input("select-all-titulaciones-gestor", "n_clicks"),
00023     State("titulaciones-gestor", "options"),
00024 )
00025 def update_filter_titulaciones_gestor(docente_id, curso_academico, n_clicks, existing_options):
00026     """
00027 Actualiza las opciones del filtro de titulaciones en base a la universidad y curso académico
00028 seleccionados.
00029 También gestiona el evento que permite seleccionar todas las opciones del filtro del perfil "Gestor"
00030 em
00031     la pestaña "Indicadores académicos".
00032
00033     Args:
00034         docente_id (str): ID del docente seleccionado
00035         curso_academico (str): Curso académico seleccionado
00036         n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00037         existing_options (list): Opciones actuales del filtro de titulaciones
00038
00039     Returns:
00040         list: Opciones del dropdown
00041         list: Valores seleccionados
00042     """
00043     ctx = callback_context
00044     trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00045     if trigger_id == "select-all-titulaciones-gestor" and n_clicks > 0:
00046         return existing_options, [option["value"] for option in existing_options]
00047
```

```
00047     if not (docente_id and curso_academico):
00048         return [], None
00049
00050     cod_universidad = universidades_gestor(docente_id)
00051
00052     if not cod_universidad:
00053         return [], None
00054
00055     data = titulaciones_universidad_gestor(cod_universidad[0][0], curso_academico)
00056
00057     if not data:
00058         return [], None
00059
00060     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00061     value = (
00062         [option["value"] for option in opciones_dropdown] if opciones_dropdown else None
00063     )
00064
00065     return opciones_dropdown, value
```

6.143. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/← Universidad/6_curso/BI_Universities_← TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_← egresados_genero_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.graphs
- namespace callbacks.gestor.graphs.indicadores
- namespace callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor

Funciones

- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.update_graph_gestor (gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.toggle_modal (btn, is_← open)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.update_table (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.generate_csv (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor.get_data (gestor_id, curso_academico, titulaciones)

6.144. callback_graph_egresados_genero_gestor.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_graph_egresados_genero_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #       de alumnos egresados por género y titulación del perfil "Gestor"
00005 #       de la pestaña "Indicadores académicos".
00006 # @version 1.0
00007 # @date 21/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
```

```

00011 #
00012
00013 from dash import Input, Output, State, callback
00014 import plotly.graph_objects as go
00015 import dash_bootstrap_components as dbc
00016 import pandas as pd
00017 from util import list_to_tuple
00018 from data.queries import alumnos_egresados_genero_titulacion, universidades_gestor
00019
00020
00021 @callback(
00022     Output("egresados-genero-gestor", "figure"),
00023     Input("selected-gestor-store", "data"),
00024     Input("curso-academico-gestor", "value"),
00025     Input("titulaciones-gestor", "value"),
00026 )
00027 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00028     """
00029     Actualiza el gráfico de alumnos egresados por género y titulación
00030     del perfil "Gestor" de la pestaña "Indicadores académicos".
00031
00032     Args:
00033         gestor_id (str): ID del gestor seleccionado
00034         curso_academico (list): Lista con los cursos académicos
00035         titulaciones (list): Lista con las titulaciones seleccionadas
00036
00037     Returns:
00038         go.Figure: Figura con el gráfico
00039
00040     """
00041 fig = go.Figure()
00042
00043 fig.update_layout(
00044     barmode="stack",
00045     title={"text": "Alumnos egresados por género y titulación", "x": 0.5},
00046     xaxis=dict(title="Titulaciones"),
00047     yaxis=dict(title="Nº Alumnos"),
00048     showlegend=True,
00049     legend={"title": "Género"},
00050 )
00051
00052 df = get_data(gestor_id, curso_academico, titulaciones)
00053
00054 if df.empty:
00055     return fig
00056
00057 def map_genero(genero):
00058     if "masculin" in genero.lower():
00059         return "Hombres"
00060     elif "fem" in genero.lower():
00061         return "Mujeres"
00062     return genero
00063
00064 df["genero"] = df["genero"].map(map_genero)
00065
00066 # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
00067 df_pivot = df.pivot_table(
00068     index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
00069 ).fillna(0)
00070
00071 if "Hombres" in df_pivot.columns:
00072     fig.add_trace(
00073         go.Bar(
00074             x=df_pivot.index,
00075             y=df_pivot["Hombres"],
00076             name="Hombres",
00077             marker_color="blue",
00078             opacity=0.7,
00079         )
00080     )
00081
00082 if "Mujeres" in df_pivot.columns:
00083     fig.add_trace(
00084         go.Bar(
00085             x=df_pivot.index,
00086             y=df_pivot["Mujeres"],
00087             name="Mujeres",
00088             marker_color="red",
00089             opacity=0.7,
00090         )
00091     )
00092
00093 return fig
00094
00095
00096 @callback(
00097     Output("modal-egresados-genero", "is_open"),

```

```
00098     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00099     State("modal-egresados-genero", "is_open"),
00100 )
00101 def toggle_modal(btn, is_open):
00102     """
00103 Alterna la visibilidad del modal con los datos de alumnos egresados por género
00104 y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00105
00106     Args:
00107         btn (int): Número de clicks en el botón
00108         is_open (bool): Estado actual del modal
00109
00110     Returns:
00111         bool: Nuevo estado del modal
00112
00113     """
00114
00115 if btn:
00116     return not is_open
00117 return is_open
00118
00119
00120 @callback(
00121     Output("table-container-egresados-genero", "children"),
00122     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00123     Input("selected-gestor-store", "data"),
00124     Input("curso-academico-gestor", "value"),
00125     Input("titulaciones-gestor", "value"),
00126 )
00127 def update_table(btn, gestor_id, curso_academico, titulaciones):
00128     """
00129 Actualiza la tabla con los datos de alumnos egresados por género y titulación
00130 del perfil "Gestor" de la pestaña "Indicadores académicos".
00131
00132     Args:
00133         btn (int): Número de clicks en el botón
00134         gestor_id (str): ID del gestor seleccionado
00135         curso_academico (list): Lista con los cursos académicos
00136         titulaciones (list): Lista con las titulaciones seleccionadas
00137
00138     Returns:
00139         dbc.Table: Tabla con los datos
00140
00141     """
00142
00143 if not btn:
00144     return ""
00145
00146 df = get_data(gestor_id, curso_academico, titulaciones)
00147
00148 if df.empty:
00149     return dbc.Alert("No hay datos disponibles", color="info")
00150
00151 return dbc.Table.from_dataframe(
00152     df.head(50), striped=True, bordered=True, hover=True
00153 )
00154
00155
00156 @callback(
00157     Output("btn-descargar-egresados-genero", "href"),
00158     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00159     State("selected-gestor-store", "data"),
00160     State("curso-academico-gestor", "value"),
00161     State("titulaciones-gestor", "value"),
00162 )
00163 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00164     """
00165 Genera un archivo CSV descargable con los datos de alumnos egresados por género
00166 y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00167
00168     Args:
00169         btn (int): Número de clicks en el botón
00170         gestor_id (str): ID del gestor seleccionado
00171         curso_academico (list): Lista con los cursos académicos
00172         titulaciones (list): Lista con las titulaciones seleccionadas
00173
00174     Returns:
00175         str: Enlace al archivo CSV
00176
00177     """
00178
00179 if not btn:
00180     return ""
00181
00182 df = get_data(gestor_id, curso_academico, titulaciones)
00183
00184 if df.empty:
```

```

00185         return ""
00186
00187     csv_string = df.to_csv(index=False, encoding="utf-8")
00188     csv_string = "data:text/csv;charset=utf-8," + csv_string
00189     return csv_string
00190
00191
00192 def get_data(gestor_id, curso_academico, titulaciones):
00193     """
00194     Obtiene los datos de la base de datos.
00195
00196     Args:
00197         gestor_id (str): ID del gestor seleccionado
00198         curso_academico (list): Lista con los cursos académicos
00199         titulaciones (list): Lista con las titulaciones seleccionadas
00200
00201     Returns:
00202         pd.DataFrame: Datos obtenidos de la base de datos
00203     """
00204     empty = pd.DataFrame()
00205
00206     if not gestor_id or not curso_academico or not titulaciones:
00207         return empty
00208
00209     try:
00210         titulaciones = list_to_tuple(titulaciones)
00211     except Exception as e:
00212         return empty
00213
00214     data_universidad = universidades_gestor(gestor_id)
00215     if not data_universidad:
00216         return empty
00217
00218     data = alumnos_egresados_genero_titulacion(
00219         data_universidad[0][0], curso_academico, titulaciones
00220     )
00221
00222     if not data:
00223         return empty
00224
00225     return pd.DataFrame(data, columns=["titulacion", "genero", "cantidad"])

```

6.145. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_egresados_nacionalidad_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.graphs
- namespace callbacks.gestor.graphs.indicadores
- namespace callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor

Funciones

- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.update_graph_gestor (gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.toggle_modal (btn, is_open)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.update_table (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.generate_csv (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor.get_data (gestor_id, curso_academico, titulaciones)

6.146. callback_graph_egresados_nacionalidad_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_egresados_nacionalidad_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #     de alumnos egresados por nacionalidad y titulación del perfil "Gestor"
00005 #     de la pestaña "Indicadores académicos".
00006 # @version 1.0
00007 # @date 22/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martin
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, State, callback
00014 import plotly.graph_objs as go
00015 import dash_bootstrap_components as dbc
00016 import pandas as pd
00017 from util import list_to_tuple
00018 from data.queries import alumnos_egresados_nacionalidad_titulacion, universidades_gestor
00019
00020
00021 @callback(
00022     Output("egresados-nacionalidad-gestor", "figure"),
00023     Input("selected-gestor-store", "data"),
00024     Input("curso-academico-gestor", "value"),
00025     Input("titulaciones-gestor", "value"),
00026 )
00027 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00028 """
00029 Actualiza el gráfico de alumnos egresados por nacionalidad y titulación
00030 del perfil "Gestor" de la pestaña "Indicadores académicos".
00031
00032     Args:
00033         gestor_id (str): ID del gestor seleccionado
00034         curso_academico (list): Lista con los cursos académicos
00035         titulaciones (list): Lista con las titulaciones seleccionadas
00036
00037     Returns:
00038         go.Figure: Figura con el gráfico
00039
00040 """
00041
00042 fig = go.Figure()
00043
00044 fig.update_layout(
00045     barmode="stack",
00046     title={"text": "Alumnos egresados por nacionalidad y titulación", "x": 0.5},
00047     xaxis=dict(title="Titulaciones"),
00048     yaxis=dict(title="Nº Alumnos"),
00049     showlegend=True,
00050     legend={"title": "Nacionalidad"},
00051 )
00052
00053 df = get_data(gestor_id, curso_academico, titulaciones)
00054
00055 if df.empty:
00056     return fig
00057
00058 df_pivot = df.pivot_table(
00059     index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
00060 ).fillna(0)
00061
00062 for nacionalidad in df_pivot.columns:
00063     fig.add_trace(
00064         go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
00065     )
00066
00067 return fig
00068
00069
00070 @callback(
00071     Output("modal-egresados-nacionalidad", "is_open"),
00072     Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
00073     State("modal-egresados-nacionalidad", "is_open"),
00074 )
00075 def toggle_modal(btn, is_open):
00076 """
00077 Alterna la visibilidad del modal con los datos de alumnos egresados por nacionalidad
00078
00079     Args:
00080         btn (int): Número de clicks en el botón
00081         is_open (bool): Estado actual del modal
00082

```

```

00083 Returns:
00084     bool: Nuevo estado del modal
00085 """
00086
00087     if btn:
00088         return not is_open
00089     return is_open
00090
00091
00092     @callback(
00093         Output("table-container-egresados-nacionalidad", "children"),
00094         Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
00095         State("selected-gestor-store", "data"),
00096         Input("curso-academico-gestor", "value"),
00097         Input("titulaciones-gestor", "value"),
00098     )
00099     def update_table(btn, gestor_id, curso_academico, titulaciones):
00100     """
00101     Actualiza la tabla con los datos de alumnos egresados por nacionalidad y titulación
00102     del perfil "Gestor" de la pestaña "Indicadores académicos".
00103
00104     Args:
00105         btn (int): Número de clicks en el botón
00106         gestor_id (str): ID del gestor seleccionado
00107         curso_academico (list): Lista con los cursos académicos
00108
00109     Returns:
00110         dbc.Table: Tabla con los datos
00111     """
00112
00113     if not btn:
00114         return ""
00115
00116     df = get_data(gestor_id, curso_academico, titulaciones)
00117
00118     if df.empty:
00119         return dbc.Alert("No hay datos disponibles", color="info")
00120
00121     return dbc.Table.from_dataframe(
00122         df.head(50), striped=True, bordered=True, hover=True
00123     )
00124
00125
00126     @callback(
00127         Output("btn-descargar-egresados-nacionalidad", "href"),
00128         Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
00129         State("selected-gestor-store", "data"),
00130         Input("curso-academico-gestor", "value"),
00131         Input("titulaciones-gestor", "value"),
00132     )
00133     def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00134     """
00135     Genera un archivo CSV descargable con los datos de alumnos egresados por nacionalidad
00136     y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00137
00138     Args:
00139         btn (int): Número de clicks en el botón
00140         gestor_id (str): ID del gestor seleccionado
00141         curso_academico (list): Lista con los cursos académicos
00142         titulaciones (list): Lista con las titulaciones seleccionadas
00143
00144     Returns:
00145         str: Enlace al archivo CSV
00146     """
00147
00148     if not btn:
00149         return ""
00150
00151     df = get_data(gestor_id, curso_academico, titulaciones)
00152
00153     if df.empty:
00154         return ""
00155
00156     csv_string = df.to_csv(index=False, encoding="utf-8")
00157     csv_string = "data:text/csv; charset=utf-8," + csv_string
00158     return csv_string
00159
00160
00161     def get_data(gestor_id, curso_academico, titulaciones):
00162     """
00163     Obtiene los datos de alumnos egresados por nacionalidad y titulación
00164     del perfil "Gestor" de la pestaña "Indicadores académicos".
00165
00166     Args:
00167         gestor_id (str): ID del gestor seleccionado
00168         curso_academico (list): Lista con los cursos académicos
00169         titulaciones (list): Lista con las titulaciones seleccionadas

```

```
00170
00171     Returns:
00172     pd.DataFrame: Datos de la consulta
00173     """
00174
00175     empty = pd.DataFrame()
00176
00177     if not gestor_id or not curso_academico or not titulaciones:
00178         return empty
00179
00180     data_universidad = universidades_gestor(gestor_id)
00181     if not data_universidad:
00182         return empty
00183
00184     try:
00185         titulaciones = list_to_tuple(titulaciones)
00186     except Exception as e:
00187         return empty
00188
00189     data = alumnos_egresados_nacionalidad_titulacion(
00190         data_universidad[0][0], curso_academico, titulaciones
00191     )
00192
00193     if not data:
00194         return empty
00195
00196     return pd.DataFrame(data, columns=["titulacion", "nacionalidad", "cantidad"])
```

6.147. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/↔ Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_↔ nuevo_ingreso_genero_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.graphs
- namespace callbacks.gestor.graphs.indicadores
- namespace callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor

Funciones

- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.update_graph_gestor (gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.toggle_modal (btn, is_open)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.update_table (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.generate_csv (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor.get_data (gestor_id, curso_academico, titulaciones)

6.148. callback_graph_nuevo_ingreso_genero_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_nuevo_ingreso_genero_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #     de alumnos de nuevo ingreso por género y titulación del perfil "Gestor"
00005 #     de la pestaña "Indicadores académicos".
00006 # @version 1.0
00007 # @date 21/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, State, callback
00014 import plotly.graph_objs as go
00015 import dash_bootstrap_components as dbc
00016 import pandas as pd
00017 from util import list_to_tuple
00018 from data.queries import alumnos_nuevo_ingreso_genero_titulacion, universidades_gestor
00019
00020
00021 @callback(
00022     Output("nuevo-ingreso-genero-gestor", "figure"),
00023     Input("selected-gestor-store", "data"),
00024     Input("curso-academico-gestor", "value"),
00025     Input("titulaciones-gestor", "value"),
00026 )
00027 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00028 """
00029 Actualiza el gráfico de alumnos de nuevo ingreso por género y titulación
00030 del perfil "Gestor" de la pestaña "Indicadores académicos".
00031
00032     Args:
00033         gestor_id (str): ID del gestor seleccionado
00034         curso_academico (list): Lista con los cursos académicos
00035         titulaciones (list): Lista con las titulaciones seleccionadas
00036
00037     Returns:
00038         go.Figure: Figura con el gráfico
00039
00040 """
00041 fig = go.Figure()
00042
00043 fig.update_layout(
00044     barmode="stack",
00045     title={"text": "Alumnos de nuevo ingreso por género y titulación", "x": 0.5},
00046     xaxis=dict(title="Titulaciones"),
00047     yaxis=dict(title="Nº Alumnos"),
00048     showlegend=True,
00049     legend={"title": "Género"},
00050 )
00051
00052     # Asegurarse de que las columnas sean consistentes
00053 def map_genero(genero):
00054     if "masculin" in genero.lower():
00055         return "Hombres"
00056     elif "fem" in genero.lower():
00057         return "Mujeres"
00058     return genero
00059
00060 df = get_data(gestor_id, curso_academico, titulaciones)
00061
00062 if df.empty:
00063     return fig
00064
00065 df["genero"] = df["genero"].map(map_genero)
00066
00067     # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
00068 df_pivot = df.pivot_table(
00069     index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
00070 ).fillna(0)
00071
00072     # Asegurarse de que las columnas existen antes de graficar
00073 if "Hombres" in df_pivot.columns:
00074     fig.add_trace(
00075         go.Bar(
00076             x=df_pivot.index,
00077             y=df_pivot["Hombres"],
00078             name="Hombres",
00079             marker_color="blue",
00080             opacity=0.7,
00081         )
00082     )

```

```

00083
00084     if "Mujeres" in df_pivot.columns:
00085         fig.add_trace(
00086             go.Bar(
00087                 x=df_pivot.index,
00088                 y=df_pivot["Mujeres"],
00089                 name="Mujeres",
00090                 marker_color="red",
00091                 opacity=0.7,
00092             )
00093         )
00094
00095     return fig
00096
00097
00098 @callback(
00099     Output("modal-nuevo-ingreso-genero", "is_open"),
00100    Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00101    State("modal-nuevo-ingreso-genero", "is_open"),
00102 )
00103 def toggle_modal(btn, is_open):
00104     """
00105 Alterna la visibilidad del modal que muestra los datos de los alumnos de nuevo ingreso
00106
00107 Args:
00108 btn (int): Número de clicks en el botón
00109 is_open (bool): Estado actual del modal
00110
00111 Returns:
00112 bool: Nuevo estado del modal
00113 """
00114 if btn:
00115     return not is_open
00116 return is_open
00117
00118
00119 @callback(
00120     Output("table-container-nuevo-ingreso-genero", "children"),
00121     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00122     Input("selected-gestor-store", "data"),
00123     Input("curso-academico-gestor", "value"),
00124     Input("titulaciones-gestor", "value"),
00125 )
00126 def update_table(btn, gestor_id, curso_academico, titulaciones):
00127     """
00128 Actualiza la tabla de datos de los alumnos de nuevo ingreso por género y titulación
00129 del perfil "Gestor" de la pestaña "Indicadores académicos".
00130
00131 Args:
00132 btn (int): Número de clicks en el botón
00133 gestor_id (str): ID del gestor seleccionado
00134 curso_academico (list): Lista con los cursos académicos
00135 titulaciones (list): Lista con las titulaciones seleccionadas
00136
00137 Returns:
00138 dbc.Table: Tabla con los datos
00139 """
00140
00141 if not btn:
00142     return ""
00143
00144 df = get_data(gestor_id, curso_academico, titulaciones)
00145
00146 if df.empty:
00147     return dbc.Alert("No hay datos disponibles", color="info")
00148
00149 return dbc.Table.from_dataframe(
00150     df.head(50), striped=True, bordered=True, hover=True
00151 )
00152
00153
00154 @callback(
00155     Output("btn-descargar-nuevo-ingreso-genero", "href"),
00156     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00157     State("selected-gestor-store", "data"),
00158     State("curso-academico-gestor", "value"),
00159     State("titulaciones-gestor", "value"),
00160 )
00161 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00162     """
00163 Genera un archivo CSV descargable con los datos de los alumnos de nuevo ingreso
00164 por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00165
00166 Args:
00167 btn (int): Número de clicks en el botón
00168 gestor_id (str): ID del gestor seleccionado
00169 curso_academico (list): Lista con los cursos académicos

```

```

00170     titulaciones (list): Lista con las titulaciones seleccionadas
00171
00172     Returns:
00173     str: Contenido del archivo CSV
00174     """
00175
00176     if not btn:
00177         return ""
00178
00179     df = get_data(gestor_id, curso_academico, titulaciones)
00180     if df.empty:
00181         return ""
00182
00183     csv_string = df.to_csv(index=False, encoding="utf-8")
00184     csv_string = "data:text/csv;charset=utf-8," + csv_string
00185     return csv_string
00186
00187
00188 def get_data(gestor_id, curso_academico, titulaciones):
00189     """
00190     Obtiene los datos de la base da datos para el gráfico y tabla
00191
00192     Args:
00193     gestor_id (str): ID del gestor seleccionado
00194     curso_academico (list): Lista con los cursos académicos
00195     titulaciones (list): Lista con las titulaciones seleccionadas
00196
00197     Returns:
00198     pd.DataFrame: Datos para el gráfico y tabla
00199     """
00200
00201     empty = pd.DataFrame()
00202
00203     if not gestor_id or not curso_academico or not titulaciones:
00204         return empty
00205
00206     try:
00207         titulaciones = list_to_tuple(titulaciones)
00208     except Exception as e:
00209         return empty, None
00210
00211     data_universidad = universidades_gestor(gestor_id)
00212     if not data_universidad:
00213         return empty
00214
00215     data = alumnos_nuevo_ingreso_genero_titulacion(
00216         curso_academico, titulaciones, data_universidad[0][0]
00217     )
00218
00219     if not data:
00220         return empty
00221
00222     return pd.DataFrame(
00223         data, columns=["curso_academico", "titulacion", "genero", "cantidad"]
00224     )

```

6.149. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_nuevo_ingreso_nacionalidad_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.graphs
- namespace callbacks.gestor.graphs.indicadores
- namespace callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor

Funciones

- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.update_graph_gestor (gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.toggle_modal (btn, is_open)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.update_table (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.generate_csv (btn, gestor_id, curso_academico, titulaciones)
- def callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor.get_data (gesto_id, curso_academico, titulaciones)

6.150. callback_graph_nuevo_ingreso_nacionalidad_gestor.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file callback_graph_nuevo_ingreso_nacionalidad_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #     de alumnos de nuevo ingreso
00005 #     por nacionalidad y titulación del perfil "Gestor"
00006 #     de la pestaña "Indicadores académicos".
00007 # @version 1.0
00008 # @date 22/05/2024
00009 # @license MIT License
00010 # @author Fabrizio Daniell Perilli Martín
00011 # @email alu0101138589@ull.edu.es
00012 #
00013
00014 from dash import Input, Output, State, callback
00015 import plotly.graph_objs as go
00016 import dash_bootstrap_components as dbc
00017 import pandas as pd
00018 from util import list_to_tuple
00019 from data.queries import (
00020     alumnos_nuevo_ingreso_nacionalidad_titulacion,
00021     universidades_gestor,
00022 )
00023
00024
00025 @callback(
00026     Output("nuevo_ingreso_nacionalidad-gestor", "figure"),
00027     Input("selected-gestor-store", "data"),
00028     Input("curso-academico-gestor", "value"),
00029     Input("titulaciones-gestor", "value"),
00030 )
00031 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00032 """
00033 Actualiza el gráfico de alumnos de nuevo ingreso por nacionalidad y titulación
00034 del perfil "Gestor" de la pestaña "Indicadores académicos".
00035
00036 Args:
00037     gestor_id (str): ID del gestor seleccionado
00038     curso_academico (list): Lista con los cursos académicos
00039     titulaciones (list): Lista con las titulaciones seleccionadas
00040
00041 Returns:
00042     go.Figure: Figura con el gráfico
00043
00044 """
00045
00046 fig = go.Figure()
00047
00048 fig.update_layout(
00049     barmode="stack",
00050     title={
00051         "text": "Alumnos de nuevo ingreso por nacionalidad y titulación",
00052         "x": 0.5,
00053     },
00054     xaxis_title="Titulaciones",
00055     yaxis_title="Nº Alumnos",
00056     showlegend=True,
00057     legend={"title": "Nacionalidad"},
00058 )
00059
```

```

00060     df = get_data(gestor_id, curso_academico, titulaciones)
00061
00062     if df.empty:
00063         return fig
00064
00065     # Pivotear el DataFrame para obtener las cantidades por nacionalidad en columnas separadas
00066     df_pivot = df.pivot_table(
00067         index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
00068     ).fillna(0)
00069
00070     for nacionalidad in df_pivot.columns:
00071         fig.add_trace(
00072             go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
00073         )
00074
00075     return fig
00076
00077
00078 @callback(
00079     Output("modal-nuevo-ingreso-nacionalidad", "is_open"),
00080     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00081     State("modal-nuevo-ingreso-nacionalidad", "is_open"),
00082 )
00083 def toggle_modal(btn, is_open):
00084     """
00085 Alterna la visibilidad del modal con la tabla de datos de alumnos de nuevo ingreso
00086 por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00087
00088     Args:
00089         btn (int): Número de clicks en el botón "Ver datos"
00090         is_open (bool): Estado actual del modal
00091
00092     Returns:
00093         bool: Nuevo estado del modal
00094
00095     """
00096
00097     if btn:
00098         return not is_open
00099     return is_open
00100
00101
00102 @callback(
00103     Output("table-container-nuevo-ingreso-nacionalidad", "children"),
00104     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00105     State("selected-gestor-store", "data"),
00106     Input("curso-academico-gestor", "value"),
00107     Input("titulaciones-gestor", "value"),
00108 )
00109 def update_table(btn, gestor_id, curso_academico, titulaciones):
00110     """
00111 Actualiza la tabla de datos de alumnos de nuevo ingreso por nacionalidad y titulación
00112 del perfil "Gestor" de la pestaña "Indicadores académicos".
00113
00114     Args:
00115         btn (int): Número de clicks en el botón "Ver datos"
00116         gestor_id (str): ID del gestor seleccionado
00117         curso_academico (list): Lista con los cursos académicos
00118         titulaciones (list): Lista con las titulaciones seleccionadas
00119
00120     Returns:
00121         dbc.Table: Tabla con los datos
00122
00123     """
00124
00125     if not btn:
00126         return ""
00127
00128     df = get_data(gestor_id, curso_academico, titulaciones)
00129
00130     if df.empty:
00131         return dbc.Alert("No hay datos disponibles", color="info")
00132
00133     return dbc.Table.from_dataframe(
00134         df.head(50), striped=True, bordered=True, hover=True
00135     )
00136
00137
00138 @callback(
00139     Output("btn-descargar-nuevo-ingreso-nacionalidad", "href"),
00140     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00141     State("selected-gestor-store", "data"),
00142     Input("curso-academico-gestor", "value"),
00143     Input("titulaciones-gestor", "value"),
00144 )
00145 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00146     """

```

6.151 Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/resultados/callback_graph_duracion_media_estudios_nota_gestor.py291

```
00147 Genera un archivo CSV descargable con los datos de alumnos de nuevo ingreso
00148 por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00149
00150     Args:
00151         btn (int): Número de clicks en el botón "Ver datos"
00152         gestor_id (str): ID del gestor seleccionado
00153         curso_academico (list): Lista con los cursos académicos
00154         titulaciones (list): Lista con las titulaciones seleccionadas
00155
00156     Returns:
00157         str: Enlace al archivo CSV
00158
00159     """
00160
00161 if not btn:
00162     return ""
00163
00164     df = get_data(gestor_id, curso_academico, titulaciones)
00165
00166     if df.empty:
00167         return ""
00168
00169     csv_string = df.to_csv(index=False, encoding="utf-8")
00170     csv_string = "data:text/csv;charset=utf-8," + csv_string
00171     return csv_string
00172
00173
00174 def get_data(gesto_id, curso_academico, titulaciones):
00175     """
00176     Obtiene los datos de la base da datos para el gráfico y tabla.
00177
00178     Args:
00179         gesto_id (str): ID del gestor seleccionado
00180         curso_academico (list): Lista con los cursos académicos
00181         titulaciones (list): Lista con las titulaciones seleccionadas
00182
00183     Returns:
00184         pd.DataFrame: Datos para el gráfico y tabla
00185
00186     """
00187
00188     empty = pd.DataFrame()
00189
00190     if not gesto_id or not curso_academico or not titulaciones:
00191         return empty
00192
00193     data_universidad = universidades_gestor(gesto_id)
00194     if not data_universidad:
00195         return empty
00196
00197     try:
00198         titulaciones = list_to_tuple(titulaciones)
00199     except Exception as e:
00200         return empty
00201
00202     data = alumnos_nuevo_ingreso_nacionalidad_titulacion(
00203         data_universidad[0][0], curso_academico, titulaciones
00204     )
00205
00206     if not data:
00207         return empty
00208
00209     return pd.DataFrame(data, columns=["titulacion", "nacionalidad", "cantidad"])
```

6.151. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/↔ Universidad/6_curso/BI_Universities_↔ TFG/src/callbacks/gestor/graphs/resultados/callback_graph_↔ duracion_media_estudios_nota_gestor.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.gestor](#)
- namespace [callbacks.gestor.graphs](#)
- namespace [callbacks.gestor.graphs.resultados](#)
- namespace [callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor](#)

Funciones

- def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.update_graph_gestor(gestor_id)
- def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.toggle_modal(btn, is_open)
- def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.update_table(btn, gestor_id)
- def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.generate_csv(btn, gestor_id)
- def callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor.get_data(gestor_id)

6.152. callback_graph_duracion_media_estudios_nota_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_duracion_media_estudios_nota_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #     de duración media de los estudios con respecto a la nota media
00005 #     del perfil "Gestor" de la pestaña "Resultados académicos".
00006 # @version 1.0
00007 # @date 25/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, State, callback
00014 import plotly.express as px
00015 import plotly.graph_objects as go
00016 import dash_bootstrap_components as dbc
00017 import pandas as pd
00018 from data.queries import duracion_media_estudios_nota_gestor, universidades_gestor
00019
00020
00021 @callback(
00022     Output("duracion-estudios-nota-media-gestor", "figure"),
00023     Input("selected-gestor-store", "data"),
00024 )
00025 def update_graph_gestor(gestor_id):
00026     """
00027 Actualiza el gráfico de duración media de los estudios con respecto a la nota media
00028 del perfil "Gestor" de la pestaña "Resultados académicos".
00029
00030     Args:
00031         gestor_id (str): ID del gestor seleccionado
00032
00033     Returns:
00034         go.Figure: Figura con el gráfico
00035     """
00036
00037 fig = go.Figure()
00038
00039 if not gestor_id:
00040     return fig
00041
00042 data = get_data(gestor_id)
00043
00044 if data.empty:
00045     return fig
00046
00047 fig = px.scatter(
00048     data,
00049     x="nota_media",
00050     y="duracion_media",
00051     color="rama",
00052     size="numero_alumnos",
00053     hover_name="titulacion",
00054     animation_frames="curso_academico",
00055     animation_group="rama",
00056     category_orders={
00057         "rama": sorted(data["rama"].unique())
00058     },
00059 )

```

```

00060
00061     fig.update_layout(
00062         title={
00063             "text": "Duración media de los estudios con respecto a la nota media",
00064             "x": 0.5,
00065         },
00066         xaxis_title="Nota media",
00067         yaxis_title="Duración media de los estudios",
00068         legend={"title": "Rama de conocimiento"},
00069     )
00070
00071     fig.update_traces(
00072         marker=dict(line=dict(width=1, color="DarkSlateGrey")),
00073         textposition="top center"
00074     )
00075
00076     # Ajustar el rango de los ejes para que las burbujas no se corten
00077     fig.update_xaxes(range=[data["nota_media"].min() - 1, data["nota_media"].max() + 1])
00078     fig.update_yaxes(
00079         range=[data["duracion_media"].min() - 1, data["duracion_media"].max() + 1]
00080     )
00081
00082     if len(data["curso_academico"].unique()) > 1:
00083         fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 1000
00084
00085     return fig
00086
00087
00088 @callback(
00089     Output("modal-duracion-estudios", "is_open"),
00090     Input("btn-ver-datos-duracion-estudios", "n_clicks"),
00091     State("modal-duracion-estudios", "is_open"),
00092 )
00093 def toggle_modal(btn, is_open):
00094     """
00095 Alternar la visibilidad del modal con los datos de duración media de los estudios
00096 con respecto a la nota media.
00097
00098 Args:
00099     btn (int): Número de clicks en el botón
00100    is_open (bool): Estado actual del modal
00101
00102 Returns:
00103    bool: Nuevo estado del modal
00104 """
00105 if btn:
00106     return not is_open
00107     return is_open
00108
00109
00110 @callback(
00111     Output("table-container-duracion-estudios", "children"),
00112     Input("btn-ver-datos-duracion-estudios", "n_clicks"),
00113     State("selected-gestor-store", "data"),
00114 )
00115 def update_table(btn, gestor_id):
00116     """
00117 Actualiza la tabla con los datos de duración media de los estudios
00118 con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".
00119
00120 Args:
00121     btn (int): Número de clicks en el botón
00122     gestor_id (str): ID del gestor seleccionado
00123
00124 Returns:
00125     dbc.Table: Tabla con los datos
00126 """
00127 if not btn:
00128     return ""
00129
00130     df = get_data(gestor_id)
00131
00132     if df.empty:
00133         return dbc.Alert("No hay datos disponibles", color="info")
00134
00135     return dbc.Table.from_dataframe(
00136         df.head(50), striped=True, bordered=True, hover=True
00137     )
00138
00139
00140 @callback(
00141     Output("btn-descargar-csv-duracion-estudios", "href"),
00142     Input("btn-descargar-csv-duracion-estudios", "n_clicks"),
00143     State("selected-gestor-store", "data"),
00144 )
00145 def generate_csv(btn, gestor_id):
00146     """

```

```

00147 Genera un archivo CSV descargable con los datos de duración media de los estudios
00148 con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".
00149
00150     Args:
00151         btn (int): Número de clicks en el botón
00152         gestor_id (str): ID del gestor seleccionado
00153
00154     Returns:
00155         str: Enlace al archivo CSV
00156
00157
00158 if not btn:
00159     return ""
00160
00161 df = get_data(gestor_id)
00162
00163 if df.empty:
00164     return ""
00165
00166 csv_string = df.to_csv(index=False, encoding="utf-8")
00167 csv_string = "data:text/csv; charset=utf-8," + csv_string
00168 return csv_string
00169
00170
00171 def get_data(gestor_id):
00172     """
00173 Obtiene los datos de la base de datos.
00174
00175     Args:
00176         gestor_id (str): ID del gestor seleccionado
00177
00178     Returns:
00179         pd.DataFrame: Datos de la base de datos
00180     """
00181 empty = pd.DataFrame()
00182
00183 if not gestor_id:
00184     return empty
00185
00186 data_universidad = universidades_gestor(gestor_id)
00187 if not data_universidad:
00188     return empty
00189
00190 data = duracion_media_estudios_nota_gestor(data_universidad[0][0])
00191 if not data:
00192     return empty
00193
00194 df = pd.DataFrame(
00195     data,
00196     columns=[
00197         "nota_media",
00198         "titulacion",
00199         "rama",
00200         "curso_academico",
00201         "duracion_media",
00202         "numero_alumnos",
00203     ],
00204 )
00205 return df

```

6.153. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/callback_graph_nota_acceso_titulacion_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.graphs
- namespace callbacks.gestor.graphs.resultados
- namespace callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor

Funciones

- def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.update_grph_gestor (gestor_id)
- def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.toggle_modal (btn, is_open)
- def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.update_table (btn, gestor_id)
- def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.generate_csv (btn, gestor_id)
- def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.get_data (gestor_id)

6.154. callback_graph_nota_acceso_titulacion_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_nota_acceso_titulacion_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #       de evolución de la nota de acceso por titulación
00005 #       del perfil "Gestor" de la pestaña "Resultados académicos".
00006 # @version 1.0
00007 # @date 22/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martin
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, State, callback
00014 import plotly.graph_objs as go
00015 import dash_bootstrap_components as dbc
00016 import pandas as pd
00017 from data.queries import nota_media_acceso_titulacion, universidades_gestor
00018
00019
00020 @callback(
00021     Output("nota-acceso-titulacion", "figure"),
00022     Input("selected-gestor-store", "data")
00023 )
00024 def update_grph_gestor(gestor_id):
00025     """
00026 Actualiza el gráfico de evolución de la nota de acceso por titulación
00027 del perfil "Gestor" de la pestaña "Resultados académicos".
00028
00029     Args:
00030         gestor_id (str): ID del gestor seleccionado
00031
00032     Returns:
00033         go.Figure: Figura con el gráfico
00034
00035     """
00036
00037 fig = go.Figure()
00038
00039 fig.update_layout(
00040 title={"text": "Evolución nota de acceso por titulación", "x": 0.5},
00041     xaxis_title="Curso académico",
00042     yaxis_title="Nota media de acceso",
00043     legend={"title": "Titulaciones"},
00044 )
00045
00046 df = get_data(gestor_id)
00047
00048 if df.empty:
00049     return fig
00050
00051 for titulacion, group in df.groupby("titulacion"):
00052     fig.add_trace(
00053         go.Scatter(
00054             x=group["curso_academico"],
00055             y=group["nota"],
00056             mode="lines+markers",
00057             name=titulacion,
00058         )
00059     )
00060
00061 return fig

```

```
00062
00063
00064     @callback(
00065         Output("modal-nota-acceso", "is_open"),
00066         Input("btn-ver-datos-nota-acceso", "n_clicks"),
00067         State("modal-nota-acceso", "is_open"),
00068     )
00069     def toggle_modal(btn, is_open):
00070         """
00071             Alternar la visibilidad del modal de datos de la nota de acceso por titulación
00072             del perfil "Gestor" de la pestaña "Resultados académicos".
00073
00074             Args:
00075                 btn (int): Número de clicks en el botón
00076                 is_open (bool): Estado actual del modal
00077
00078             Returns:
00079                 bool: Nuevo estado del modal
00080
00081         """
00082     if btn:
00083         return not is_open
00084     return is_open
00085
00086
00087     @callback(
00088         Output("table-container-nota-acceso", "children"),
00089         Input("btn-ver-datos-nota-acceso", "n_clicks"),
00090         State("selected-gestor-store", "data"),
00091     )
00092     def update_table(btn, gestor_id):
00093         """
00094             Actualiza la tabla con los datos de la nota de acceso por titulación
00095             del perfil "Gestor" de la pestaña "Resultados académicos".
00096
00097             Args:
00098                 btn (int): Número de clicks en el botón
00099                 gestor_id (str): ID del gestor seleccionado
00100
00101             Returns:
00102                 dbc.Table: Tabla con los datos
00103
00104         """
00105
00106     if not btn:
00107         return ""
00108
00109     df = get_data(gestor_id)
00110
00111     if df.empty:
00112         return dbc.Alert("No hay datos disponibles", color="info")
00113
00114     return dbc.Table.from_dataframe(
00115         df.head(50), striped=True, bordered=True, hover=True
00116     )
00117
00118
00119     @callback(
00120         Output("btn-descargar-csv-nota-acceso", "href"),
00121         Input("btn-ver-datos-nota-acceso", "n_clicks"),
00122         State("selected-gestor-store", "data"),
00123     )
00124     def generate_csv(btn, gestor_id):
00125         """
00126             Genera un archivo CSV con los datos de la nota de acceso por titulación
00127             del perfil "Gestor" de la pestaña "Resultados académicos".
00128
00129             Args:
00130                 btn (int): Número de clicks en el botón
00131                 gestor_id (str): ID del gestor seleccionado
00132
00133             Returns:
00134                 str: Enlace al archivo CSV
00135
00136         """
00137
00138     if not btn:
00139         return ""
00140
00141     df = get_data(gestor_id)
00142
00143     if df.empty:
00144         return ""
00145
00146     csv_string = df.to_csv(index=False, encoding="utf-8")
00147     csv_string = "data:text/csv;charset=utf-8," + csv_string
00148
00149     return csv_string
```

```
00149
00150
00151 def get_data(gestor_id):
00152     """
00153     Obtiene los datos de la base de datos.
00154
00155     Args:
00156         gestor_id (str): ID del gestor seleccionado
00157
00158     Returns:
00159         pd.DataFrame: Datos de la nota de acceso por titulación
00160     """
00161     empty = pd.DataFrame()
00162
00163     if not gestor_id:
00164         return empty
00165
00166     data_universidad = universidades_gestor(gestor_id)
00167
00168     if not data_universidad:
00169         return empty
00170
00171     data = nota_media_acceso_titulacion(data_universidad[0][0])
00172
00173     if not data:
00174         return empty
00175
00176     return pd.DataFrame(data, columns=["curso_academico", "titulacion", "nota"])
```

6.155. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo_abandono/callback_graph_tasa_abandono_gestor.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.gestor](#)
- namespace [callbacks.gestor.graphs](#)
- namespace [callbacks.gestor.graphs riesgo_abandono](#)
- namespace [callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor](#)

Funciones

- def [callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor.update_graph_gestor](#) (curso_academico, gestor_id)
- def [callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor.toggle_modal](#) (btn, is_open)
- def [callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor.update_table](#) (btn, gestor_id, curso_academico)
- def [callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor.generate_csv](#) (btn, gestor_id, curso_academico)
- def [callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono_gestor.get_data](#) (gestor_id, curso_academico)

6.156. callback_graph_tasa_abandono_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_tasa_abandono_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 #       de la tasa de abandono por titulación del perfil "Gestor"
00005 #       de la pestaña "Riesgo de abandono".
00006 # @version 1.0
00007 # @date 26/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martin
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Output, Input, State, callback
00014 import plotly.graph_objs as go
00015 import dash_bootstrap_components as dbc
00016 import pandas as pd
00017 from data.queries import tasa_abandono_titulacion_gestor, universidades_gestor
00018 from util import list_to_tuple
00019
00020
00021 @callback(
00022     Output("tasa-abandono-gestor", "figure"),
00023     Input("curso-all-academico-gestor", "value"),
00024     Input("selected-gestor-store", "data"),
00025 )
00026 def update_graph_gestor(curso_academico, gestor_id):
00027 """
00028 Actualiza el gráfico de la tasa de abandono por titulación
00029 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00030
00031     Args:
00032         curso_academico (list): Lista con los cursos académicos
00033         gestor_id (str): ID del gestor seleccionado
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037 """
00038 fig = go.Figure()
00039
00040 fig.update_layout(
00041     title={"text": "Tasa de abandono por titulación ", "x": 0.5},
00042         xaxis_title="Curso académico",
00043         yaxis_title="Tasa de abandono (%)",
00044         showlegend=True,
00045         legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
00046 )
00047
00048 df = get_data(gestor_id, curso_academico)
00049
00050 if df.empty:
00051     return fig
00052
00053 df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00054
00055 for titulacion in df["titulacion"].unique():
00056     df_titulacion = df[df["titulacion"] == titulacion]
00057     fig.add_trace(
00058         go.Scatter(
00059             x=df_titulacion["curso_academico"],
00060             y=df_titulacion["tasa_abandono"],
00061             mode="lines+markers",
00062             name=titulacion,
00063         )
00064     )
00065
00066 return fig
00067
00068
00069 @callback(
00070     Output("modal-tasa-abandono", "is_open"),
00071     Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00072     State("modal-tasa-abandono", "is_open"),
00073 )
00074 def toggle_modal(btn, is_open):
00075 """
00076 Alternar la visibilidad del modal de datos de la tasa
00077 de abandono por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".
00078
00079     Args:
00080         btn (int): Número de clicks en el botón
00081         is_open (bool): Estado actual del modal
00082

```

```
00083     Returns:
00084     bool: Nuevo estado del modal
00085     """
00086
00087     if btn:
00088         return not is_open
00089     return is_open
00090
00091
00092     @callback(
00093         Output("table-container-tasa-abandono", "children"),
00094         Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00095         State("selected-gestor-store", "data"),
00096         Input("curso-all-academico-gestor", "value"),
00097     )
00098     def update_table(btn, gestor_id, curso_academico):
00099     """
00100     Actualiza la tabla con los datos de la tasa de abandono por titulación
00101     del perfil "Gestor" de la pestaña "Riesgo de abandono".
00102
00103     Args:
00104     btn (int): Número de clicks en el botón
00105     gestor_id (str): ID del gestor seleccionado
00106     curso_academico (list): Lista con los cursos académicos
00107
00108     Returns:
00109     dbc.Table: Tabla con los datos
00110     """
00111
00112     if not btn:
00113         return ""
00114
00115     df = get_data(gestor_id, curso_academico)
00116
00117     if df.empty:
00118         return dbc.Alert("No hay datos disponibles", color="info")
00119
00120     df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00121     return dbc.Table.from_dataframe(
00122         df.head(50), striped=True, bordered=True, hover=True, responsive=True
00123     )
00124
00125
00126     @callback(
00127         Output("btn-descargar-tasa-abandono", "href"),
00128         Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00129         State("selected-gestor-store", "data"),
00130         Input("curso-all-academico-gestor", "value"),
00131     )
00132     def generate_csv(btn, gestor_id, curso_academico):
00133     """
00134     Genera un archivo CSV descargable con los datos de la tasa de abandono por titulación
00135     del perfil "Gestor" de la pestaña "Riesgo de abandono".
00136
00137     Args:
00138     btn (int): Número de clicks en el botón
00139     gestor_id (str): ID del gestor seleccionado
00140     curso_academico (list): Lista con los cursos académicos
00141
00142     Returns:
00143     str: Enlace al archivo CSV
00144     """
00145
00146     if not btn:
00147         return ""
00148     df = get_data(gestor_id, curso_academico)
00149
00150     if df.empty:
00151         return ""
00152
00153     df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00154     csv_string = df.to_csv(index=False, encoding="utf-8")
00155     csv_string = "data:text/csv;charset=utf-8," + csv_string
00156     return csv_string
00157
00158
00159     def get_data(gestor_id, curso_academico):
00160     """
00161     Obtiene los datos de la base de datos.
00162
00163     Args:
00164     gestor_id (str): ID del gestor seleccionado
00165     curso_academico (list): Lista con los cursos académicos
00166
00167     Returns:
00168     pd.DataFrame: Datos de la tasa de abandono por titulación
00169     """
```

```

00170 empty = pd.DataFrame()
00171
00172 if not gestor_id or not curso_academico:
00173     return empty
00174
00175     data_universidad = universidades_gestor(gestor_id)
00176     if not data_universidad:
00177         return empty
00178     try:
00179         curso_academico = list_to_tuple(curso_academico)
00180     except Exception as e:
00181         print("Error:", e)
00182         return empty
00183
00184     data = tasa_abandono_titulacion_gestor(data_universidad[0][0], curso_academico)
00185
00186     if not data:
00187         return empty
00188
00189     return pd.DataFrame(
00190         data,
00191         columns=[
00192             "curso_academico",
00193             "titulacion",
00194             "numero_matriculados",
00195             "numero_abandonos",
00196         ],
00197     )

```

6.157. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/callbacks/gestor/graphs/riesgo_←
abandono/callback_graph_tasa_graduacion_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.graphs
- namespace callbacks.gestor.graphs riesgo_abandono
- namespace callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor

Funciones

- def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.update_graph_gestor (curso_academico, gestor_id)
- def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.toggle_modal (btn, is_open)
- def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.update_table (btn, curso_academico, gestor_id)
- def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.generate_csv (n, curso_academico, gestor_id)
- def callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_graduacion_gestor.get_data (gestor_id, curso_academico)

6.158. callback_graph_tasa_graduacion_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_graph_tasa_graduacion_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el gráfico
00004 # de la tasa de graduación por titulación del perfil "Gestor"
00005 # de la pestaña "Riesgo de abandono".
00006 # @version 1.0
00007 # @date 27/05/2024
00008 # @license MIT License
00009 # @author Fabrizzio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012
00013 from dash import Input, Output, State, callback
00014 import plotly.graph_objs as go
00015 import dash_bootstrap_components as dbc
00016 import pandas as pd
00017 from util import list_to_tuple
00018 from data.queries import tasa_graduacion_titulacion_gestor, universidades_gestor
00019
00020
00021 @callback(
00022     Output("tasa-graduacion-gestor", "figure"),
00023     Input("curso-all-academico-gestor", "value"),
00024     Input("selected-gestor-store", "data"),
00025 )
00026 def update_graph_gestor(curso_academico, gestor_id):
00027 """
00028 Actualiza el gráfico de la tasa de graduación por titulación
00029 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00030
00031     Args:
00032         curso_academico (list): Lista con los cursos académicos
00033         gestor_id (str): ID del gestor seleccionado
00034
00035     Returns:
00036         go.Figure: Figura con el gráfico
00037 """
00038
00039 fig = go.Figure()
00040
00041 fig.update_layout(
00042     title={"text": "Tasa de graduación por titulación ", "x": 0.5},
00043     xaxis_title="Curso académico",
00044     yaxis_title="Tasa de graduación (%)",
00045     showlegend=True,
00046     legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
00047 )
00048
00049 df = get_data(gestor_id, curso_academico)
00050
00051 if df.empty:
00052     return fig
00053
00054 for titulacion, group in df.groupby("titulacion"):
00055     fig.add_trace(
00056         go.Scatter(
00057             x=group["curso_academico"],
00058             y=group["tasa_graduacion"],
00059             mode="lines+markers",
00060             name=titulacion,
00061         )
00062     )
00063
00064 return fig
00065
00066
00067 @callback(
00068     Output("modal-tasa-graduacion", "is_open"),
00069     Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
00070     State("modal-tasa-graduacion", "is_open"),
00071 )
00072 def toggle_modal(btn, is_open):
00073 """
00074 Alternar la visibilidad del modal de datos de la tasa de graduación por titulación
00075 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00076
00077     Args:
00078         btn (int): Número de clicks en el botón
00079         is_open (bool): Estado actual del modal
00080 """
00081
00082 if btn:

```

```

00083         return not is_open
00084     return is_open
00085
00086
00087     @callback(
00088         Output("table-container-tasa-graduacion", "children"),
00089         Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
00090         State("curso-all-academico-gestor", "value"),
00091         State("selected-gestor-store", "data"),
00092     )
00093     def update_table(btn, curso_academico, gestor_id):
00094         """
00095         Actualiza la tabla de datos de la tasa de graduación por titulación
00096         del perfil "Gestor" de la pestaña "Riesgo de abandono".
00097
00098         Args:
00099             btn (int): Número de clicks en el botón
00100            curso_academico (list): Lista con los cursos académicos
00101            gestor_id (str): ID del gestor seleccionado
00102
00103         Returns:
00104             dbc.Table: Tabla con los datos
00105         """
00106     if not btn:
00107         return ""
00108
00109     df = get_data(gestor_id, curso_academico)
00110
00111     if df.empty:
00112         return dbc.Alert("No hay datos disponibles", color="info")
00113
00114     return dbc.Table.from_dataframe(
00115         df.head(50), striped=True, bordered=True, hover=True, responsive=True
00116     )
00117
00118
00119     @callback(
00120         Output("btn-descargar-tasa-graduacion", "href"),
00121         Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
00122         State("curso-all-academico-gestor", "value"),
00123         State("selected-gestor-store", "data"),
00124     )
00125     def generate_csv(n, curso_academico, gestor_id):
00126         """
00127         Genera un archivo CSV descargable con los datos de la tasa de graduación por titulación
00128         del perfil "Gestor" de la pestaña "Riesgo de abandono".
00129
00130         Args:
00131             n (int): Número de clicks en el botón
00132             curso_academico (list): Lista con los cursos académicos
00133             gestor_id (str): ID del gestor seleccionado
00134         """
00135
00136     if not n:
00137         return None
00138
00139     df = get_data(gestor_id, curso_academico)
00140
00141     if df is None:
00142         return None
00143
00144     csv_string = df.to_csv(index=False, encoding="utf-8")
00145     csv_string = "data:text/csv;charset=utf-8," + csv_string
00146
00147     return csv_string
00148
00149
00150     def get_data(gestor_id, curso_academico):
00151         """
00152         Obtiene los datos de la tasa de graduación por titulación
00153         del perfil "Gestor" de la pestaña "Riesgo de abandono".
00154
00155         Args:
00156             gestor_id (str): ID del gestor seleccionado
00157             curso_academico (list): Lista con los cursos académicos
00158
00159         Returns:
00160             pd.DataFrame: Datos de la tasa de graduación
00161         """
00162
00163     empty = pd.DataFrame()
00164
00165     if not gestor_id or not curso_academico:
00166         return empty
00167
00168     try:
00169         curso_academico = list_to_tuple(curso_academico)

```

```
00170     except Exception as e:
00171         print("Error:", e)
00172         return empty
00173
00174     data_universidad = universidades_gestor(gestor_id)
00175
00176     if not data_universidad:
00177         return empty
00178
00179     data = tasa_graduacion_titulacion_gestor(data_universidad[0][0], curso_academico)
00180
00181     if not data:
00182         return empty
00183
00184     df = pd.DataFrame(
00185         data,
00186         columns=[
00187             "numero_matriculados",
00188             "numero_egresados",
00189             "curso_academico",
00190             "titulacion",
00191         ],
00192     )
00193     df["tasa_graduacion"] = (df["numero_egresados"] / df["numero_matriculados"]) * 100
00194
00195     return df
```

6.159. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/← Universidad/6_curso/BI_Universities_← TFG/src/callbacks/gestor/utils/callback_resumen_gestor.py

Namespaces

- namespace callbacks
- namespace callbacks.gestor
- namespace callbacks.gestor.utils
- namespace callbacks.gestor.utils.callback_resumen_gestor

Funciones

- def callbacks.gestor.utils.callback_resumen_gestor.update_resumen_gestor (gestor_id)
- def callbacks.gestor.utils.callback_resumen_gestor.not_data ()

6.160. callback_resumen_gestor.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file callback_resumen_gestor.py
00003 # @brief Este fichero contiene el callback para actualizar el resumen del gestor seleccionado.
00004 # @version 1.0
00005 # @date 21/05/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, Output, Input, callback
00012 from data.queries import numero_alumnos_matriculados_universidad, universidades_gestor
00013
00014
00015 @callback(
00016     Output("resumen-gestor", "children"),
00017     Input("selected-gestor-store", "data")
00018 )
00019 def update_resumen_gestor(gestor_id):
```

```

00020      """
00021 Actualiza el resumen del gestor seleccionado.
00022
00023 Args:
00024     gestor_id (str): ID del gestor seleccionado
00025
00026 Returns:
00027     html.Div: Resumen del gestor seleccionado
00028 """
00029
00030 if not gestor_id:
00031     return not_data()
00032
00033     data = universidades_gestor(gestor_id)
00034
00035     if not data:
00036         return not_data()
00037
00038     data_alumnos = numero_alumnos_matriculados_universidad(data[0][1])
00039
00040     if not data_alumnos:
00041         return not_data()
00042
00043     return html.Div(
00044         [
00045             html.H2("Resumen"),
00046             html.P("Universidad:", className="resumen-label"),
00047             html.P(data[0][1]),
00048             html.P("Gestor:", className="resumen-label"),
00049             html.P(gestor_id),
00050             html.P("Número de alumnos matriculados:", className="resumen-label"),
00051             html.P(data_alumnos[0][0]),
00052             html.Hr(),
00053         ]
00054     )
00055
00056
00057 def not_data():
00058     return html.Div(
00059         [
00060             html.H2("Resumen"),
00061             html.P("Universidad:", className="resumen-label"),
00062             html.P("No disponible"),
00063             html.P("Gestor:", className="resumen-label"),
00064             html.P("No disponible"),
00065             html.P("Número de alumnos matriculados:", className="resumen-label"),
00066             html.P("No disponible"),
00067             html.Hr(),
00068         ]
00069     )

```

6.161. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/callback_select_gestor.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.gestor](#)
- namespace [callbacks.gestor.utils](#)
- namespace [callbacks.gestor.utils.callback_select_gestor](#)

Funciones

- def [callbacks.gestor.utils.callback_select_gestor.store_selected_gestor](#) (selected_value, stored_value)

6.162. callback_select_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_select_gestor.py
00003 # @brief Este fichero contiene el callback para almacenar el gestor seleccionado en un store.
00004 # @version 1.0
00005 # @date 21/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import Input, Output, callback, State
00012 from data.queries import gestores_all
00013
00014
00015 @callback(
00016     Output("gestor-dropdown", "value"),
00017     Output("gestor-dropdown", "options"),
00018     Output("selected-gestor-store", "data"),
00019     Input("gestor-dropdown", "value"),
00020     State("selected-gestor-store", "data"),
00021 )
00022 def store_selected_gestor(selected_value, stored_value):
00023     """
00024 Almacena el gestor seleccionado en un store.
00025
00026 Args:
00027 selected_value (str): Valor seleccionado en el dropdown
00028 stored_value (dict): Datos almacenados en el store
00029
00030 Returns:
00031 str: Valor seleccionado en el dropdown
00032 list: Opciones del dropdown
00033 str: Valor almacenado en el store
00034 """
00035
00036 data = gestores_all()
00037
00038 if not data:
00039     return None, [], None
00040
00041 opciones_dropdown = [{"label": gestor[0], "value": gestor[0]} for gestor in data]
00042
00043 if selected_value is None and stored_value in [
00044     op["value"] for op in opciones_dropdown
00045 ]:
00046     return stored_value, opciones_dropdown, stored_value
00047
00048 return selected_value, opciones_dropdown, selected_value

```

6.163. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/callback_tabs_gestor.py

Namespaces

- namespace [callbacks](#)
- namespace [callbacks.gestor](#)
- namespace [callbacks.gestor.utils](#)
- namespace [callbacks.gestor.utils.callback_tabs_gestor](#)

Funciones

- def [callbacks.gestor.utils.callback_tabs_gestor.render_content](#)(tab, selected_gestor)

6.164. callback_tabs_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file callback_tabs_gestor.py
00003 # @brief Este fichero contiene el callback para renderizar el
00004 #       contenido de las pestañas de la sección "Gestor".
00005 # @version 1.0
00006 # @date 17/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import Input, Output, State, html, callback
00013 from components.common.sidebar import sidebar
00014 from components.common.filters import filters
00015 from components.gestor.utils.select_gestor import select_gestor
00016 from components.gestor.utils.resumen_gestor import resumen_gestor
00017 from components.gestor.utils.recomendador_gestor import recomendador_gestor
00018 from components.gestor.filters.filter_curso_academico_gestor import filter_curso_academico_gestor
00019 from components.gestor.filters.filter_titulaciones_gestor import filter_titulaciones_gestor
00020 from components.gestor.filters.filter_all_curso_academico_gestor import
00021     filter_all_curso_academico_gestor
00022 from components.gestor.graphs.graphs_indicadores_gestor import graphs_indicadores_gestor
00023 from components.gestor.graphs.graphs_resultados_gestor import graphs_resultados_gestor
00024 from components.gestor.graphs.graphs_riesgo_abandono_gestor import graphs_riesgo_abandono_gestor
00025
00026     @callback(
00027         Output('tabs-gestor-content', 'children'),
00028         Input('tabs-gestor', 'value'),
00029         State('selected-gestor-store', 'data')
00030     )
00031     def render_content(tab, selected_gestor):
00032         """
00033             Renderiza el contenido de las pestañas de la sección "Gestor".
00034
00035             Args:
00036                 tab (str): Pestaña seleccionada
00037                 selected_gestor (dict): Datos del gestor seleccionado
00038
00039             Returns:
00040                 html.Div: Contenido de la pestaña seleccionada
00041
00042         if tab == 'indicadores-academicos-tab':
00043             return html.Div([
00044                 select_gestor(),
00045                 html.H2("Dashboard Gestor", style={'text-align': 'center'}),
00046                 html.Div([
00047                     sidebar([
00048                         resumen_gestor(),
00049                         filters([
00050                             filter_curso_academico_gestor(),
00051                             filter_titulaciones_gestor()
00052                         ]),
00053                         ],
00054                         graphs_indicadores_gestor()
00055                     ], className='content-layout-dashboard')
00056
00057                 ])
00058         elif tab == 'resultados-academicos-tab':
00059             return html.Div([
00060                 html.H2("Dashboard Gestor", style={'text-align': 'center'}),
00061                 html.Div([
00062                     sidebar([
00063                         resumen_gestor(),
00064                         ],
00065                         graphs_resultados_gestor()
00066                     ], className='content-layout-dashboard')
00067                 ])
00068         elif tab == 'riesgo-abandono-tab':
00069             return html.Div([
00070                 html.H2("Dashboard Gestor", style={'text-align': 'center'}),
00071                 html.Div([
00072                     sidebar([
00073                         resumen_gestor(),
00074                         filters([
00075                             filter_all_curso_academico_gestor()
00076                         ]),
00077                         ],
00078                         graphs_riesgo_abandono_gestor()
00079                     ], className='content-layout-dashboard')
00080                 ])
00081         elif tab == 'recomendaciones-tab':

```

```
00082     return html.Div([
00083         recomendador_gestor()
00084     ])
00085
00086
00087
```

6.165. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/alumnado/filters/filter_←
asignaturas_matri_alumnado.py

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.filters
- namespace components.alumnado.filters.filter_asignaturas_matri_alumnado

Funciones

- def components.alumnado.filters.filter_asignaturas_matri_alumnado.filter_asignaturas_matri_alumnado ()

6.166. filter_asignaturas_matri_alumnado.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file filter_asignaturas_matri_alumnado.py
00003 # @brief Este archivo contiene componente para filtrar asignaturas matriculadas por el alumno.
00004 # @version 1.0
00005 # @date 07/05/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
0010
0011 from dash import html, dcc
0012 from callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado import
    update_filter_asignaturas_matri_alumnado
0013
0014
0015 def filter_asignaturas_matri_alumnado():
0016     """
0017 Crea un componente con un dropdown que contiene las asignaturas matriculadas por el alumno.
0018
0019 Returns:
0020 html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
0021 """
0022 return html.Div(
0023     [
0024         html.Br(),
0025         html.Label("Asignaturas matriculadas"),
0026         dcc.Dropdown(
0027             id="asignaturas-matriculadas",
0028             searchable=True,
0029             multi=True,
0030             value=None,
0031             clearable=True,
0032             options=[],
0033             maxHeight=300,
0034             optionHeight=50,
0035             placeholder="Seleccione una opción",
0036         ),
0037         html.Button(
0038             "Seleccionar todo",
0039             id="select-all-button",
0040             className="button-select-all-filter",
0041             n_clicks=0,
0042         ),
0043     ],
0044 )
```

6.167. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/alumnado/filters/filter_curso_←
academico_alumnado.py**

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.filters
- namespace components.alumnado.filters.filter_curso_academico_alumnado

Funciones

- def `components.alumnado.filters.filter_curso_academico_alumnado.filter_curso_academico_alumnado()`

6.168. filter_curso_academico_alumnado.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file filter_curso_academico_alumnado.py
00003 # @brief Este archivo contiene componente para filtrar cursos académicos del alumnado.
00004 # @version 1.0
00005 # @date 07/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado import
    update_filter_curso_academico_alumnado
00013
00014
00015 def filter_curso_academico_alumnado():
00016     """
00017 Crea un componente con un dropdown que contiene los cursos académicos.
00018
00019 Returns:
00020 html.Div: Componente con un dropdown y un botón para seleccionar todos los cursos académicos
00021 """
00022 return html.Div(
00023     [
00024         html.Br(),
00025         html.Label("Curso académico"),
00026         dcc.Dropdown(
00027             id="curso-academico",
00028             searchable=False,
00029             multi=True,
00030             clearable=True,
00031             options=[],
00032             value=None,
00033             maxHeight=300,
00034             placeholder="Seleccione una opción",
00035         ),
00036         html.Button(
00037             "Seleccionar todo",
00038             id="select-all-cursos-academicos",
00039             className="button-select-all-filter",
00040             n_clicks=0,
00041         ),
00042     ],
00043 )
```

6.169. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/alumnado/filters/filter_←
titulacion_alumnado.py

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.filters
- namespace components.alumnado.filters.filter_titulacion_alumnado

Funciones

- def components.alumnado.filters.filter_titulacion_alumnado ()

6.170. filter_titulacion_alumnado.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file filter_titulacion_alumnado.py
00003 # @brief Este archivo contiene componente para filtrar titulaciones del alumnado.
00004 # @version 1.0
00005 # @date 18/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.alumnado.filters.callback_filter_titulacion_alumnado import
    update_filter_titulacion_alumnado
00013
00014
00015 def filter_titulacion_alumnado():
00016     """
00017     Crea un componente con un dropdown que contiene las titulaciones del perfil "Alumno".
00018
00019     Returns:
00020         html.Div: Componente con un dropdown y almacenamiento de la titulación seleccionada
00021     """
00022     return html.Div(
00023         [
00024             html.Label("Titulación"),
00025             dcc.Dropdown(
00026                 id="titulacion-alumnado",
00027                 options=[],
00028                 value=None,
00029                 searchable=False,
00030                 clearable=False,
00031                 optionHeight=50,
00032                 maxHeight=300,
00033                 placeholder="Seleccione una opción",
00034             ),
00035             dcc.Store(id="selected-titulacion-alumnado-store", storage_type="local"),
00036         ]
00037     )
```

6.171. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
 Universities_TFG/src/components/alumnado/graphs/graphs_←
 general_alumnado.py

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.graphs
- namespace components.alumnado.graphs.graphs_general_alumnado

Funciones

- def `components.alumnado.graphs.graphs_general_alumnado.graphs_general_alumnado()`

6.172. graphs_general_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file graphs_general_alumnado.py
00003 # @brief Este archivo contiene el componente para los gráficos del perfil "Alumnado".
00004 # @version 1.0
00005 # @date 07/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 from callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa import
  update_graph_alumnado
00013 from callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_notas import update_graph_alumnado
00014 from callbacks.alumnado.graphs.general.callback_graph_asig_superadas_media import
  update_graph_alumnado
00015 from components.common.create_graph import create_graph
00016
00017
00018 def graphs_general_alumnado():
00019     """
00020 Retorna los gráficos de la pestaña "Rendimiento Académico general" del perfil "Alumnado"
00021
00022     Returns:
00023         html.Div: Gráficos
00024     """
00025 graph_ids = [
00026 "asignaturas-superadas-general-mi-nota",
00027     "nota-cualitativa-general-mi-nota",
00028     "nota-media-general-mi-nota",
00029 ]
00030
00031 config = {"displayModeBar": False}
00032 item_class = "graph-item-general-alumnado"
00033
00034 return html.Div(
00035     [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
00036     className="graphs-container-general-alumnado",
00037 )

```

6.173. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/alumnado/graphs/graphs_←
personal_alumnado.py

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.graphs
- namespace components.alumnado.graphs.graphs_personal_alumnado

Funciones

- def `components.alumnado.graphs.graphs_personal_alumnado.graphs_personal_alumnado()`

6.174. graphs_personal_alumnado.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file graphs_personal_alumnado.py
00003 # @brief Este archivo contiene el componente para los gráficos del perfil "Alumnado".
00004 # @version 1.0
00005 # @date 07/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 from callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado import
    update_graph_alumnado
00013 from callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado import
    update_graph_alumnado
00014 from callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_alumnado import
    update_graph_alumnado
00015 from callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado import
    update_graph_alumnado
00016 from components.common.create_graph import create_graph
00017
00018
00019 def graphs_personal_alumnado():
00020     """
00021 Retorna los gráficos de la pestaña "Expediente académico Personal" del perfil "Alumnado"
00022
00023     Returns:
00024         html.Div: Gráficos
00025     """
00026 graph_ids = [
00027     "graph-evolucion-progreso-academico",
00028     "graph-bar-evolucion-asignaturas-matriculadas",
00029     "graph-bar-calificaciones-por-asignatura",
00030     "graph-bar-tasa-rendimiento",
00031 ]
00032
00033 config = {"displayModeBar": False}
00034 item_class = "graph-item-personal-alumnado"
00035
00036     return html.Div(
00037         [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
00038         className="graphs-container-personal-alumnado",
00039     )
```

6.175. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/recomendador_alumnado.py

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.utils
- namespace components.alumnado.utils.recomendador_alumnado

Funciones

- def components.alumnado.utils.recomendador_alumnado()

6.176. recomendador_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file recomendador_alumnado.py
00003 # @brief Este archivo contiene el componente para la pestaña "Recomendador" del perfil "Alumnado".
00004 # @version 1.0
00005 # @date 19/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012
00013
00014 def recomendador_alumnado():
00015     """
00016 Esta función retorna el contenido de la pestaña "Recomendador" del perfil "Alumnado".
00017
00018     Returns:
00019         html.Div: Layout de la pestaña "Recomendador" del perfil "Alumnado"
00020     """
00021
00022     return html.Div([
00023         html.H1("Deserción universitaria: ¿Cuáles son las razones y cómo prevenirla?", className='titulo-recomendador-alumnado'),
00024         html.Img(src='assets/images/abandono_academico.jpg', className='imagen-recomendador-alumnado'),
00025         html.P("La deserción universitaria es un problema que puede afectar a muchas instituciones educativas, pero que puede evitarse con las estrategias adecuadas. Reducir las tasas de deserción universitaria es uno de los grandes retos de las Instituciones de Educación Superior (IES).", className='p-recomendador-alumnado'),
00026         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-alumnado'),
00027         html.P("La deserción universitaria es el abandono de los estudios universitarios por parte de un estudiante, sin haber finalizado la carrera. Esto puede tener graves consecuencias para tu futuro y tu bienestar.", className='p-recomendador-alumnado'),
00028         html.Br(),
00029         html.Br(),
00030         html.H2("¿Cuáles son las razones de la deserción universitaria?", className='sb-recomendador-alumnado'),
00031         html.P("Existen muchas razones por las que un estudiante puede abandonar sus estudios universitarios. Algunas de las razones más comunes son:", className='p-recomendador-alumnado'),
00032         html.Ul([
00033             html.Li(html.Span([html.Strong("Problemas económicos:"), " La falta de recursos económicos puede dificultar tu acceso a la educación superior."], className='li-recomendador-alumnado')),
00034             html.Li(html.Span([html.Strong("Problemas académicos:"), " Las dificultades para aprobar asignaturas pueden hacer que te sientas frustrado."], className='li-recomendador-alumnado')),
00035             html.Li(html.Span([html.Strong("Problemas personales:"), " Situaciones como problemas de salud, familiares o relaciones personales pueden afectarte."], className='li-recomendador-alumnado')),
00036             html.Li(html.Span([html.Strong("Falta de motivación:"), " La falta de interés en los estudios puede llevarte a abandonar la universidad."], className='li-recomendador-alumnado')),
00037             html.Li(html.Span([html.Strong("Falta de orientación:"), " No recibir la orientación adecuada puede dificultar tu adaptación a la vida universitaria."], className='li-recomendador-alumnado')),

```

```
00038      ], className='ul-recomendador-alumnado'),
00039      html.H2("¿Cómo prevenir la deserción universitaria?", className='sb-recomendador-alumnado'),
00040      html.P("Para prevenir la deserción, puedes buscar ayuda en los siguientes recursos que la
universidad ofrece:", className='p-recomendador-alumnado'),
00041      html.Ul([
00042          html.Li("Solicitar becas y ayudas económicas si tienes dificultades financieras.",
className='li-recomendador-alumnado'),
00043          html.Li("Participar en programas de tutoría y apoyo académico si tienes problemas con tus
asignaturas.", className='li-recomendador-alumnado'),
00044          html.Li("Aprovechar los programas de orientación y asesoramiento para adaptarte mejor a la
vida universitaria.", className='li-recomendador-alumnado'),
00045          html.Li("Asistir a actividades y talleres que mejoren tu motivación y habilidades de
estudio.", className='li-recomendador-alumnado'),
00046      ], className='ul-recomendador-alumnado'),
00047  ])
```

6.177. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↔ Universidad/6_curso/BI_Universities_↔ TFG/src/components/alumnado/utils/resumen_alumnado.py

Namespaces

- namespace `components`
- namespace `components.alumnado`
- namespace `components.alumnado.utils`
- namespace `components.alumnado.utils.resumen_alumnado`

Funciones

- def `components.alumnado.utils.resumen_alumnado.resumen_alumnado()`

6.178. resumen_alumnado.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file resumen_alumnado.py
00003 # @brief Este archivo contiene el componente para el resumen del perfil "Alumnado".
00004 # @version 1.0
00005 # @date 28/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 from callbacks.alumnado.utils.callback_resumen_alumnado import update_resumen_alumnado
00013
00014
00015 def resumen_alumnado():
00016     """
00017 Crea el layout del resumen del alumnado.
00018
00019 Returns:
00020     html.Div: Layout del resumen del alumnado
00021     """
00022     return html.Div([], id="resumen-alumnado")
```

6.179. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/select_alumnado.py

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.utils
- namespace components.alumnado.utils.select_alumnado

Funciones

- def components.alumnado.utils.select_alumnado.select_alumnado ()

6.180. select_alumnado.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file select_alumnado.py
00003 # @brief Este archivo contiene el componente para seleccionar un alumno.
00004 # @version 1.0
00005 # @date 05/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.alumnado.utils.callback_select_alumnado import store_selected_alumnado
00013
00014
00015 def select_alumnado():
00016     """
00017 Crea el desplegable para seleccionar un alumno.
00018
00019 Returns:
00020 html.Div: Desplegable para seleccionar un alumno
00021 """
00022 return html.Div(
00023     [
00024         html.Div(
00025             [
00026                 dcc.Dropdown(
00027                     options=[],
00028                     value=None,
00029                     id="alumnado-dropdown",
00030                     clearable=False,
00031                     placeholder="Seleccione un alumno",
00032                 )
00033             ],
00034             className="select-alumnado",
00035         ),
00036     ]
00037 )
```

6.181. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/tabs_alumnado.py

Namespaces

- namespace components
- namespace components.alumnado
- namespace components.alumnado.utils
- namespace components.alumnado.utils.tabs_alumnado

Funciones

- def `components.alumnado.utils.tabs_alumnado.tabs_alumnado ()`

6.182. tabs_alumnado.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file tabs_alumnado.py
00003 # @brief Este archivo contiene el componente para las pestañas del perfil "Alumnado".
00004 # @version 1.0
00005 # @date 28/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.alumnado.utils.callback_tabs_alumnado import render_content
00013
00014
00015 def tabs_alumnado():
00016     """
00017 Contiene las pestañas de la vista del alumnado.
00018
00019 Returns:
00020 html.Div: Pestañas del alumnado
00021 """
00022 return html.Div(
00023     [
00024         dcc.Tabs(
00025             id="tabs-alumnado",
00026             value="expediente-personal-tab",
00027             children=[
00028                 dcc.Tab(
00029                     label="Expediente académico personal",
00030                     value="expediente-personal-tab",
00031                 ),
00032                 dcc.Tab(
00033                     label="Rendimiento académico general",
00034                     value="rendimiento-academico-tab",
00035                 ),
00036                 dcc.Tab(label="Recomendaciones", value="recomendador-tab"),
00037             ],
00038             className="tabs",
00039         ),
00040         html.Div(id="tabs-alumnado-content"),
00041         dcc.Store(id="selected-alumnado-store", storage_type="local"),
00042         dcc.Location(id="url", refresh=False),
00043     ]
00044 )

```

6.183. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/common/create_graph.py**

Namespaces

- namespace `components`
- namespace `components.common`
- namespace `components.common.create_graph`

Funciones

- def `components.common.create_graph.create_graph (graph_id, item_class, config)`

6.184. create_graph.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file create_graph.py
00003 # @brief Este archivo contiene la función para crear un gráfico con un id, una clase y una
00004 # configuración específica.
00005 # @version 1.0
00006 # @date 14/06/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011 from dash import html, dcc
00012
00013
00014 def create_graph(graph_id, item_class, config):
00015     """
00016 Crea un gráfico con un id, una clase y una configuración específica.
00017
00018 Args:
00019 graph_id (str): ID del gráfico
00020 item_class (str): Clase del gráfico
00021 config (dict): Configuración del gráfico
00022
00023 Returns:
00024 html.Div: Gráfico
00025 """
00026 return html.Div(
00027     [dcc.Loading(children=[dcc.Graph(id=graph_id, figure={}, config=config)])],
00028     className=item_class,
00029 )

```

6.185. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/create_graph_with_table.py

Namespaces

- namespace [components](#)
- namespace [components.common](#)
- namespace [components.common.create_graph_with_table](#)

Funciones

- def [components.common.create_graph_with_table.create_graph_with_table](#) (graph_id, item_class, config, modal_config)

6.186. create_graph_with_table.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file create_graph_with_table.py
00003 # @brief Este archivo contiene la función para crear un gráfico con un modal para mostrar los datos y
00004 # descargarlos.
00005 # @version 1.0
00006 # @date 14/06/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010

```

```
00011 from components.common.create_graph import create_graph
00012 from components.common.modal_data import create_modal
00013 from dash import html
00014
00015
00016 def create_graph_with_table(graph_id, item_class, config, modal_config):
00017     """
00018     Crea un gráfico con un modal para mostrar los datos y descargarlos.
00019
00020     Args:
00021         graph_id (str): ID del gráfico
00022         item_class (str): Clase del gráfico
00023         config (dict): Configuración del gráfico
00024         modal_config (dict): Configuración del modal
00025
00026     Returns:
00027         html.Div: Gráfico con modal para mostrar los datos
00028     """
00029     return html.Div(
00030         [create_graph(graph_id, item_class, config), create_modal(**modal_config)],
00031         className=item_class,
00032     )
```

6.187. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_↔
Universities_TFG/src/components/common/filters.py

Namespaces

- namespace `components`
- namespace `components.common`
- namespace `components.common.filters`

Funciones

- def `components.common.filters.filters (filters)`

6.188. filters.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file filters.py
00003 # @brief Este archivo contiene componente que contiene los filtros de la aplicación.
00004 # @version 1.0
00005 # @date 28/04/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012
00013
00014 def filters(filters):
00015     """
00016     Crea un componente que contiene los filtros de la aplicación.
00017
00018     Args:
00019         filters (list): Lista de filtros
00020
00021     Returns:
00022         html.Div: Componente que contiene los filtros
00023     """
00024     return html.Div([html.H2("Filtros"), html.Div(filters)], className="filters")
```

6.189. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/common/footer.py

Namespaces

- namespace components
- namespace components.common
- namespace components.common.footer

Funciones

- def components.common.footer.footer ()

6.190. footer.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file footer.py
00003 # @brief Este archivo contiene el footer de la aplicación.
00004 # @version 1.0
00005 # @date 25/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012
00013
00014 def footer():
00015     """
00016 Retorna el footer de la aplicación
00017
00018 Returns:
00019 html.Div: Footer de la aplicación
00020 """
00021 return html.Div([
00022     html.P('2023-2024 Universidad de la Laguna - Visualización de datos académicos'),
00023     html.P('Pabellón de Gobierno, C/ Padre Herrera s/n Apartado Postal 456 38200, San Cristóbal de
La Laguna'),
00024     html.P('Santa Cruz de Tenerife - España')], className='footer')
```

6.191. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/common/header.py

Namespaces

- namespace components
- namespace components.common
- namespace components.common.header

Funciones

- def components.common.header.header (store_role)

6.192. header.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file header.py
00003 # @brief Este archivo contiene el componente del header de la aplicación.
00004 # @version 1.0
00005 # @date 25/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.common.callback_user_role import initialize_dropdown, update_role
00013
00014
00015 def header(store_role):
00016     """
00017     Contiene el header de la aplicación y el desplegable para seleccionar el rol.
00018
00019     Args:
00020         store_role: str: ID del store que contiene el rol del usuario
00021
00022     Returns:
00023         html.Div: Header de la aplicación
00024     """
00025     return html.Div(
00026         [
00027             html.Img(src="assets/images/logoULL.png", className="logo"),
00028             html.H1("Visualización de datos académicos", className="title"),
00029             dcc.Dropdown(
00030                 options=[
00031                     {"label": "Alumno", "value": "Alumno"},
00032                     {"label": "Docente", "value": "Docente"},
00033                     {"label": "Gestor", "value": "Gestor"},
00034                 ],
00035                 id="dropdown_role",
00036                 className="dropdown_role",
00037                 clearable=False,
00038                 searchable=False,
00039                 placeholder="Selecciona un rol",
00040                 value="Alumno",
00041             ),
00042             dcc.Store(id=store_role, storage_type="session"),
00043         ],
00044         className="header",
00045     )

```

6.193. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/common/modal_data.py

Namespaces

- namespace [components](#)
- namespace [components.common](#)
- namespace [components.common.modal_data](#)

Funciones

- def [components.common.modal_data.create_modal](#) (modal_id, table_container_id, download_button_id, view_data_button_id)

6.194. modal_data.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file modal_data.py
00003 # @brief Este archivo contiene el componente para mostrar los datos de una tabla en un modal.
00004 # @version 1.0
00005 # @date 29/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 import dash_bootstrap_components as dbc
00013
00014
00015 def create_modal(modal_id, table_container_id, download_button_id, view_data_button_id):
00016     """
00017     Crea el modal para mostrar los datos de una tabla. El modal tiene un botón para descargar los datos en
00018     formato CSV.
00019
00020     Args:
00021         modal_id (str): ID del modal
00022         table_container_id (str): ID del contenedor de la tabla
00023         download_button_id (str): ID del botón para descargar los datos
00024         view_data_button_id (str): ID del botón para abrir el modal
00025
00026     Returns:
00027         dbc.Modal: Modal para mostrar los datos
00028     """
00029     return html.Div([
00030         dbc.Button('Ver datos', id=view_data_button_id, n_clicks=0, color='primary'),
00031         dbc.Modal([
00032             dbc.ModalHeader("Datos"),
00033             dbc.ModalBody(html.Div(id=table_container_id), style={'maxHeight': 'calc(100vh - 200px)' ,
00034             'overflowY': 'auto'}),
00035             dbc.ModalFooter(
00036                 dbc.Button("Descargar CSV", id=download_button_id, color='primary')
00037             )
00038         ], id=modal_id, is_open=False, size="lg")
00039     ])

```

6.195. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/common/sidebar.py

Namespaces

- namespace [components](#)
- namespace [components.common](#)
- namespace [components.common.sidebar](#)

Funciones

- def [components.common.sidebar.sidebar](#) (components)

6.196. sidebar.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file sidebar.py
00003 # @brief Este archivo contiene el componente del sidebar del dashboard.
00004 # @version 1.0

```

```
00005 # @date 27/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 import dash_bootstrap_components as dbc
00013 from callbacks.common.callback_sidebarCollapse import sidebarCollapse
00014
00015
00016 def sidebar(components):
00017     """
00018 Sidebar del dashboard donde se encuentra el resumen de la información y los filtros
00019
00020 Args:
00021 components (list): Componentes que se desplegarán en el sidebar
00022
00023 Returns:
00024 html.Div: Sidebar del dashboard
00025 """
00026 sidebar_content = html.Div(components, className="sidebar")
00027 sidebarCollapse = dbc.Collapse(sidebar_content, id="collapse", is_open=True)
00028
00029 return html.Div(sidebarCollapse)
```

6.197. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_↔
Universities_TFG/src/components/docente/filters/filter_all_↔
asignaturas_titulacion_docente.py

Namespaces

- namespace `components`
- namespace `components.docente`
- namespace `components.docente.filters`
- namespace `components.docente.filters.filter_all_asignaturas_titulacion_docente`

Funciones

- def `components.docente.filters.filter_all_asignaturas_titulacion_docente.filter_all_asignaturas_titulacion_docente()`

6.198. filter_all_asignaturas_titulacion_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file filter_all_asignaturas_titulacion_docente
00003 # @brief Este fichero contiene el componente que contiene un filtro multiopción con todas las
00004 # asignaturas
00005 # @version 1.0
00006 # @date 20/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html, dcc
00013 from callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente import
00014     update_filter_asignaturas_docente
00015
00016 def filter_all_asignaturas_titulacion_docente():
```

```

00017 """
00018 Crea un componente con un dropdown que contiene todas las asignaturas de la titulación del perfil
00019 "Docente"
00020 en la pestaña "Rendimiento académico general".
00021
00022 Returns:
00023     html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
00024 """
00025
00026     return html.Div(
00027         [
00028             html.Br(),
00029             html.Label("Asignaturas"),
00030             dcc.Dropdown(
00031                 id="all-asignaturas-titulacion-docente",
00032                 searchable=True,
00033                 value=None,
00034                 clearable=True,
00035                 options=[],
00036                 maxHeight=300,
00037                 optionHeight=50,
00038                 multi=True,
00039             ),
00040             html.Button(
00041                 "Seleccionar todo",
00042                 id="select-all-asignaturas-titulacion-docente",
00043                 className="button-select-all-filter",
00044                 n_clicks=0,
00045             ),
00046         ],
00047     )

```

6.199. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_ Universities_TFG/src/components/docente/filters/filter_all_curso_academico.py

Namespaces

- namespace `components`
- namespace `components.docente`
- namespace `components.docente.filters`
- namespace `components.docente.filters.filter_all_curso_academico`

Funciones

- def `components.docente.filters.filter_all_curso_academico.filter_all_curso_academico()`

6.200. filter_all_curso_academico.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file filter_all_curso_academico.py
00003 # @brief Este fichero contiene el componente de filtro que contiene un dropdown con todos los cursos
00004 # académicos
00005 # @version 1.0
00006 # @date 20/05/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011 from dash import html, dcc
00012 from callbacks.docente.filters.callback_filter_all_curso_academico import
00013     update_filter_all_cursos_academicos_docente

```

```
00013
00014
00015 def filter_all_curso_academico():
00016     """
00017     Crea un componente con un dropdown que contiene todos los cursos académicos del perfil "Docente"
00018     en la pestaña "Rendimiento académico general".
00019
00020     Returns:
00021         .Div: Componente con un dropdown
00022     """
00023     return html.Div(
00024         [
00025             html.Br(),
00026             html.Label("Curso académico"),
00027             dcc.Dropdown(
00028                 id="all-cursos-academicos-docente",
00029                 searchable=False,
00030                 multi=False,
00031                 clearable=False,
00032                 options=[],
00033                 value=None,
00034                 maxHeight=300,
00035             ),
00036         ]
00037     )
```

6.201. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/docente/filters/filter_←
asignaturas_docente.py

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.filters
- namespace components.docente.filters.filter_asignaturas_docente

Funciones

- def components.docente.filters.filter_asignaturas_docente.filter_asignaturas_docente()

6.202. filter_asignaturas_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file filter_asignaturas_docente.py
00003 # @brief Este fichero contiene el componente de un filtro con un dropdown que contiene las asignaturas
00004 # del perfil "Docente"
00005 # @version 1.0
00006 # @date 15/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010
00011 from dash import html, dcc
00012 from callbacks.docente.filters.callback_filter_asignaturas_docente import
00013     update_filter_asignaturas_docente
00014
00015 def filter_asignaturas_docente():
00016     """
00017     Crea un componente con un dropdown que contiene las asignaturas del perfil "Docente"
```

```

00018     en la pestaña "Rendimiento académico personal".
00019
00020     Returns:
00021         html.Div: Componente con un dropdown
00022     """
00023
00024     return html.Div([
00025         html.Br(),
00026         html.Label("Asignaturas"),
00027         dcc.Dropdown(
00028             id="asignaturas-docente",
00029             searchable=True,
00030             value=None,
00031             clearable=True,
00032             options=[],
00033             maxHeight=300,
00034             optionHeight=50,
00035         )
00036     ])

```

6.203. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/docente/filters/filter_curso_←
academico_docente.py**

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.filters
- namespace components.docente.filters.filter_curso_academico_docente

Funciones

- def components.docente.filters.filter_curso_academico_docente.filter_curso_academico_docente ()

6.204. filter_curso_academico_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file filter_curso_academico_docente.py
00003 # @brief Este fichero contiene el componente de filtro que contiene los cursos académicos del
00004 # perfil "Docente"
00005 # @version 1.0
00006 # @date 20/05/2024
00007 # @author Fabrizzio Daniell Perilli Martin
00008 # @email alu0101138589@ull.edu.es
00009 #
00010 from dash import html, dcc
00011 from callbacks.docente.filters.callback_filter_curso_academico_docente import
00012     update_filter_curso_academico_docente
00013
00014 def filter_curso_academico_docente():
00015     """
00016 Crea un componente con un dropdown que contiene los cursos académicos del perfil "Docente"
00017     en la pestaña "Rendimiento académico personal".
00018
00019     Returns:
00020         html.Div: Componente con un dropdown
00021     """
00022     return html.Div(
00023         [

```

```
00024         html.Br(),
00025         html.Label("Curso académico"),
00026         dcc.Dropdown(
00027             id="curso-academico-docente",
00028             searchable=False,
00029             multi=True,
00030             clearable=True,
00031             options=[],
00032             value=None,
00033             maxHeight=300,
00034         ),
00035         html.Button(
00036             "Seleccionar todo",
00037             id="select-all-cursos-academicos-docente",
00038             className="button-select-all-filter",
00039             n_clicks=0,
00040         ),
00041     ],
00042 )
```

6.205. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/← Universidad/6_curso/BI_Universities_← TFG/src/components/docente/filters/filter_titulacion_docente.py

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.filters
- namespace components.docente.filters.filter_titulacion_docente

Funciones

- def components.docente.filters.filter_titulacion_docente.filter_titulacion_docente ()

6.206. filter_titulacion_docente.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file filter_titulacion_docente.py
00003 # @brief Este fichero contiene el componente que contiene un filtro con las titulaciones del perfil
00004 # "Docente"
00005 # @version 1.0
00006 # @date 19/05/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010
00011 from dash import html, dcc
00012 from callbacks.docente.filters.callback_filter_titulacion_docente import
00013     update_filter_titulacion_docente
00014
00015 def filter_titulacion_docente():
00016     """
00017     Crea un componente con un dropdown que contiene las titulaciones del perfil "Docente"
00018     en la pestaña "Rendimiento académico personal" y "Rendimiento académico general".
00019
00020     Returns:
00021         html.Div: Componente con un dropdown
00022     """
00023
00024     return html.Div(
00025         [
00026             html.Label("Titulación"),
```

```

00027         dcc.Dropdown(
00028             id="titulacion-docente",
00029             options=[],
00030             value=None,
00031             searchable=False,
00032             clearable=False,
00033             optionHeight=50,
00034             maxHeight=300,
00035         ),
00036         dcc.Store(id="selected-titulacion-docente-store", storage_type="local"),
00037     ]
00038 )

```

6.207. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/graphs/graphs_general_docente.py

Namespaces

- namespace [components](#)
- namespace [components.docente](#)
- namespace [components.docente.graphs](#)
- namespace [components.docente.graphs.graphs_general_docente](#)

Funciones

- def [components.docente.graphs.graphs_general_docente.graphs_general_docente\(\)](#)

6.208. graphs_general_docente.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file graphs_general_docente.py
00003 # @brief Este fichero contiene el componente que muestra los gráficos del perfil
00004 # "Docente" en la pestaña "Rendimiento académico general"
00005 # @version 1.0
00006 # @date 16/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html
00013 from callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente import
    update_graph_docente
00014 from callbacks.docente.graphs.general.callback_graph_all_calif_media_docente import
    update_graph_docente
00015 from components.common.create_graph import create_graph
00016 from util import config_mode_bar_buttons_gestor
00017
00018
00019 def graphs_general_docente():
00020     """
00021 Retorna los gráficos de la pestaña "Rendimiento académico general"
00022     del perfil "Docente"
00023
00024     Returns:
00025         html.Div: Gráficos
00026     """
00027
00028 graphs_ids = [
00029     "calificaciones-cuali-all-asig-docente",
00030     "calificaciones-media-all-asig-docente",
00031 ]
00032
00033     item_class = "graph-item-general-docente"
00034     config = config_mode_bar_buttons_gestor
00035
00036     return html.Div(
00037         [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
00038         className="graphs-container-general-docente",
00039     )

```

6.209. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_↔
Universities_TFG/src/components/docente/graphs/graphs_↔
personal_docente.py

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.graphs
- namespace components.docente.graphs.graphs_personal_docente

Funciones

- def `components.docente.graphs.graphs_personal_docente.graphs_personal_docente()`

6.210. graphs_personal_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file graphs_personal_docente.py
00003 # @brief Este fichero contiene el componente que muestra los gráficos del perfil
00004 #         "Docente" en la pestaña "Rendimiento académico personal"
00005 # @version 1.0
00006 # @date 16/05/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html
00013 from callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente import
    update_graph_docente
00014 from callbacks.docente.graphs.personal.callback_graph_alu_genero_docente import update_graph_docente
00015 from callbacks.docente.graphs.personal.callback_graph_alu_media_docente import update_graph_docente
00016 from callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente import
    update_graph_docente
00017 from components.common.create_graph import create_graph
00018 from util import config_mode_bar_buttons_gestor
00019
00020
00021 def graphs_personal_docente():
00022     """
00023 Retorna los gráficos de la pestaña "Rendimiento académico personal"
00024     del perfil "Docente"
00025
00026     Returns:
00027         html.Div: Gráficos
00028     """
00029 graphs_ids = [
00030     "graph-alumnos-repetidores-nuevos",
00031     "graph-alumnos-matri-genero",
00032     "graph-alumnos-nota-media",
00033     "graph-alumnos-nota-cualitativa",
00034 ]
00035
00036     item_class = "graph-item-personal-docente"
00037     config = config_mode_bar_buttons_gestor
00038
00039     return html.Div(
00040         [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
00041         className="graphs-container-personal-docente",
00042     )
00043
```

6.211. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/recomendador_docente.py

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.utils
- namespace components.docente.utils.recomendador_docente

Funciones

- def components.docente.utils.recomendador_docente.recomendador_docente ()

6.212. recomendador_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file recomendador_docente.py
00003 # @brief Este fichero contiene el componente del recomendador para docentes.
00004 # @version 1.0
00005 # @date 23/05/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012
00013 def recomendador_docente():
00014     """
00015 Retorna el componente del recomendador para docentes.
00016
00017 Returns:
00018 html.Div: Componente del recomendador para docentes
00019 """
00020
00021 return html.Div([
00022     html.H1("Deserción universitaria: ¿Cómo pueden ayudar los docentes a prevenirla?", className='titulo-recomendador-docente'),
00023     html.Img(src='assets/images/docente_clase.jpg', className='imagen-recomendador-docente'),
00024     html.P("La deserción universitaria es un problema serio que afecta a muchas instituciones educativas. Los docentes juegan un papel crucial en la identificación y prevención de este fenómeno.", className='p-recomendador-docente'),
00025     html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-docente'),
00026     html.P("La deserción universitaria es el abandono de los estudios universitarios por parte de un estudiante, sin haber finalizado la carrera. Este problema puede tener graves consecuencias tanto para los estudiantes como para la institución.", className='p-recomendador-docente'),
00027     html.Br(),
00028     html.H2("¿Cómo pueden los docentes ayudar a prevenir la deserción universitaria?", className='sb-recomendador-docente'),
00029     html.Ul([
00030         html.Li("Identificar y apoyar a los estudiantes con dificultades académicas proporcionando tutorías y asesorías.", className='li-recomendador-docente'),
00031         html.Li("Fomentar un ambiente de clase inclusivo y motivador que promueva la participación y el interés por el aprendizaje.", className='li-recomendador-docente'),
00032         html.Li("Mantener una comunicación abierta y constante con los estudiantes para entender sus problemas y necesidades.", className='li-recomendador-docente'),
00033         html.Li("Colaborar con los servicios de apoyo estudiantil para proporcionar la ayuda necesaria en caso de problemas personales o económicos.", className='li-recomendador-docente'),
00034         html.Li("Participar en programas de formación continua para mejorar las estrategias pedagógicas y adaptarse a las necesidades cambiantes de los estudiantes.", className='li-recomendador-docente'),
00035     ], className='ul-recomendador-docente'),
00036 ])
```

6.213. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_← TFG/src/components/docente/utils/resumen_docente.py

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.utils
- namespace components.docente.utils.resumen_docente

Funciones

- def components.docente.utils.resumen_docente.resumen_docente ()

6.214. resumen_docente.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file recomendador_docente.py
00003 # @brief Este fichero contiene el componente del recomendador para docentes.
00004 # @version 1.0
00005 # @date 14/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 from callbacks.docente.utils.callback_resumen_docente import update_resumen_docente
00013
00014
00015 def resumen_docente():
00016     """
00017     Crea el layout del resumen del docente
00018
00019 Returns:
00020     html.Div: Layout del resumen del docente
00021     """
00022     return html.Div([], id="resumen-docente")
```

6.215. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_← TFG/src/components/docente/utils/select_docente.py

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.utils
- namespace components.docente.utils.select_docente

Funciones

- def components.docente.utils.select_docente.select_docente ()

6.216. select_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file select_docente.py
00003 # @brief Este fichero contiene el componente para seleccionar un docente.
00004 # @version 1.0
00005 # @date 14/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.docente.utils.callback_select_docente import store_selected_docente
00013
00014
00015 def select_docente():
00016     """
00017 Crea el desplegable para seleccionar un docente.
00018
00019 Returns:
00020 html.Div: Desplegable para seleccionar un docente
00021 """
00022 return html.Div(
00023     [
00024         html.Div(
00025             [
00026                 dcc.Dropdown(
00027                     options=[],
00028                     value=None,
00029                     id="docente-dropdown",
00030                     clearable=False,
00031                     placeholder="Selecciona un docente",
00032                 )
00033             ],
00034             className="select-docente",
00035         ),
00036     ],
00037 )
```

6.217. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/docente/utils/tabs_docente.py

Namespaces

- namespace components
- namespace components.docente
- namespace components.docente.utils
- namespace components.docente.utils.tabs_docente

Funciones

- def components.docente.utils.tabs_docente.tabs_docente ()

6.218. tabs_docente.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file tabs_docente.py
00003 # @brief Este fichero contiene el componente de las pestañas del perfil "Docente"
00004 # @version 1.0
```

```
00005 # @date 28/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.docente.utils.callback_tabs_docente import render_content
00013
00014
00015 def tabs_docente():
00016     """
00017     Crea las pestañas del perfil "Docente"
00018
00019     Returns:
00020         html.Div: Pestañas del docente
00021     """
00022
00023     return html.Div(
00024         [
00025             dcc.Tabs(
00026                 id="tabs-docente",
00027                 value="rendimiento-academico-asignatura-tab",
00028                 children=[
00029                     dcc.Tab(
00030                         label="Rendimiento académico personal",
00031                         value="rendimiento-academico-asignatura-tab",
00032                     ),
00033                     dcc.Tab(
00034                         label="Rendimiento académico general",
00035                         value="rendimiento-academico-tab",
00036                     ),
00037                     dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
00038                 ],
00039                 className="tabs",
00040             ),
00041             html.Div(id="tabs-docente-content"),
00042             dcc.Store(id="selected-docente-store", storage_type="local"),
00043         ]
00044     )
```

6.219. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_all_curso_academico_gestor.py

Namespaces

- namespace components
- namespace components.gestor
- namespace components.gestor.filters
- namespace components.gestor.filters.filter_all_curso_academico_gestor

Funciones

- def components.gestor.filters.filter_all_curso_academico_gestor.filter_all_curso_academico_gestor ()

6.220. filter_all_curso_academico_gestor.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file filter_all_curso_academico_gestor.py
00003 # @brief Este fichero contiene el componente de filtro que contiene
00004 #         todos los cursos académicos para el perfil "Gestor"
```

```

00005 # @version 1.0
00006 # @date 27/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html, dcc
00013 from callbacks.gestor.filters.callback_filter_all_curso_academico_gestor import
    update_filter_all_curso_academico_gestor
00014
00015
00016 def filter_all_curso_academico_gestor():
00017     """
00018 Crea el filtro que muestra todos los cursos académicos disponibles
00019 para el perfil "Gestor" de la pestaña "Riesgo académico".
00020
00021     Returns:
00022         html.Div: Componente con el filtro de curso académico
00023     """
00024
00025     return html.Div(
00026         [
00027             html.Label("Curso académico"),
00028             dcc.Dropdown(
00029                 id="curso-all-academico-gestor",
00030                 searchable=True,
00031                 multi=True,
00032                 clearable=True,
00033                 options=[],
00034                 value=None,
00035                 maxHeight=200,
00036                 placeholder="Seleccione una opción",
00037             ),
00038             html.Button(
00039                 "Seleccionar todo",
00040                 id="select-all-curso-academico-button",
00041                 className="button-select-all-filter",
00042                 n_clicks=0,
00043             ),
00044             dcc.Store(id="curso-all-academico-gestor-store", storage_type="local"),
00045         ]
00046     )

```

6.221. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
**Universities_TFG/src/components/gestor/filters/filter_curso_←
 academico_gestor.py**

Namespaces

- namespace [components](#)
- namespace [components.gestor](#)
- namespace [components.gestor.filters](#)
- namespace [components.gestor.filters.filter_curso_academico_gestor](#)

Funciones

- def [components.gestor.filters.filter_curso_academico_gestor.filter_curso_academico_gestor \(\)](#)

6.222. filter_curso_academico_gestor.py

[Ir a la documentación de este archivo.](#)

```
00001 #
```

```
00002 # @file filter_curso_academico_gestor.py
00003 # @brief Este fichero contiene el componente de filtro que contiene
00004 #     el curso académico para el perfil "Gestor"
00005 # @version 1.0
00006 # @date 21/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html, dcc
00013 from callbacks.gestor.filters.callback_filter_curso_academico_gestor import
00014     update_filter_curso_academico_gestor
00015
00016 def filter_curso_academico_gestor():
00017     """
00018     Crea el filtro de curso académico para el perfil "Gestor" de la pestaña
00019     "Indicadores académicos".
00020
00021     Returns:
00022         html.Div: Componente con el filtro de curso académico
00023     """
00024     return html.Div(
00025         [
00026             html.Label("Curso académico"),
00027             dcc.Dropdown(
00028                 id="curso-academico-gestor",
00029                 searchable=True,
00030                 multi=False,
00031                 clearable=False,
00032                 options=[],
00033                 value=None,
00034                 maxHeight=300,
00035                 placeholder="Selecciona una opción",
00036             ),
00037         ],
00038     )
```

6.223. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/← Universidad/6_curso/BI_Universities_← TFG/src/components/gestor/filters/filter_titulaciones_gestor.py

Namespaces

- namespace [components](#)
- namespace [components.gestor](#)
- namespace [components.gestor.filters](#)
- namespace [components.gestor.filters.filter_titulaciones_gestor](#)

Funciones

- def [components.gestor.filters.filter_titulaciones_gestor.filter_titulaciones_gestor\(\)](#)

6.224. filter_titulaciones_gestor.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file filter_titulaciones_gestor.py
00003 # @brief Este fichero contiene el componente de filtro que contiene
00004 #     las titulaciones para el perfil "Gestor"
00005 # @version 1.0
00006 # @date 21/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
```

```

00010 #
00011
00012 from dash import html, dcc
00013 from callbacks.gestor.filters.callback_filter_titulaciones_gestor import
    update_filter_titulaciones_gestor
00014
00015
00016 def filter_titulaciones_gestor():
00017     """
00018     Crea el filtro de titulaciones para el perfil "Gestor" de la pestaña
00019     "Indicadores académicos".
00020
00021     Returns:
00022         html.Div: Componente con el filtro de titulaciones
00023     """
00024     return html.Div(
00025         [
00026             html.Br(),
00027             html.Label("Titulaciones"),
00028             dcc.Dropdown(
00029                 id="titulaciones-gestor",
00030                 searchable=True,
00031                 multi=True,
00032                 clearable=True,
00033                 options=[],
00034                 value=None,
00035                 maxHeight=200,
00036                 placeholder="Seleccione una opción",
00037             ),
00038             html.Button(
00039                 "Seleccionar todo",
00040                 id="select-all-titulaciones-gestor",
00041                 className="button-select-all-filter",
00042                 n_clicks=0,
00043             ),
00044         ],
00045     )

```

6.225. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/components/gestor/graphs/graphs_←
indicadores_gestor.py**

Namespaces

- namespace [components](#)
- namespace [components.gestor](#)
- namespace [components.gestor.graphs](#)
- namespace [components.gestor.graphs.graphs_indicadores_gestor](#)

Funciones

- def [components.gestor.graphs.graphs_indicadores_gestor.components.gestor.graphs.graphs_indicadores_gestor\(\)](#)

6.226. graphs_indicadores_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file graphs_indicadores_gestor.py
00003 # @brief Este fichero contiene el componente que muestra los gráficos
00004 #       de los indicadores del perfil "Gestor"
00005 # @version 1.0
00006 # @date 21/05/2024
00007 # @license MIT License

```

```
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html
00013 from callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor import
00014     update_graph_gestor
00015 from callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor import
00016     update_graph_gestor
00017 from callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor import
00018     update_graph_gestor
00019 from callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor import
00020     update_graph_gestor
00021 from components.common.create_graph_with_table import create_graph_with_table
00022 from util import config_mode_bar_buttons_gestor
00023
00024
00025
00026     Returns:
00027         html.Div: Gráficos
00028 """
00029
00030 graphs_info = [
00031 {
00032     "graph_id": "nuevo-ingreso-genero-gestor",
00033         "modal_id": "modal-nuevo-ingreso-genero",
00034         "table_container_id": "table-container-nuevo-ingreso-genero",
00035         "download_button_id": "btn-descargar-nuevo-ingreso-genero",
00036         "view_data_button_id": "btn-ver-datos-nuevo-ingreso-genero",
00037     },
00038     {
00039         "graph_id": "nuevo_ingreso_nacionalidad_gestor",
00040         "modal_id": "modal-nuevo-ingreso-nacionalidad",
00041         "table_container_id": "table-container-nuevo-ingreso-nacionalidad",
00042         "download_button_id": "btn-descargar-nuevo-ingreso-nacionalidad",
00043         "view_data_button_id": "btn-ver-datos-nuevo-ingreso-nacionalidad",
00044     },
00045     {
00046         "graph_id": "egresados-genero-gestor",
00047         "modal_id": "modal-egresados-genero",
00048         "table_container_id": "table-container-egresados-genero",
00049         "download_button_id": "btn-descargar-egresados-genero",
00050         "view_data_button_id": "btn-ver-datos-egresados-genero",
00051     },
00052     {
00053         "graph_id": "egresados-nacionalidad-gestor",
00054         "modal_id": "modal-egresados-nacionalidad",
00055         "table_container_id": "table-container-egresados-nacionalidad",
00056         "download_button_id": "btn-descargar-egresados-nacionalidad",
00057         "view_data_button_id": "btn-ver-datos-egresados-nacionalidad",
00058     },
00059 ]
00060
00061 graph_elements = [
00062     create_graph_with_table(
00063         graph_info["graph_id"],
00064         "graph-item-indicadores-gestor",
00065         config_mode_bar_buttons_gestor,
00066         {
00067             "modal_id": graph_info["modal_id"],
00068             "table_container_id": graph_info["table_container_id"],
00069             "download_button_id": graph_info["download_button_id"],
00070             "view_data_button_id": graph_info["view_data_button_id"],
00071         },
00072     )
00073     for graph_info in graphs_info
00074 ]
00075
00076 return html.Div(graph_elements, className="graphs-container-indicadores-gestor")
```

6.227. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_resultados_gestor.py

Namespaces

- namespace `components`
- namespace `components.gestor`
- namespace `components.gestor.graphs`
- namespace `components.gestor.graphs.graphs_resultados_gestor`

Funciones

- def `components.gestor.graphs.graphs_resultados_gestor()`

6.228. graphs_resultados_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file graphs_resultados_gestor.py
00003 # @brief Este fichero contiene el componente que muestra los gráficos
00004 # de los resultados académicos del perfil "Gestor"
00005 # @version 1.0
00006 # @date 22/05/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 from dash import html
00013 from callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor import
00014     update_grph_gestor
00015 from callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor import
00016     update_graph_gestor
00017 from components.common.create_graph_with_table import create_graph_with_table
00018 from util import config_mode_bar_buttons_gestor
00019 def graphs_resultados_gestor():
00020     """
00021 Retorna los gráficos de la pestaña "Resultados académicos" del perfil
00022 "Gestor". Incluye un modal y una tabla para cada gráfico.
00023
00024     Returns:
00025         html.Div: Gráficos
00026     """
00027 graphs_info = [
00028 {
00029     "graph_id": "duracion-estudios-nota-media-gestor",
00030         "modal_id": "modal-duracion-estudios",
00031         "table_container_id": "table-container-duracion-estudios",
00032         "download_button_id": "btn-descargar-csv-duracion-estudios",
00033         "view_data_button_id": "btn-ver-datos-duracion-estudios",
00034     },
00035     {
00036         "graph_id": "nota-acceso-titulacion",
00037         "modal_id": "modal-nota-acceso",
00038         "table_container_id": "table-container-nota-acceso",
00039         "download_button_id": "btn-descargar-csv-nota-acceso",
00040         "view_data_button_id": "btn-ver-datos-nota-acceso",
00041     },
00042 ]
00043
00044 graph_elements = [
00045     create_graph_with_table(
00046         graph_info["graph_id"],
00047         "graph-item-resultados-gestor",
00048         config_mode_bar_buttons_gestor,
00049     {

```

```
00050         "modal_id": graph_info["modal_id"],
00051         "table_container_id": graph_info["table_container_id"],
00052         "download_button_id": graph_info["download_button_id"],
00053         "view_data_button_id": graph_info["view_data_button_id"],
00054     },
00055     )
00056     for graph_info in graphs_info
00057 ]
00058
00059 return html.Div(graph_elements, className="graphs-container-resultados-gestor")
```

6.229. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_riesgo_abandono_gestor.py

Namespaces

- namespace [components](#)
- namespace [components.gestor](#)
- namespace [components.gestor.graphs](#)
- namespace [components.gestor.graphs.graphs_riesgo_abandono_gestor](#)

Funciones

- def [components.gestor.graphs.graphs_riesgo_abandono_gestor.graphs_riesgo_abandono_gestor\(\)](#)

6.230. graphs_riesgo_abandono_gestor.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file graphs_riesgo_abandono_gestor.py
00003 # @brief Este fichero contiene el componente que muestra los gráficos
00004 #       de los indicadores de riesgo de abandono del perfil "Gestor"
00005 # @version 1.0
00006 # @date 26/05/2024
00007 # @license MIT License
00008 # @author Fabrizzio Daniell Perilli Martín
00009 # @email alu010138589@ull.edu.es
00010 #
00011
00012 from dash import html
00013 from callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor import
00014     update_graph_gestor
00015 from callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor import
00016     update_graph_gestor
00017 from components.common.create_graph_with_table import create_graph_with_table
00018 from util import config_mode_bar_buttons_gestor
00019
00020 def graphs_riesgo_abandono_gestor():
00021     """
00022     Retorna los gráficos de la pestaña "Riesgo de abandono" del perfil
00023     "Gestor". Incluye un modal y una tabla para cada gráfico.
00024
00025     Returns:
00026         html.Div: Gráficos
00027     """
00028     graphs_info = [
00029         "graph_id": "tasa-abandono-gestor",
00030         "modal_id": "modal-tasa-abandono",
00031         "table_container_id": "table-container-tasa-abandono",
00032         "download_button_id": "btn-descargar-tasa-abandono",
```

```

00033         "view_data_button_id": "btn-ver-datos-tasa-abandono",
00034     },
00035     {
00036         "graph_id": "tasa-graduacion-gestor",
00037         "modal_id": "modal-tasa-graduacion",
00038         "table_container_id": "table-container-tasa-graduacion",
00039         "download_button_id": "btn-descargar-tasa-graduacion",
00040         "view_data_button_id": "btn-ver-datos-tasa-graduacion",
00041     },
00042 ]
00043
00044 graph_elements = [
00045     create_graph_with_table(
00046         graph_info["graph_id"],
00047         "graph-item-riesgo-abandono-gestor",
00048         config_mode_bar_buttons_gestor,
00049         {
00050             "modal_id": graph_info["modal_id"],
00051             "table_container_id": graph_info["table_container_id"],
00052             "download_button_id": graph_info["download_button_id"],
00053             "view_data_button_id": graph_info["view_data_button_id"],
00054         },
00055     )
00056     for graph_info in graphs_info
00057 ]
00058
00059 return html.Div(graph_elements, className="graphs-container-riesgo-abandono-gestor")

```

6.231. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/recomendador_gestor.py

Namespaces

- namespace `components`
- namespace `components.gestor`
- namespace `components.gestor.utils`
- namespace `components.gestor.utils.recomendador_gestor`

Funciones

- def `components.gestor.utils.recomendador_gestor.recomendador_gestor()`

6.232. recomendador_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file recomendador_gestor.py
00003 # @brief Este fichero contiene el componente con información y recomendaciones para el perfil "Gestor"
00004 # @version 1.0
00005 # @date 23/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012
00013
00014 def recomendador_gestor():
00015     """
00016     Crea un contenedor con información y recomendaciones sobre la deserción universitaria
00017     para los gestores de la sección "Recomendaciones" de la pestaña "Gestor".
00018
00019     Returns:
00020         html.Div: Contenedor con información y recomendaciones sobre la deserción universitaria

```

```
00021     """
00022     return html.Div([
00023         html.H1("Deserción universitaria", className='titulo-recomendador-gestor'),
00024         html.Img(src='assets/images/abandono-gestores.jpg', className='imagen-recomendador-gestor'),
00025         html.P("La deserción universitaria es un desafío significativo para las instituciones de
00026             educación superior. Los gestores tienen la responsabilidad de desarrollar e implementar políticas
00027             efectivas para reducirla.", className='p-recomendador-gestor'),
00028         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-gestor'),
00029         html.P("La deserción universitaria se refiere al abandono de los estudios por parte de los
00030             estudiantes antes de completar su formación. Este fenómeno tiene implicaciones negativas para la
00031             universidad y para la sociedad.", className='p-recomendador-gestor'),
00032         html.Br(),
00033         html.H2("Estrategias para prevenir la deserción universitaria",
00034             className='sb-recomendador-gestor'),
00035         html.Ul([
00036             html.Li("Desarrollar programas de becas y ayudas financieras para apoyar a los estudiantes
00037                 con dificultades económicas.", className='li-recomendador-gestor'),
00038             html.Li("Implementar sistemas de tutoría y mentoría para ofrecer apoyo académico y
00039                 emocional a los estudiantes.", className='li-recomendador-gestor'),
00040             html.Li("Establecer programas de orientación y adaptación para nuevos estudiantes,
00041                 facilitando su integración en la vida universitaria.", className='li-recomendador-gestor'),
00042             html.Li("Promover la formación continua del personal docente en metodologías pedagógicas
00043                 innovadoras.", className='li-recomendador-gestor'),
00044             html.Li("Utilizar sistemas de seguimiento y análisis de datos para identificar a
00045                 estudiantes en riesgo y actuar de manera preventiva.", className='li-recomendador-gestor'),
00046             html.Li("Fomentar un clima institucional inclusivo y de apoyo que motive a los estudiantes
00047                 a continuar con sus estudios.", className='li-recomendador-gestor'),
00048         ], className='ul-recomendador-gestor'),
00049     ])
00050 ])
```

6.233. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_↔
Universities_TFG/src/components/gestor/utils/resumen_gestor.py

Namespaces

- namespace components
- namespace components.gestor
- namespace components.gestor.utils
- namespace components.gestor.utils.resumen_gestor

Funciones

- def components.gestor.utils.resumen_gestor.resumen_gestor ()

6.234. resumen_gestor.py

Ir a la documentación de este archivo.

```
00001 #
00002 # @file resumen_gestor.py
00003 # @brief Este fichero contiene el componente para mostrar un resumen de los datos del gestor
00004 # @version 1.0
00005 # @date 21/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 from callbacks.gestor.utils.callback_resumen_gestor import update_resumen_gestor
00013
00014
00015 def resumen_gestor():
00016     """
00017     Crea un contenedor para mostrar un resumen de los datos del gestor seleccionado.
00018
00019     Returns:
00020     html.Div: Contenedor para mostrar el resumen
00021     """
00022
00023     return html.Div([], id="resumen-gestor")
```

6.235. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/gestor/utils/select_gestor.py

Namespaces

- namespace components
- namespace components.gestor
- namespace components.gestor.utils
- namespace components.gestor.utils.select_gestor

Funciones

- def components.gestor.utils.select_gestor.select_gestor ()

6.236. select_gestor.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file select_gestor.py
00003 # @brief Este fichero contiene el componente para seleccionar un gestor.
00004 # @version 1.0
00005 # @date 21/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.gestor.utils.callback_select_gestor import store_selected_gestor
00013
00014
00015 def select_gestor():
00016     """
00017 Crea un dropdown para seleccionar un gestor.
00018
00019 Returns:
00020 html.Div: Dropdown para seleccionar un gestor
00021 """
00022
00023 return html.Div(
00024     [
00025         html.Div(
00026             [
00027                 dcc.Dropdown(
00028                     options=[],
00029                     value=None,
00030                     id="gestor-dropdown",
00031                     clearable=False,
00032                     placeholder="Seleccione un gestor",
00033                 )
00034             ],
00035             className="select-gestor",
00036         ),
00037     ]
00038 )
```

6.237. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/components/gestor/utils/tabs_gestor.py

Namespaces

- namespace components
- namespace components.gestor
- namespace components.gestor.utils
- namespace components.gestor.utils.tabs_gestor

Funciones

- def components.gestor.utils.tabs_gestor.tabs_gestor ()

6.238. tabs_gestor.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file graphs_indicadores_gestor.py
00003 # @brief Este fichero contiene el componente que contiene las pestañas de la sección "Gestor"
00004 # @version 1.0
00005 # @date 28/04/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martin
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html, dcc
00012 from callbacks.gestor.utils.callback_tabs_gestor import render_content
00013
00014
00015 def tabs_gestor():
00016     """
00017     Crea las pestañas de la sección "Gestor". Cada pestaña contiene un conjunto de gráficos
00018     y tablas con los datos de los alumnos del perfil "Gestor".
00019
00020     Returns:
00021         html.Div: Pestañas de la sección "Gestor"
00022     """
00023
00024     return html.Div(
00025         [
00026             dcc.Tabs(
00027                 id="tabs-gestor",
00028                 value="indicadores-academicos-tab",
00029                 children=[
00030                     dcc.Tab(
00031                         label="Indicadores académicos",
00032                         value="indicadores-academicos-tab",
00033                     ),
00034                     dcc.Tab(
00035                         label="Resultados académicos", value="resultados-academicos-tab"
00036                     ),
00037                     dcc.Tab(label="Riesgo de abandono", value="riesgo-abandono-tab"),
00038                     dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
00039                 ],
00040                 className="tabs",
00041             ),
00042             html.Div(id="tabs-gestor-content"),
00043             dcc.Store(id="selected-gestor-store", storage_type="local"),
00044             dcc.Location(id="url", refresh=False),
00045         ]
00046     )

```

6.239. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_←
Universities_TFG/src/data/db_connector.py

Clases

- class data.db_connector.DatabaseConnector

Namespaces

- namespace data
- namespace data.db_connector

Variables

- `data.db_connector.config = json.load(f)`
- `data.db_connector.db`

6.240. db_connector.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file db_connector.py
00003 # @brief Este archivo contiene la clase DatabaseConnector para conectarse a la base de datos.
00004 # @details Se define la clase DatabaseConnector con un método para ejecutar consultas en la base de
00005 #           datos.
00006 # @version 1.0
00007 # @date 04/05/2024
00008 # @license MIT License
00009 # @author Fabrizio Daniell Perilli Martín
00010 # @email alu0101138589@ull.edu.es
00011 #
00012 from sqlalchemy import create_engine, text
00013 from sqlalchemy.pool import QueuePool
00014 import json
00015
00016
00017 class DatabaseConnector:
00018     """
00019 Clase para conectarse a la base de datos
00020
00021 Attributes:
00022 db_url (str): URL de la base de datos
00023 engine (Engine): Motor de la base de datos
00024
00025 Methods:
00026 execute_query: Ejecuta una consulta en la base de datos
00027 close: Cierra la conexión a la base de datos
00028
00029 """
00030 def __init__(self, dbname, user, password, host, port, pool_size=5, max_overflow=10):
00031     self.db_url = f"postgresql+psycopg2://{user}:{password}@{host}:{port}/{dbname}"
00032     self.engine = create_engine(
00033         self.db_url,
00034         poolclass=QueuePool,
00035         pool_size=pool_size,
00036         max_overflow=max_overflow,
00037     )
00038
00039
00040
00041     def execute_query(self, query, params=None):
00042         """
00043 Ejecuta una consulta en la base de datos
00044
00045 Args:
00046 query (str): Consulta SQL
00047 params (dict): Parámetros de la consulta
00048
00049 Returns:
00050 list: Resultado de la consulta
00051 """
00052     with self.engine.connect() as connection:
00053         result = connection.execute(text(query), params or {})
00054         return result.fetchall()
00055
00056     def close(self):
00057         """
00058 Cierra la conexión a la base de datos
00059
00060 """
00061 self.engine.dispose()
00062
00063
00064 with open("src/data/db_properties.json") as f:
00065     config = json.load(f)
00066
00067 # Se crea la instancia del conector
00068 db = DatabaseConnector(
00069     dbname=config["dbname"],
00070     user=config["user"],
00071     password=config["password"],

```

```
00072     host=config["host"],  
00073     port=config["port"],  
00074 )
```

6.241. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/generate_synthetic_data.py

Namespaces

- namespace `data`
- namespace `data.generate_synthetic_data`

Funciones

- def `data.generate_synthetic_data.generate_asignaturas` (`educacion, min_asignaturas=38`)
- def `data.generate_synthetic_data.generate_unique_cod_asignaturas` (`asignaturas_dict`)
- def `data.generate_synthetic_data.generate_alumnos` (`n`)
- def `data.generate_synthetic_data.generate_curso_aca` (`start_year, end_year`)
- def `data.generate_synthetic_data.generate_matricula` (`alumnos_df`)
- def `data.generate_synthetic_data.generate_asignaturas_matriculadas` (`matricula_df, cod_asignaturas`)
- def `data.generate_synthetic_data.generate_lineas_actas` (`asignaturas_matriculadas_df`)
- def `data.generate_synthetic_data.generate_docentes` (`n, universidades, cod_asignaturas, asignaturas_dict`)
- def `data.generate_synthetic_data.generate_ebau_prueba` (`n, alumnos_ids, universidades`)
- def `data.generate_synthetic_data.generate_egresados` (`alumnos_df, matricula_df, lineas_actas_df`)
- def `data.generate_synthetic_data.generate_gestores` (`n, universidades`)

Variables

- `data.generate_synthetic_data.fake = Faker('es_ES')`
- dictionary `data.generate_synthetic_data.educacion`
- dictionary `data.generate_synthetic_data.universidades_dict`
- def `data.generate_synthetic_data.asignaturas_dict = generate_asignaturas(educacion)`
- def `data.generate_synthetic_data.cod_asignaturas_dict = generate_unique_cod_asignaturas(asignaturas_dict)`
- dictionary `data.generate_synthetic_data.cod_plan_dict = {titulacion: fake.bothify(text='PLAN###??') for titulacion in asignaturas_dict.keys()}`
- def `data.generate_synthetic_data.cursos_academicos = generate_curso_aca(2017, 2024)`
- int `data.generate_synthetic_data.num_alumnos = 5000`
- int `data.generate_synthetic_data.num_docentes = 100`
- int `data.generate_synthetic_data.num_gestores = 50`
- def `data.generate_synthetic_data.alumnos_df = generate_alumnos(num_alumnos)`
- def `data.generate_synthetic_data.matricula_df = generate_matricula(alumnos_df)`
- def `data.generate_synthetic_data.asignaturas_matriculadas_df = generate_asignaturas_matriculadas(matricula_df, cod_asignaturas_dict)`
- def `data.generate_synthetic_data.lineas_actas_df = generate_lineas_actas(asignaturas_matriculadas_df)`
- def `data.generate_synthetic_data.docentes_df = generate_docentes(num_docentes, universidades_dict, cod_asignaturas_dict, asignaturas_dict)`
- def `data.generate_synthetic_data.ebau_prueba_df = generate_ebau_prueba(num_alumnos, alumnos_df['id'].tolist(), universidades_dict)`
- def `data.generate_synthetic_data.egresados_df = generate_egresados(alumnos_df, matricula_df, lineas_actas_df)`
- def `data.generate_synthetic_data.gestores_df = generate_gestores(num_gestores, universidades_dict)`
- `data.generate_synthetic_data.index`

6.242. generate_synthetic_data.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file generate_synthetic_data.py
00003 # @brief Este archivo contiene el código para generar datos sintéticos.
00004 # @details Se generan datos sintéticos para las tablas de la base de datos.
00005 # @version 1.0
00006 # @date 12/06/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martín
00009 # @email alu010138589@ull.edu.es
00010 #
00011
00012 import pandas as pd
00013 import numpy as np
00014 from faker import Faker
00015 import random
00016 from tqdm import tqdm
00017 import os
00018
00019 fake = Faker('es_ES')
00020
00021 # Diccionario con ramas, titulaciones y palabras clave
00022 educacion = {
00023     'Ingeniería y Tecnología': {
00024         'Ingeniería Informática': [
00025             'Programación', 'Algoritmos', 'Estructuras de Datos', 'Bases de Datos', 'Sistemas
Operativos',
00026             'Redes', 'Inteligencia Artificial', 'Seguridad Informática', 'Desarrollo Web', 'Software',
00027             'Ciberseguridad', 'Ingeniería del Software', 'Bases de Datos Avanzadas', 'Arquitectura de
Computadoras',
00028             'Gráficos por Computadora', 'Desarrollo de Aplicaciones', 'Computación en la Nube',
00029             'Blockchain',
00030             'Big Data', 'Analítica de Datos', 'Aprendizaje Automático', 'Computación Paralela',
00031             'Ciencias de la Computación',
00032             'Tecnologías de la Información', 'Interacción Humano-Computadora', 'Teoría de la
Computación', 'Robótica',
00033             'Computación Cuántica', 'Sistemas Embebidos', 'Desarrollo de Videojuegos', 'Visión por
Computador',
00034             'Minería de Datos', 'Internet de las Cosas', 'Bioinformática', 'Sistemas Distribuidos',
00035             'Cómputo de Alto Rendimiento',
00036             'Cálculo Numérico', 'Realidad Virtual', 'Realidad Aumentada'
00037         ],
00038     },
00039     'Ciencias de la Salud': {
00040         'Medicina': [
00041             'Anatomía', 'Fisiología', 'Bioquímica', 'Farmacología', 'Patología',
00042             'Genética', 'Neurología', 'Cardiología', 'Cirugía', 'Dermatología',
00043             'Endocrinología', 'Epidemiología', 'Gastroenterología', 'Ginecología',
00044             'Hematología', 'Infectología', 'Medicina Interna', 'Nefrología',
00045             'Neumología', 'Oncología', 'Pediatría', 'Psiquiatría', 'Radiología',
00046             'Reumatología', 'Urología', 'Medicina Familiar', 'Medicina Preventiva',
00047             'Emergencias Médicas', 'Cuidados Intensivos', 'Medicina Deportiva', 'Medicina Nuclear',
00048             'Oftalmología', 'Otorrinolaringología', 'Rehabilitación', 'Medicina del Trabajo',
00049             'Medicina Forense', 'Geriatría', 'Inmunología', 'Toxicología', 'Anestesiología',
00050             'Nutriología', 'Microbiología', 'Parasitología', 'Patología Clínica', 'Genómica',
00051             'Neurocirugía'
00052         ],
00053     },
00054     'Ciencias Sociales y Jurídicas': {
00055         'Derecho': [
00056             'Derecho Civil', 'Derecho Penal', 'Derecho Constitucional', 'Derecho Administrativo',
00057             'Derecho Laboral',
00058             'Derecho Mercantil', 'Derecho Internacional', 'Derecho Tributario', 'Derecho Ambiental',
00059             'Derecho de Familia',
00060             'Derecho Procesal', 'Derecho Romano', 'Derecho Comparado', 'Derecho Financiero', 'Derecho
Agrario',
00061             'Derecho Informático', 'Derecho de la Competencia', 'Derecho de la Propiedad Intelectual',
00062             'Derecho Bancario',
00063             'Derecho Marítimo', 'Derecho Aéreo', 'Derecho del Consumo', 'Derecho de la Seguridad
Social',
00064             'Derecho Minero',
00065             'Derecho de la Energía', 'Derecho Urbanístico', 'Derecho de la Salud', 'Derecho del
Deporte',
00066             'Derecho de las Telecomunicaciones',
00067             'Derecho Internacional Privado', 'Derecho Internacional Público', 'Derecho de la Unión
Europea',
00068             'Derecho Canónico',
00069             'Derecho Militar', 'Derecho Electoral', 'Derecho Humanitario', 'Derecho Notarial',
00070             'Derecho Penitenciario',
00071             'Derecho Registral'
00072         ],
00073     },
00074     'Humanidades': {
00075         'Historia': [
00076             'Historia Antigua', 'Historia Medieval', 'Historia Moderna', 'Historia Contemporánea',
00077             'Historia de América',

```

```

00067         'Historia de Europa', 'Historia de Asia', 'Historia de África', 'Historia de Oceanía',
00068         'Historia del Arte',
00069         'Historia Económica', 'Historia Política', 'Historia Social', 'Historiografía', 'Historia
00070         de la Ciencia',
00071         'Arqueología', 'Historia Militar', 'Historia de las Religiones', 'Historia Cultural',
00072         'Historia de las Ideas',
00073         'Teoría de la Historia', 'Métodos de Investigación Histórica', 'Historia de la Educación',
00074         'Historia de la Mujer',
00075         'Historia Ambiental', 'Historia Oral', 'Historia de la Tecnología', 'Historia Urbana',
00076         'Historia de la Salud'
00077     ]
00078 }
00079
00080 # Diccionario de universidades con su código correspondiente
00081 universidades_dict = {
00082     'Universidad de Madrid': 'UDM01',
00083     'Universidad de Barcelona': 'UDB02',
00084     'Universidad de La Laguna': 'ULL03',
00085 }
00086
00087
00088 def generate_asignaturas(educacion, min_asignaturas=38):
00089     """
00090     Genera un diccionario con asignaturas aleatorias para cada titulación.
00091
00092     Args:
00093         educacion (dict): Diccionario con ramas, titulaciones y palabras clave.
00094         min_asignaturas (int): Número mínimo de asignaturas por titulación.
00095
00096     Returns:
00097         dict: Diccionario con asignaturas para cada titulación.
00098     """
00099     asignaturas_dict = {}
00100     for rama, titulaciones in educacion.items():
00101         for titulacion, palabras in titulaciones.items():
00102             num_asignaturas = min(min_asignaturas, len(palabras))
00103             asignaturas = random.sample(palabras, num_asignaturas)
00104             asignaturas_dict[titulacion] = asignaturas
00105
00106     return asignaturas_dict
00107
00108 asignaturas_dict = generate_asignaturas(educacion)
00109
00110
00111 def generate_unique_cod_asignaturas(asignaturas_dict):
00112     """
00113     Genera un diccionario con códigos únicos para cada asignatura.
00114
00115     Args:
00116         asignaturas_dict (dict): Diccionario con asignaturas para cada titulación.
00117
00118     Returns:
00119         dict: Diccionario con códigos únicos para cada asignatura.
00120
00121     """
00122     cod_asignaturas = {}
00123     for titulacion, asignaturas in asignaturas_dict.items():
00124         for asignatura in asignaturas:
00125             cod_asignatura = asignatura[:4].upper() + fake.bothify(text='#####')
00126             cod_asignaturas[asignatura] = cod_asignatura
00127
00128     return cod_asignaturas
00129
00130 def generate_alumnos(n):
00131     """
00132     Genera datos sintéticos para la tabla 'alumnos'.
00133
00134     Args:
00135         n (int): Número de alumnos a generar.
00136
00137     Returns:
00138         pd.DataFrame: DataFrame con los datos de los alumnos.
00139
00140     """
00141     ids = [fake.unique.bothify(text='???#####') for _ in range(n)]
00142     anios_nac = np.random.randint(1980, 2000, n)
00143     universidades = random.choices(list(universidades_dict.keys()), k=n)
00144     cod_universidades = [universidades_dict[uni] for uni in universidades]
00145     abandona = random.choices(['si', 'no'], k=n)
00146     data = {
00147         'id': ids,
00148         'anio_nac': anios_nac,
00149         'universidad': universidades,
00150         'cod_universidad': cod_universidades,

```

```

00149         'abandona': abandona
00150     }
00151     return pd.DataFrame(data)
00152
00153
00154 def generate_curso_aca(start_year, end_year):
00155     """
00156 Genera una lista de cursos académicos con el formato 'YYYY/YYYY+1'.
00157
00158     Args:
00159         start_year (int): Año de inicio.
00160         end_year (int): Año de fin.
00161
00162     Returns:
00163         list: Lista de cursos académicos.
00164     """
00165     return [f"{year}/{year+1}" for year in range(start_year, end_year)]
00166
00167 # Generar una lista de cursos académicos
00168 cursos_academicos = generate_curso_aca(2017, 2024)
00169
00170
00171 def generate_matricula(alumnos_df):
00172     """
00173 Genera datos sintéticos para la tabla 'matricula'.
00174
00175     Args:
00176         alumnos_df (pd.DataFrame): DataFrame con los datos de los alumnos.
00177
00178     Returns:
00179         pd.DataFrame: DataFrame con los datos de matrícula.
00180     """
00181 n = len(alumnos_df)
00182 num_matriculas = np.random.randint(1, 6, n)
00183 matricula_data = []
00184 for i, row in tqdm(alumnos_df.iterrows(), total=n, desc="matrícula"):
00185     titulacion = random.choice(list(asignaturas_dict.keys()))
00186     # Encontrar la rama correspondiente a la titulación
00187     rama = next(rama for rama, titulaciones in educacion.items() if titulacion in titulaciones)
00188
00189     common_data = {
00190         'municipio': fake.city(),
00191         'nacionalidad': fake.country(),
00192         'provincia': fake.state(),
00193         'rama': rama,
00194         'sexo': random.choice(['Masculino', 'Femenino']),
00195         'titulacion': titulacion,
00196         'nombre_plan_propio': fake.word(),
00197         'universidad': row['universidad'],
00198         'cod_universidad': row['cod_universidad'],
00199         'cod_tipo_matricula': fake.bothify(text='T##'),
00200         'tipo_matricula': fake.word(),
00201         'cod_plan': cod_plan_dict[titulacion],
00202     }
00203     cursos_aca_usados = set()
00204     for j in range(num_matriculas[i]):
00205         curso_aca = random.choice([ca for ca in cursos_academicos if ca not in cursos_aca_usados])
00206         cursos_aca_usados.add(curso_aca)
00207         nuevo_ingreso = 'si' if j == 0 else 'no'
00208         matricula_data.append({
00209             'indice': len(matricula_data) + 1,
00210             'ambito_isced': fake.bothify(text='IS##'),
00211             'cod_plan': common_data['cod_plan'],
00212             'cod_mec': fake.bothify(text='MEC##'),
00213             'curso_aca': curso_aca,
00214             'municipio': common_data['municipio'],
00215             'nacionalidad': common_data['nacionalidad'],
00216             'nuevo_ingreso': nuevo_ingreso,
00217             'provincia': common_data['provincia'],
00218             'rama': common_data['rama'],
00219             'sexo': common_data['sexo'],
00220             'titulacion': common_data['titulacion'],
00221             'nombre_plan_propio': common_data['nombre_plan_propio'],
00222             'universidad': common_data['universidad'],
00223             'cod_universidad': common_data['cod_universidad'],
00224             'cod_tipo_matricula': common_data['cod_tipo_matricula'],
00225             'tipo_matricula': common_data['tipo_matricula'],
00226             'id': row['id']
00227         })
00228     return pd.DataFrame(matricula_data)
00229
00230
00231 def generate_asignaturas_matriculadas(matricula_df, cod_asignaturas):
00232     """
00233 Genera datos sintéticos para la tabla 'asignaturas_matriculadas'.
00234
00235     Args:

```

```

00236     matricula_df (pd.DataFrame): DataFrame con los datos de matrícula.
00237     cod_asignaturas (dict): Diccionario con códigos únicos para cada asignatura.
00238
00239     Returns:
00240         pd.DataFrame: DataFrame con los datos de asignaturas matriculadas.
00241     """
00242     asignaturas_data = []
00243     for i, row in tqdm(matricula_df.iterrows(), total=len(matricula_df), desc="asignaturas_matriculadas"):
00244         num_asignaturas = min(random.randint(1, 20), len(asignaturas_dict[row['titulacion']]))
00245         asignaturas = random.sample(asignaturas_dict[row['titulacion']], num_asignaturas)
00246         for asignatura in asignaturas:
00247             cod_asignatura = cod_asignaturas[asignatura]
00248             asignaturas_data.append({
00249                 'indice': len(asignaturas_data) + 1,
00250                 'asignatura': asignatura,
00251                 'cod_asignatura': cod_asignatura,
00252                 'cod_plan': row['cod_plan'],
00253                 'curso_aca': row['curso_aca'],
00254                 'id': row['id'],
00255                 'cod_tipologia': fake.bothify(text='T##'),
00256                 'plan': row['titulacion'],
00257                 'tipología': fake.word(),
00258                 'universidad': row['universidad'],
00259                 'cod_universidad': row['cod_universidad']
00260             })
00261     return pd.DataFrame(asignaturas_data)
00262
00263
00264 def generate_lineas_actas(asignaturas_matriculadas_df):
00265     """
00266     Genera datos sintéticos para la tabla 'lineas_actas'.
00267
00268     Args:
00269         asignaturas_matriculadas_df (pd.DataFrame): DataFrame con los datos de asignaturas
00270         matriculadas.
00271
00272     Returns:
00273         pd.DataFrame: DataFrame con los datos de actas.
00274     """
00275     actas_data = []
00276     for i, row in tqdm(asignaturas_matriculadas_df.iterrows(), total=len(asignaturas_matriculadas_df),
00277                         desc="lineas_actas"):
00278         calif_numerica = round(random.uniform(0, 10), 2)
00279         if calif_numerica < 5:
00280             calif = 'Suspenso'
00281         elif calif_numerica < 7:
00282             calif = 'Aprobado'
00283         elif calif_numerica < 9:
00284             calif = 'Notable'
00285         else:
00286             calif = 'Sobresaliente'
00287
00288         actas_data.append({
00289             'indice': len(actas_data) + 1,
00290             'asignatura': row['asignatura'],
00291             'calif_numerica': calif_numerica,
00292             'calif': calif,
00293             'cod_asig': row['cod_asignatura'],
00294             'cod_plan': row['cod_plan'],
00295             'conv': fake.bothify(text='C##'),
00296             'curso_aca': row['curso_aca'],
00297             'id': row['id'],
00298             'plan': row['plan'],
00299             'universidad': row['universidad'],
00300             'cod_universidad': row['cod_universidad']
00301         })
00302     return pd.DataFrame(actas_data)
00303
00304
00305 def generate_docentes(n, universidades, cod_asignaturas, asignaturas_dict):
00306     """
00307     Genera datos sintéticos para la tabla 'docentes'.
00308
00309     Args:
00310         n (int): Número de docentes a generar.
00311         universidades (dict): Diccionario con universidades y sus códigos.
00312         cod_asignaturas (dict): Diccionario con códigos únicos para cada asignatura.
00313         asignaturas_dict (dict): Diccionario con asignaturas para cada titulación.
00314
00315     Returns:
00316         pd.DataFrame: DataFrame con los datos de los docentes.
00317     """
00318     docentes_data = []
00319     for i in tqdm(range(n), desc="docentes"):
00320         id_docente = fake.bothify(text='D#####')
00321         universidad, cod_universidad = random.choice(list(universidades.items()))

```

```

00321
00322     num_asignaturas = random.randint(1, 5)
00323     asignaturas_seleccionadas = random.sample(list(cod_asignaturas.items()), num_asignaturas)
00324
00325     for asignatura, cod_asignatura in asignaturas_seleccionadas:
00326         titulacion = next(titulacion for titulacion, asignaturas in asignaturas_dict.items() if
00327             asignatura in asignaturas)
00328         cod_plan = cod_plan_dict[titulacion]
00329         docentes_data.append({
00330             'indice': len(docentes_data) + 1,
00331             'id_docente': id_docente,
00332             'cod_universidad': cod_universidad,
00333             'universidad': universidad,
00334             'titulacion': titulacion,
00335             'cod_plan': cod_plan,
00336             'cod_asignatura': cod_asignatura,
00337             'asignatura': asignatura,
00338             'curso_aca': random.choice(cursos_academicos)
00339         })
00340
00341     return pd.DataFrame(docentes_data)
00342
00343 def generate_ebau_prueba(n, alumnos_ids, universidades):
00344     """
00345     Genera datos sintéticos para la tabla 'ebau_prueba'.
00346
00347     Args:
00348         n (int): Número de registros a generar.
00349         alumnos_ids (list): Lista de IDs de alumnos.
00350         universidades (dict): Diccionario con universidades y sus códigos.
00351
00352     Returns:
00353         pd.DataFrame: DataFrame con los datos de las pruebas EBAU.
00354     """
00355     assert n == len(alumnos_ids), "El número de registros debe ser igual al número de alumnos"
00356     data = {
00357         'indice': range(1, n + 1),
00358         'conv': [fake.bothify(text='C##') for _ in range(n)],
00359         'curso_aca': np.random.choice(cursos_academicos, n),
00360         'especialidad': [fake.word() for _ in range(n)],
00361         'id': alumnos_ids,
00362         'nota_bach': np.round(np.random.uniform(5, 10, n), 2),
00363         'nota_def': np.round(np.random.uniform(5, 10, n), 2),
00364         'nota_prue': np.round(np.random.uniform(5, 10, n), 2),
00365         'universidad': np.random.choice(list(universidades.keys()), n),
00366     }
00367     data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]
00368
00369     return pd.DataFrame(data)
00370
00371
00372 def generate_egresados(alumnos_df, matricula_df, lineas_actas_df):
00373     """
00374     Genera datos sintéticos para la tabla 'egresados'.
00375
00376     Args:
00377         alumnos_df (pd.DataFrame): DataFrame con los datos de los alumnos.
00378         matricula_df (pd.DataFrame): DataFrame con los datos de matrícula.
00379         lineas_actas_df (pd.DataFrame): DataFrame con los datos de actas.
00380
00381     Returns:
00382         pd.DataFrame: DataFrame con los datos de los egresados.
00383     """
00384     aprobados = lineas_actas_df[lineas_actas_df['calif_numerica'] >= 5].groupby('id').size()
00385     egresados_ids = aprobados[aprobados >= 38].index.tolist()
00386     alumnos_no_abandona = alumnos_df[alumnos_df['abandona'] == 'no']['id'].tolist()
00387     egresados_ids = [id for id in egresados_ids if id in alumnos_no_abandona]
00388
00389     egresados_data = []
00390     for i, id_alumno in tqdm(enumerate(egresados_ids), total=len(egresados_ids), desc="egresados"):
00391         universidad, cod_universidad = alumnos_df.loc[alumnos_df['id'] == id_alumno, ['universidad',
00392             'cod_universidad']].values[0]
00393         cod_plan = matricula_df.loc[matricula_df['id'] == id_alumno, 'cod_plan'].values[0]
00394         ultimo_curso_aca = matricula_df.loc[matricula_df['id'] == id_alumno, 'curso_aca'].max()
00395         egresados_data.append({
00396             'indice': i + 1,
00397             'cod_plan': cod_plan,
00398             'curso_aca': ultimo_curso_aca,
00399             'id': id_alumno,
00400             'nota_media': round(random.uniform(5, 10), 2),
00401             'plan': fake.word(),
00402             'universidad': universidad,
00403             'cod_universidad': cod_universidad
00404         })
00405
00406     return pd.DataFrame(egresados_data)
00407

```

6.243 Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/queries.py 349

```
00406
00407 def generate_gestores(n, universidades):
00408     """
00409     Genera datos sintéticos para la tabla 'gestores'.
00410
00411     Args:
00412         n (int): Número de gestores a generar.
00413         universidades (dict): Diccionario con universidades y sus códigos.
00414
00415     Returns:
00416         pd.DataFrame: DataFrame con los datos de los gestores.
00417     """
00418 data = {
00419     'gestor_id': range(1, n + 1),
00420     'universidad': np.random.choice(list(universidades.keys()), n)
00421 }
00422 data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]
00423 return pd.DataFrame(data)
00424
00425
00426 print("Generando datos....")
00427 num_alumnos = 5000
00428 num_docentes = 100
00429 num_gestores = 50
00430
00431 #Crea un directorio llamado src/data/csv si no existe
00432
00433 if not os.path.exists('src/data/csv'):
00434     os.makedirs('src/data/csv')
00435
00436 alumnos_df = generate_alumnos(num_alumnos)
00437 matricula_df = generate_matricula(alumnos_df)
00438 asignaturas_matriculadas_df = generate_asignaturas_matriculadas(matricula_df, cod_asignaturas_dict)
00439 lineas_actas_df = generate_lineas_actas(asignaturas_matriculadas_df)
00440 docentes_df = generate_docentes(num_docentes, universidades_dict, cod_asignaturas_dict,
00441 asignaturas_dict)
00441 ebau_prueba_df = generate_ebau_prueba(num_alumnos, alumnos_df['id'].tolist(), universidades_dict)
00442 egresados_df = generate_egresados(alumnos_df, matricula_df, lineas_actas_df)
00443 gestores_df = generate_gestores(num_gestores, universidades_dict)
00444
00445 # Guarda datos en archivos CSV
00446 alumnos_df.to_csv('src/data/csv/alumnos.csv', index=False)
00447 matricula_df.to_csv('src/data/csv/matricula.csv', index=False)
00448 asignaturas_matriculadas_df.to_csv('src/data/csv/asignaturas_matriculadas.csv', index=False)
00449 lineas_actas_df.to_csv('src/data/csv/lineas_actas.csv', index=False)
00450 docentes_df.to_csv('src/data/csv/docentes.csv', index=False)
00451 ebau_prueba_df.to_csv('src/data/csv/ebau_prueba.csv', index=False)
00452 egresados_df.to_csv('src/data/csv/egresados.csv', index=False)
00453 gestores_df.to_csv('src/data/csv/gestores.csv', index=False)
00454
00455 print("Datos generados correctamente.")
```

6.243. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/queries.py

Namespaces

- namespace [data](#)
- namespace [data.queries](#)

Funciones

- def [data.queries.check_data](#) (query, params)
- def [data.queries.cache_query](#) (func)
- def [data.queries.alumnos_all](#) ()
- def [data.queries.resumen_alumno](#) (alumno_id, titulacion)
- def [data.queries.resumen_gestor](#) (gestor_id)
- def [data.queries.nota_media_alumno_titulacion](#) (alumno_id, titulacion)
- def [data.queries.curso_academico_alumnado](#) (alumno_id, titulacion)
- def [data.queries.asignaturas_matriculadas](#) (alumno_id, curso_academico, titulacion)

- def `data.queries.titulacion_alumnado` (`alumno_id`)
- def `data.queries.asignaturas_superadas` (`alumno_id, curso_academico, titulacion`)
- def `data.queries.calif_cualitativa_asignatura` (`alumno_id, curso_academico, titulacion`)
- def `data.queries.calif_numerica_asignatura` (`alumno_id, curso_academico, titulacion`)
- def `data.queries.asignaturas_matriculadas_y_superadas` (`alumno_id, curso_academico, titulacion`)
- def `data.queries.asignaturas_superadas_media_abandono` (`curso_academico, asignaturas_matriculadas, titulacion, cod_universidad`)
- def `data.queries.calif_cualitativa_comparativa` (`curso_academico, asignaturas_matriculadas, titulacion, cod_universidad`)
- def `data.queries.calif_cualitativa_alumno_asignaturas` (`alumno_id, curso_academico, asignaturas_matriculadas, titulacion`)
- def `data.queries.nota_media_general_mi_nota` (`curso_academico, asignaturas_matriculadas, alumno_id, titulacion, cod_universidad`)
- def `data.queries.alumnos_repetidores_nuevos` (`docente_id, curso_academico, asignaturas`)
- def `data.queries.asignaturas_docente` (`id_docente, titulacion`)
- def `data.queries.curso_academico_docente` (`id_docente, asignatura`)
- def `data.queries.titulacion_docente` (`id_docente`)
- def `data.queries.resumen_docente` (`id_docente, titulacion`)
- def `data.queries.docentes_all` ()
- def `data.queries.alumnos_genero_docente` (`id_docente, asignaturas, curso_academico`)
- def `data.queries.alumnos_nota_media_docente` (`asignaturas, curso_academico`)
- def `data.queries.alumnos_nota_cualitativa_docente` (`asignaturas, curso_academico`)
- def `data.queries.curso_academico_actas_titulacion` (`titulacion`)
- def `data.queries.asignaturas_actas_titulacion` (`titulacion, curso_academico`)
- def `data.queries.calif_all_cualitativa_asignaturas` (`titulacion, curso_academico, asignaturas`)
- def `data.queries.calif_media_asignaturas` (`titulacion, curso_academico, asignaturas`)
- def `data.queries.gestores_all` ()
- def `data.queries.numero_alumnos_matriculados_universidad` (`universidad`)
- def `data.queries.universidades_gestor` (`gestor_id`)
- def `data.queries.curso_academico_universidad` (`cod_universidad`)
- def `data.queries.titulaciones_universidad_gestor` (`cod_universidad, curso_academico`)
- def `data.queries.alumnos_nuevo_ingreso_genero_titulacion` (`curso_academico, titulaciones, cod_universidad`)
- def `data.queries.alumnos_egresados_genero_titulacion` (`cod_universidad, curso_academico, titulaciones`)
- def `data.queries.alumnos_egresados_nacionalidad_titulacion` (`cod_universidad, curso_academico, titulaciones`)
- def `data.queries.alumnos_nuevo_ingreso_nacionalidad_titulacion` (`cod_universidad, curso_academico, titulaciones`)
- def `data.queries.nota_media_acceso_titulacion` (`cod_universidad`)
- def `data.queries.duracion_media_estudios_nota_gestor` (`cod_universidad`)
- def `data.queries.cursos_academicos_egresados` (`cod_universidad`)
- def `data.queries.tasa_abandono_titulacion_gestor` (`cod_universidad, curso_academico`)
- def `data.queries.tasa_graduacion_titulacion_gestor` (`cod_universidad, curso_academico`)
- def `data.queries.universidad_alumno` (`alumno_id`)
- def `data.queries.universidades_docente` (`id_docente`)
- def `data.queries.data_for_model` ()

6.244. queries.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file queries.py
00003 # @brief Este archivo contiene funciones para ejecutar consultas en la base de datos.
00004 # @version 1.0
00005 # @date 19/06/2024
```

```
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 import pandas as pd
00012 from functools import lru_cache
00013 from data.queries_dictionary import queries
00014 from data.db_connector import db
00015
00016 def check_data(query, params):
00017     """
00018 Ejecuta una consulta en la base de datos y maneja los errores.
00019
00020 Args:
00021 query (str): Consulta SQL
00022 params (dict): Parámetros de la consulta
00023
00024 Returns:
00025 list: Resultado de la consulta
00026 """
00027 try:
00028     data = db.execute_query(query, params)
00029 except Exception as e:
00030     print("Query execution failed: ", e)
00031     return []
00032
00033
00034
00035 def cache_query(func):
00036     """
00037 Decorador para cachear los resultados de las consultas.
00038
00039 Args:
00040 func: Función a decorar
00041
00042 Returns:
00043 wrapper: Función decorada
00044 """
00045 @lru_cache(maxsize=32)
00046     def wrapper(*args, **kwargs):
00047         return func(*args, **kwargs)
00048
00049
00050
00051
00052 @cache_query
00053 def alumnos_all():
00054     query = queries["alumnado"]["common"]["alumnos_all"]
00055     return check_data(query, {})
00056
00057
00058 @cache_query
00059 def resumen_alumno(alumno_id, titulacion):
00060     query = queries["alumnado"]["common"]["resumen_alumno"]
00061     params = {"alumno_id": alumno_id, "titulacion": titulacion}
00062
00063
00064
00065
00066 @cache_query
00067 def resumen_gestor(gestor_id):
00068     query = queries["gestor"]["common"]["resumen_gestor"]
00069     params = {"gestor_id": gestor_id}
00070
00071
00072
00073
00074 @cache_query
00075 def nota_media_alumno_titulacion(alumno_id, titulacion):
00076     query = queries["alumnado"]["common"]["nota_media_alumno_titulacion"]
00077     params = {"alumno_id": alumno_id, "titulacion": titulacion}
00078
00079     data = check_data(query, params)
00080
00081     if not data:
00082         return "No disponible"
00083
00084     return round(data[0][0], 2)
00085
00086
00087 @cache_query
00088 def curso_academico_alumnado(alumno_id, titulacion):
00089     query = queries["alumnado"]["filters"]["curso_academico_alumnado"]
00090     params = {"alumno_id": alumno_id, "titulacion": titulacion}
00091
00092
00093     return check_data(query, params)
```

```
00093
00094
00095 @cache_query
00096 def asignaturas_matriculadas(alumno_id, curso_academico, titulacion):
00097     query = queries["alumnado"]["filters"]["asignaturas_matriculadas"]
00098     params = {
00099         "alumno_id": alumno_id,
00100         "curso_academico": curso_academico,
00101         "titulacion": titulacion,
00102     }
00103
00104     return check_data(query, params)
00105
00106
00107 @cache_query
00108 def titulacion_alumnado(alumno_id):
00109     query = queries["alumnado"]["filters"]["titulacion_alumnado"]
00110     params = {"alumno_id": alumno_id}
00111
00112     return check_data(query, params)
00113
00114
00115 @cache_query
00116 def asignaturas_superadas(alumno_id, curso_academico, titulacion):
00117     query = queries["alumnado"]["graphs"]["personal"][
00118         "curso_academico_asignaturas_superadas"
00119     ]
00120     params = {
00121         "alumno_id": alumno_id,
00122         "curso_academico": curso_academico,
00123         "titulacion": titulacion,
00124     }
00125
00126     return check_data(query, params)
00127
00128
00129 @cache_query
00130 def calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion):
00131     query = queries["alumnado"]["graphs"]["personal"]["calif_cualitativa_asignatura"]
00132     params = {
00133         "alumno_id": alumno_id,
00134         "curso_academico": curso_academico,
00135         "titulacion": titulacion,
00136     }
00137
00138     return check_data(query, params)
00139
00140
00141 @cache_query
00142 def calif_numerica_asignatura(alumno_id, curso_academico, titulacion):
00143     query = queries["alumnado"]["graphs"]["personal"]["calif_numerica_asignatura"]
00144     params = {
00145         "alumno_id": alumno_id,
00146         "curso_academico": curso_academico,
00147         "titulacion": titulacion,
00148     }
00149
00150     return check_data(query, params)
00151
00152
00153 @cache_query
00154 def asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion):
00155     query = queries["alumnado"]["graphs"]["personal"][
00156         "asignaturas_matriculadas_y_superadas"
00157     ]
00158     params = {
00159         "alumno_id": alumno_id,
00160         "curso_academico": curso_academico,
00161         "titulacion": titulacion,
00162     }
00163
00164     return check_data(query, params)
00165
00166
00167 @cache_query
00168 def asignaturas_superadas_media_abandono(
00169     curso_academico, asignaturas_matriculadas, titulacion, cod_universidad
00170 ):
00171     query = queries["alumnado"]["graphs"]["general"][
00172         "asignaturas_superadas_media_abandono"
00173     ]
00174     params = {
00175         "curso_academico": curso_academico,
00176         "asignaturas_matriculadas": asignaturas_matriculadas,
00177         "titulacion": titulacion,
00178         "cod_universidad": cod_universidad,
00179     }
```

```
00180     return check_data(query, params)
00181
00182
00183
00184 @cache_query
00185 def calif_cualitativa_comparativa(
00186     curso_academico, asignaturas_matriculadas, titulacion, cod_universidad
00187 ):
00188     query = queries["alumnado"]["graphs"]["general"]["calif_cualitativa_comparativa"]
00189     params = {
00190         "curso_academico": curso_academico,
00191         "asignaturas_matriculadas": asignaturas_matriculadas,
00192         "titulacion": titulacion,
00193         "cod_universidad": cod_universidad,
00194     }
00195
00196     return check_data(query, params)
00197
00198
00199 @cache_query
00200 def calif_cualitativa_alumno_asignaturas(
00201     alumno_id, curso_academico, asignaturas_matriculadas, titulacion
00202 ):
00203     query = queries["alumnado"]["graphs"]["general"][
00204         "calif_cualitativa_alumno_asignaturas"
00205     ]
00206     params = {
00207         "alumno_id": alumno_id,
00208         "curso_academico": curso_academico,
00209         "asignaturas_matriculadas": asignaturas_matriculadas,
00210         "titulacion": titulacion,
00211     }
00212
00213     return check_data(query, params)
00214
00215
00216 @cache_query
00217 def nota_media_general_mi_nota(
00218     curso_academico, asignaturas_matriculadas, alumno_id, titulacion, cod_universidad
00219 ):
00220     query = queries["alumnado"]["graphs"]["general"]["nota_media_general_mi_nota"]
00221     params = {
00222         "curso_academico": curso_academico,
00223         "asignaturas_matriculadas": asignaturas_matriculadas,
00224         "alumno_id": alumno_id,
00225         "titulacion": titulacion,
00226         "cod_universidad": cod_universidad,
00227     }
00228
00229     return check_data(query, params)
00230
00231
00232 @cache_query
00233 def alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas):
00234     query = queries["docente"]["graphs"]["personal"]["alumnos_repetidores_nuevos"]
00235     params = {
00236         "docente_id": docente_id,
00237         "curso_academico": curso_academico,
00238         "asignaturas": asignaturas,
00239     }
00240
00241     return check_data(query, params)
00242
00243
00244 @cache_query
00245 def asignaturas_docente(id_docente, titulacion):
00246     query = queries["docente"]["filters"]["asignaturas_docente"]
00247     params = {"id_docente": id_docente, "titulacion": titulacion}
00248
00249     return check_data(query, params)
00250
00251
00252 @cache_query
00253 def curso_academico_docente(id_docente, asignatura):
00254     query = queries["docente"]["filters"]["curso_academico_docente"]
00255     params = {"id_docente": id_docente, "asignatura": asignatura}
00256
00257     return check_data(query, params)
00258
00259
00260 @cache_query
00261 def titulacion_docente(id_docente):
00262     query = queries["docente"]["filters"]["titulacion_docente"]
00263     params = {"id_docente": id_docente}
00264
00265     return check_data(query, params)
00266
```

```
00267
00268 @cache_query
00269 def resumen_docente(id_docente, titulacion):
00270     query = queries["docente"]["common"]["resumen_docente"]
00271     params = {"id_docente": id_docente, "titulacion": titulacion}
00272
00273     return check_data(query, params)
00274
00275
00276 @cache_query
00277 def docentes_all():
00278     query = queries["docente"]["common"]["docentes_all"]
00279     return check_data(query, {})
00280
00281
00282 @cache_query
00283 def alumnos_genero_docente(id_docente, asignaturas, curso_academico):
00284     query = queries["docente"]["graphs"]["personal"]["alumnos_genero_docente"]
00285     params = {
00286         "id_docente": id_docente,
00287         "asignaturas": asignaturas,
00288         "curso_academico": curso_academico,
00289     }
00290
00291     return check_data(query, params)
00292
00293
00294 @cache_query
00295 def alumnos_nota_media_docente(asignaturas, curso_academico):
00296     query = queries["docente"]["graphs"]["personal"]["alumnos_nota_media_docente"]
00297     params = {"asignaturas": asignaturas, "curso_academico": curso_academico}
00298
00299     return check_data(query, params)
00300
00301
00302 @cache_query
00303 def alumnos_nota_cualitativa_docente(asignaturas, curso_academico):
00304     query = queries["docente"]["graphs"]["personal"]["alumnos_nota_cualitativa_docente"]
00305     params = {"asignaturas": asignaturas, "curso_academico": curso_academico}
00306
00307     return check_data(query, params)
00308
00309
00310 @cache_query
00311 def curso_academico_actas_titulacion(titulacion):
00312     query = queries["docente"]["filters"]["curso_academico_actas_titulacion"]
00313     params = {"titulacion": titulacion}
00314
00315     return check_data(query, params)
00316
00317
00318 @cache_query
00319 def asignaturas_actas_titulacion(titulacion, curso_academico):
00320     query = queries["docente"]["filters"]["asignaturas_actas_titulacion"]
00321     params = {"titulacion": titulacion, "curso_academico": curso_academico}
00322
00323     return check_data(query, params)
00324
00325
00326 @cache_query
00327 def calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignaturas):
00328     query = queries["docente"]["graphs"]["general"]["calif_all_cualitativa_asignaturas"]
00329     params = {
00330         "titulacion": titulacion,
00331         "curso_academico": curso_academico,
00332         "asignaturas": asignaturas,
00333     }
00334
00335     return check_data(query, params)
00336
00337
00338 @cache_query
00339 def calif_media_asignaturas(titulacion, curso_academico, asignaturas):
00340     query = queries["docente"]["graphs"]["general"]["calif_media_asignaturas"]
00341     params = {
00342         "titulacion": titulacion,
00343         "curso_academico": curso_academico,
00344         "asignaturas": asignaturas,
00345     }
00346
00347     return check_data(query, params)
00348
00349
00350 @cache_query
00351 def gestores_all():
00352     query = queries["gestor"]["common"]["gestores_all"]
00353     return check_data(query, {})
```

```
00354
00355
00356 @cache_query
00357 def numero_alumnos_matriculados_universidad(universidad):
00358     query = queries["gestor"]["common"]["numero_alumnos_matriculados_universidad"]
00359     params = {"universidad": universidad}
00360
00361     return check_data(query, params)
00362
00363
00364 @cache_query
00365 def universidades_gestor(gestor_id):
00366     query = queries["gestor"]["common"]["universidades_gestor"]
00367     params = {"gestor_id": gestor_id}
00368
00369     return check_data(query, params)
00370
00371
00372 @cache_query
00373 def curso_academico_universidad(cod_universidad):
00374     query = queries["gestor"]["filters"]["curso_academico_universidad"]
00375     params = {"cod_universidad": cod_universidad}
00376
00377     return check_data(query, params)
00378
00379
00380 @cache_query
00381 def titulaciones_universidad_gestor(cod_universidad, curso_academico):
00382     query = queries["gestor"]["filters"]["titulaciones_universidad_gestor"]
00383     params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00384
00385     return check_data(query, params)
00386
00387
00388 @cache_query
00389 def alumnos_nuevo_ingreso_genero_titulacion(
00390     curso_academico, titulaciones, cod_universidad
00391 ):
00392     query = queries["gestor"]["graphs"]["indicadores"][
00393         "alumnos_nuevo_ingreso_genero_titulacion"
00394     ]
00395     params = {
00396         "curso_academico": curso_academico,
00397         "titulaciones": titulaciones,
00398         "cod_universidad": cod_universidad,
00399     }
00400
00401     return check_data(query, params)
00402
00403
00404 @cache_query
00405 def alumnos_egresados_genero_titulacion(cod_universidad, curso_academico, titulaciones):
00406     query = queries["gestor"]["graphs"]["indicadores"][
00407         "alumnos_egresados_genero_titulacion"
00408     ]
00409     params = {
00410         "cod_universidad": cod_universidad,
00411         "curso_academico": curso_academico,
00412         "titulaciones": titulaciones,
00413     }
00414
00415     return check_data(query, params)
00416
00417
00418 @cache_query
00419 def alumnos_egresados_nacionalidad_titulacion(
00420     cod_universidad, curso_academico, titulaciones
00421 ):
00422     query = queries["gestor"]["graphs"]["indicadores"][
00423         "alumnos_egresados_nacionalidad_titulacion"
00424     ]
00425     params = {
00426         "cod_universidad": cod_universidad,
00427         "curso_academico": curso_academico,
00428         "titulaciones": titulaciones,
00429     }
00430
00431     return check_data(query, params)
00432
00433
00434 @cache_query
00435 def alumnos_nuevo_ingreso_nacionalidad_titulacion(
00436     cod_universidad, curso_academico, titulaciones
00437 ):
00438     query = queries["gestor"]["graphs"]["indicadores"][
00439         "alumnos_nuevo_ingreso_nacionalidad_titulacion"
00440     ]
```

```

00441     params = {
00442         "cod_universidad": cod_universidad,
00443         "curso_academico": curso_academico,
00444         "titulaciones": titulaciones,
00445     }
00446
00447     return check_data(query, params)
00448
00449
00450     @cache_query
00451     def nota_media_acceso_titulacion(cod_universidad):
00452         query = queries["gestor"]["graphs"]["resultados"]["nota_media_acceso_titulacion"]
00453         params = {"cod_universidad": cod_universidad}
00454
00455         return check_data(query, params)
00456
00457
00458     @cache_query
00459     def duracion_media_estudios_nota_gestor(cod_universidad):
00460         query = queries["gestor"]["graphs"]["resultados"][
00461             "duracion_media_estudios_nota_gestor"
00462         ]
00463         params = {"cod_universidad": cod_universidad}
00464
00465         return check_data(query, params)
00466
00467
00468     @cache_query
00469     def cursos_academicos_egresados(cod_universidad):
00470         query = queries["gestor"]["common"]["cursos_academicos_de_egresados"]
00471         params = {"cod_universidad": cod_universidad}
00472
00473         return check_data(query, params)
00474
00475
00476     @cache_query
00477     def tasa_abandono_titulacion_gestor(cod_universidad, curso_academico):
00478         query = queries["gestor"]["graphs"]["riesgo_abandono"][
00479             "tasa_abandono_titulacion_gestor"
00480         ]
00481         params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00482
00483         return check_data(query, params)
00484
00485
00486     @cache_query
00487     def tasa_graduacion_titulacion_gestor(cod_universidad, curso_academico):
00488         query = queries["gestor"]["graphs"]["riesgo_abandono"][
00489             "tasa_graduacion_titulacion_gestor"
00490         ]
00491         params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00492
00493         return check_data(query, params)
00494
00495
00496     @cache_query
00497     def universidad_alumno(alumno_id):
00498         query = queries["alumnado"]["common"]["universidad_alumno"]
00499         params = {"alumno_id": alumno_id}
00500
00501         return check_data(query, params)
00502
00503
00504     @cache_query
00505     def universidades_docente(id_docente):
00506         query = queries["docente"]["common"]["universidades_docente"]
00507         params = {"id_docente": id_docente}
00508
00509         return check_data(query, params)
00510
00511     @cache_query
00512     def data_for_model():
00513         query = queries["alumnado"]["common"]["data_for_model"]
00514         data = check_data(query, {})
00515
00516         if not data:
00517             return []
00518
00519         df = pd.DataFrame(data)
00520
00521         # Ajustar los tipos de datos de las columnas
00522         df = df.astype({
00523             'id': str,
00524             'anio_nac': int,
00525             'nacionalidad': str,
00526             'sexo': str,
00527             'titulacion': str,

```

```
00528     'nota_def_acceso': float,
00529     'nota_media': float,
00530     'abandona': str
00531   })
00532   return df
00533
00534 @cache_query
00535 def resumen_alumno(alumno_id, titulacion):
00536   query = queries["alumnado"]["common"]["resumen_alumno"]
00537   params = {"alumno_id": alumno_id, "titulacion": titulacion}
00538
00539   return check_data(query, params)
```

6.245. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/data/queries_dictionary.py

Namespaces

- namespace [data](#)
- namespace [data.queries_dictionary](#)

Variables

- dictionary [data.queries_dictionary.queries](#)

6.246. queries_dictionary.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file queries_dictionary.py
00003 # @brief Este archivo contiene el diccionario de todas las consultas SQL de la aplicación.
00004 # @details Se definen las consultas SQL para obtener los datos de la base de datos.
00005 # @version 1.0
00006 # @date 19/05/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martín
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 queries = {
00013   "alumnado": {
00014     "common": {
00015       # Consulta para obtener los alumnos.
00016       "alumnos_all": """
00017 SELECT id FROM alumnos
00018 ORDER BY id
00019 LIMIT 50;
00020 """,
00021     "resumen_alumno": """
00022 SELECT DISTINCT alu.id, ma.universidad, titulacion, abandona
00023 FROM alumnos alu
00024 JOIN matricula ma ON ma.id = alu.id
00025 WHERE alu.id = :alumno_id AND ma.titulacion = :titulacion;
00026 """,
00027     # Consulta para calcular la media de las calificaciones de un alumno en una titulación específica.
00028     "nota_media_alumno_titulacion": """
00029 SELECT AVG(li.calif_numerica) AS media_calif
00030 FROM public.lineas_actas li
00031 JOIN public.matricula m ON m.id = li.id AND m.cod_plan = li.cod_plan
00032 WHERE li.id = :alumno_id
00033 AND li.calif_numerica >= 5
00034 AND m.titulacion = :titulacion;
00035 """,
00036     "universidad_alumno": """
00037 SELECT cod_universidad, universidad
00038 FROM matricula
```

```

00039 WHERE id = :alumno_id;
00040 """
00041 # Consulta para obtener los datos necesarios para el modelo de aprendizaje automático.
00042         "data_for_model": """
00043 WITH average_grades AS (
00044     SELECT
00045         ma.id,
00046         ma.titulacion,
00047         AVG(la.calif_numerica) AS nota_media
00048     FROM
00049         public.lineas_actas la
00050     JOIN
00051         matricula ma ON la.id = ma.id AND ma.cod_plan = la.cod_plan
00052     WHERE
00053         la.calif_numerica >= 5
00054     GROUP BY
00055         ma.id, ma.titulacion
00056 )
00057     SELECT DISTINCT
00058         alu.id,
00059         alu.anio_nac,
00060         ma.nacionalidad,
00061         ma.sexo,
00062         ma.titulacion,
00063         ebau.nota_def AS nota_def_acceso,
00064         alu.abandona,
00065         ag.nota_media
00066     FROM
00067         alumnos alu
00068     JOIN
00069         matricula ma ON alu.id = ma.id
00070     JOIN
00071         ebau_prueba ebau ON ebau.id = ma.id
00072     LEFT JOIN
00073         average_grades ag ON ma.id = ag.id AND ma.titulacion = ag.titulacion
00074     ORDER BY
00075         alu.id;
00076 """
00077 },
00078 "filters": {
00079     # Consulta para obtener los cursos académicos en los que un alumno está matriculado en una
00080     # titulación específica.
00081         "curso_academico_alumnado": """
00082     SELECT curso_aca
00083     FROM matricula
00084     WHERE id = :alumno_id AND titulacion = :titulacion;
00085 """
00086     # Consulta para obtener las asignaturas matriculadas por un alumno en un curso académico y titulación
00087     # específicos.
00088         "asignaturas_matriculadas": """
00089     SELECT DISTINCT AM.asignatura
00090     FROM asignaturas_matriculadas AM
00091     JOIN matricula MA ON AM.id = MA.id AND AM.cod_plan = MA.cod_plan
00092     WHERE AM.id = :alumno_id AND AM.curso_aca IN :curso_academico AND MA.titulacion = :titulacion;
00093 """
00094     # Consulta para obtener las titulaciones de un alumno.
00095         "titulacion_alumnado": """
00096     SELECT DISTINCT titulacion
00097     FROM matricula
00098     WHERE id = :alumno_id;
00099 """
00100     "personal": {
00101         # Consulta para obtener las asignaturas superadas por un alumno en un curso académico
00102         # y titulación específicos.
00103             "curso_academico_asignaturas_superadas": """
00104     SELECT li.curso_aca AS curso_academico, COUNT(DISTINCT li.asignatura) AS n_asig_superadas
00105     FROM lineas_actas li
00106     JOIN matricula ma ON li.id = ma.id AND li.cod_plan = ma.cod_plan
00107     WHERE li.id = :alumno_id AND li.calif_numerica >= 5 AND li.curso_aca IN :curso_academico
00108     AND ma.titulacion = :titulacion
00109     GROUP BY li.curso_aca
00110     ORDER BY li.curso_aca;
00111 """
00112     # Consulta para obtener la calificación cualitativa más alta por curso académico y asignatura de un
00113     # alumno.
00114         "calif_cualitativa_asignatura": """
00115     WITH RankedGrades AS (
00116         SELECT li.curso_aca, li.calif,
00117             ROW_NUMBER() OVER (PARTITION BY li.curso_aca, li.asignatura ORDER BY
00118             CASE
00119                 WHEN li.calif = 'Sobresaliente' THEN 5
00120                     WHEN li.calif = 'Notable' THEN 4
00121                     WHEN li.calif = 'Aprobado' THEN 3
00122                     WHEN li.calif = 'Suspens' THEN 2
00123                     WHEN li.calif = 'No presentado' THEN 1

```

```

00122                     ELSE 0
00123                         END DESC) AS rk
00124                 FROM lineas_actas li
00125                 JOIN matricula ma ON li.id = ma.id AND li.cod_plan = ma.cod_plan
00126                 WHERE li.id = :alumno_id
00127                 AND li.curso_aca IN :curso_academico
00128                 AND ma.titulacion = :titulacion
00129             )
00130         SELECT curso_aca AS curso_academico, calif AS calificaciones, COUNT(*) AS
00131             n_calificaciones
00132             FROM RankedGrades
00133             WHERE rk = 1
00134             GROUP BY curso_aca, calif
00135             ORDER BY curso_aca;
00136     """
00136 # Consulta para obtener la calificación numérica de las asignaturas matriculadas por un alumno.
00137     "calif_numerica_asignatura": """
00138     SELECT li.asignatura, MAX(li.calif_numerica) AS calif_numerica
00139     FROM lineas_actas li
00140     JOIN matricula ma ON li.id = ma.id AND li.cod_plan = ma.cod_plan
00141     WHERE li.id = :alumno_id
00142     AND li.curso_aca IN :curso_academico
00143     AND ma.titulacion = :titulacion
00144     GROUP BY li.asignatura
00145     ORDER BY li.asignatura;
00146 """
00147 # Consulta para obtener el número de asignaturas matriculadas y superadas por un alumno.
00148     "asignaturas_matriculadas_y_superadas": """
00149     SELECT
00150         AM.curso_aca AS "curso_academico",
00151             COUNT(DISTINCT AM.asignatura) AS "Total asignaturas matriculadas",
00152             COUNT(DISTINCT CASE WHEN LA.calif_numerica >= 5 THEN LA.asignatura ELSE NULL
00153             END) AS "Total asignaturas superadas"
00154             FROM asignaturas_matriculadas AM
00155             LEFT JOIN lineas_actas LA ON AM.cod_asignatura = LA.cod_asig AND AM.id = LA.id
00156             AND AM.curso_aca = LA.curso_aca
00157             JOIN matricula MA ON AM.id = MA.id AND AM.cod_plan = MA.cod_plan
00158             WHERE
00159                 AM.id = :alumno_id
00160                 AND AM.curso_aca IN :curso_academico
00161                 AND MA.titulacion = :titulacion
00162             GROUP BY
00163                 AM.curso_aca;
00164 """
00164 "general": {
00165     # Esta consulta devuelve el id de los alumnos, nota media, abandono y total de
00166     # asignaturas superadas.
00167     "asignaturas_superadas_media_abandono": """
00168     SELECT
00169         a.id,
00170         a.abandona,
00171         AVG(NULLIF(la.max_calif, 0)) AS NotaMedia,
00172         COUNT(la.asignatura) AS "Total asignaturas superadas"
00173         FROM
00174             (
00175                 SELECT
00176                     la.id,
00177                     la.asignatura,
00178                     MAX(la.calif_numerica) AS max_calif
00179                 FROM
00180                     public.lineas_actas la
00181                 JOIN
00182                     public.matricula ma
00183                 ON
00184                     la.id = ma.id AND la.cod_plan = ma.cod_plan
00185                 WHERE
00186                     la.curso_aca IN :curso_academico
00187                     AND la.asignatura IN :asignaturas_matriculadas
00188                     AND ma.titulacion = :titulacion
00189                     AND ma.cod_universidad = :cod_universidad
00190                 GROUP BY
00191                     la.id, la.asignatura
00192             ) la
00193             JOIN
00194                 public.alumnos a
00195             ON
00196                 a.id = la.id
00197                 WHERE
00198                     la.max_calif >= 5
00199                 GROUP BY
00200                     a.id, a.abandona
00201                 ORDER BY
00202                     a.id;
00203 """
00203 # Consulta para obtener la calificación cualitativa de los alumnos para compararlo con los datos
00204     generales.

```

```

00204           "calif_cualitativa_comparativa": """
00205 WITH CalificacionesMaximas AS (
00206   SELECT
00207     la.id,
00208     la.asignatura,
00209     la.curso_aca,
00210     la.calif,
00211     ROW_NUMBER() OVER (PARTITION BY la.id, la.curso_aca, la.asignatura ORDER BY CASE
00212       WHEN la.calif = 'Sobresaliente' THEN 5
00213                 WHEN la.calif = 'Notable' THEN 4
00214                 WHEN la.calif = 'Aprobado' THEN 3
00215                 WHEN la.calif = 'Suspens' THEN 2
00216                 WHEN la.calif = 'No presentado' THEN 1
00217               ELSE 0 END DESC) AS rk
00218   FROM lineas_actas la
00219   JOIN
00220     matricula ma ON la.id = ma.id AND la.cod_plan = ma.cod_plan
00221   WHERE
00222     la.curso_aca IN :curso_academico AND
00223     la.asignatura IN :asignaturas_matriculadas AND
00224     la.calif IN ('Sobresaliente', 'Notable', 'Aprobado', 'Suspens', 'No
00225     presentado') AND
00226       ma.titulacion = :titulacion AND
00227       ma.cod_universidad = :cod_universidad
00228   )
00229   SELECT
00230     c.asignatura,
00231     c.calif,
00232     COUNT(*) AS count_grades
00233   FROM CalificacionesMaximas c
00234   WHERE c.rk = 1
00235   GROUP BY c.asignatura, c.calif;
00236   """
00236 # Consulta para obtener la calificación cualitativa del alumno.
00237           "calif_cualitativa_alumno_asignaturas": """
00238 WITH CalificacionesMaximas AS (
00239   SELECT
00240     la.id,
00241     la.asignatura,
00242     MAX(
00243       CASE
00244         WHEN la.calif = 'Sobresaliente' THEN 5
00245                 WHEN la.calif = 'Notable' THEN 4
00246                 WHEN la.calif = 'Aprobado' THEN 3
00247                 WHEN la.calif = 'Suspens' THEN 2
00248                 WHEN la.calif = 'No presentado' THEN 1
00249               ELSE 0
00250             END
00251       ) AS max_calif_value
00252   FROM
00253     public.lineas_actas la
00254   JOIN
00255     public.matricula ma ON la.id = ma.id AND la.cod_plan = ma.cod_plan
00256   WHERE
00257     la.curso_aca IN :curso_academico AND
00258     la.asignatura IN :asignaturas_matriculadas AND
00259     la.id = :alumno_id AND
00260     la.calif IN ('Sobresaliente', 'Notable', 'Aprobado', 'Suspens', 'No
00261     presentado') AND
00262       ma.titulacion = :titulacion
00263     GROUP BY
00264       la.id, la.asignatura
00265   )
00266   SELECT
00267     cm.asignatura,
00268     CASE cm.max_calif_value
00269       WHEN 5 THEN 'Sobresaliente'
00270       WHEN 4 THEN 'Notable'
00271       WHEN 3 THEN 'Aprobado'
00272       WHEN 2 THEN 'Suspens'
00273       WHEN 1 THEN 'No presentado'
00274       ELSE 'Desconocido'
00275     END AS calif,
00276     COUNT(*) AS count_grades
00277   FROM
00278     CalificacionesMaximas cm
00279   GROUP BY
00280     cm.asignatura, cm.max_calif_value
00281   ORDER BY
00282     cm.asignatura;
00283   """
00283 # Consulta para obtener la calificación media de los alumnos y la calificación del alumno.
00284           "nota_media_general_mi_nota": """
00285 SELECT
00286   subquery.asignatura,
00287   AVG(subquery.max_calif_numerica) AS media_calif,
00288   MAX(CASE WHEN subquery.id = :alumno_id THEN subquery.max_calif_numerica ELSE NULL END) AS calif_alumno

```

```

00289 FROM (
00290 SELECT
00291 l.asignatura,
00292 l.id,
00293 l.curso_aca,
00294 MAX(l.calif_numerica) AS max_calif_numerica
00295 FROM
00296 lineas_actas l
00297 JOIN
00298 matricula m ON l.id = m.id AND l.cod_plan = m.cod_plan
00299 WHERE
00300 l.asignatura IN :asignaturas_matriculadas AND
00301 l.curso_aca IN :curso_academico AND
00302 m.titulacion = :titulacion AND
00303 m.cod_universidad = :cod_universidad
00304 GROUP BY
00305 l.asignatura,
00306 l.id,
00307 l.curso_aca
00308 ) subquery
00309 GROUP BY
00310 subquery.asignatura
00311 ORDER BY
00312 subquery.asignatura;
00313 """,
00314 },
00315 },
00316 },
00317 "docente": {
00318     "common": {
00319         "# Consulta para obtener los docentes.
00320         "docentes_all": """
00321 SELECT DISTINCT id_docente FROM docentes
00322 LIMIT 20;
00323 """",
00324 # Consulta para obtener la universidad de un docente.
00325         "universidades_docente": """
00326 SELECT DISTINCT cod_universidad, universidad
00327 FROM docentes
00328 WHERE id_docente = :id_docente;
00329 """,
00330 },
00331 "filters": {
00332         "# Consulta para obtener las asignaturas de un docente según la titulación.
00333         "asignaturas_docente": """
00334 SELECT DISTINCT asignatura
00335 FROM docentes
00336 WHERE id_docente = :id_docente AND titulacion = :titulacion;
00337 """,
00338 # Consulta para obtener los cursos académicos de un docente según la asignatura.
00339         "curso_academico_docente": """
00340 SELECT curso_aca
00341 FROM docentes
00342 WHERE id_docente = :id_docente AND asignatura = :asignatura;
00343 """",
00344 # Consulta para obtener la titulación de un docente.
00345         "titulacion_docente": """
00346 SELECT DISTINCT titulacion
00347 FROM docentes
00348 WHERE id_docente = :id_docente;
00349 """,
00350 # Consulta para obtener todos los cursos académicos de las actas según la titulación
00351         "curso_academico_actas_titulacion": """
00352 SELECT DISTINCT li.curso_aca
00353 FROM lineas_actas li
00354 JOIN docentes ON li.cod_plan = docentes.cod_plan AND
00355 li.cod_asig = docentes.cod_asignatura AND
00356 li.curso_aca = docentes.curso_aca
00357 WHERE docentes.titulacion = :titulacion
00358 ORDER BY curso_aca;
00359 """",
00360 # Consulta para obtener todas las asignaturas por curso académico y titulación
00361         "asignaturas_actas_titulacion": """
00362 SELECT DISTINCT li.asignatura
00363 FROM lineas_actas li
00364 JOIN docentes ON li.cod_plan = docentes.cod_plan
00365 WHERE docentes.titulacion = :titulacion AND
00366 li.curso_aca = :curso_academico AND
00367 li.cod_asig = docentes.cod_asignatura
00368 ORDER BY li.asignatura;
00369
00370 """,
00371 },
00372 "graphs": {
00373     "personal": {
00374         "# Consulta para obtener el número de alumnos repetidores y de nuevo ingreso en una
asignatura.

```

```

00375           "alumnos_repetidores_nuevos": """
00376 WITH repetidores AS (
00377 SELECT
00378 am.id AS student_id,
00379 MIN(am.curso_aca) AS primer_curso_aca
00380 FROM public.asignaturas_matriculadas am
00381 WHERE am.asignatura = :asignaturas
00382 GROUP BY am.id
00383 HAVING COUNT(DISTINCT am.curso_aca) > 1
00384 ),
00385 alumnos_categoria AS (
00386 SELECT
00387 am.id,
00388 am.curso_aca,
00389 CASE
00390 WHEN r.student_id IS NOT NULL AND r.primer_curso_aca <> am.curso_aca THEN 'repetidor'
00391 ELSE 'nuevo_ingreso'
00392 END AS categoria
00393 FROM public.asignaturas_matriculadas am
00394 LEFT JOIN repetidores r ON am.id = r.student_id
00395 WHERE am.asignatura = :asignaturas
00396 AND am.curso_aca IN :curso_academico
00397 AND EXISTS (
00398     SELECT 1
00399     FROM public.docentes d
00400     WHERE d.cod_asignatura = am.cod_asignatura
00401     AND d.id_docente = :docente_id
00402 )
00403 )
00404 SELECT
00405     am.curso_aca AS curso_academico,
00406     COUNT(DISTINCT am.id) FILTER (WHERE categoria = 'repetidor') AS
00407     alumnos_repetidores,
00408     COUNT(DISTINCT am.id) FILTER (WHERE categoria = 'nuevo_ingreso') AS
00409     alumnos_nuevo_ingreso
00410     FROM alumnos_categoria am
00411     GROUP BY am.curso_aca
00412     ORDER BY am.curso_aca;
00413 """
00414 # Consulta para el número de alumnos por género en una asignatura.
00415           "alumnos_genero_docente": """
00416 WITH docente_asignaturas AS (
00417     SELECT cod_asignatura
00418     FROM public.docentes
00419     WHERE id_docente = :id_docente
00420     AND asignatura = :asignaturas
00421 ),
00422 asignaturas_matriculadas AS (
00423     SELECT DISTINCT am.id, am.cod_asignatura, am.curso_aca
00424     FROM public.asignaturas_matriculadas am
00425     JOIN docente_asignaturas da ON am.cod_asignatura = da.cod_asignatura
00426     WHERE am.curso_aca IN :curso_academico
00427     AND am.asignatura = :asignaturas
00428 ),
00429 estudiantes_sexo AS (
00430     SELECT DISTINCT m.id AS alumno_id, m.sexo
00431     FROM public.matricula m
00432     JOIN asignaturas_matriculadas am ON m.id = am.id
00433     SELECT am.curso_aca, m.sexo, COUNT(*) AS cantidad
00434     FROM estudiantes_sexo m
00435     JOIN asignaturas_matriculadas am ON m.alumno_id = am.id
00436     GROUP BY am.curso_aca, m.sexo
00437     ORDER BY am.curso_aca, m.sexo;
00438 """
00439 # Consulta para obtener la nota media de los alumnos por asignatura y curso académico.
00440           "alumnos/nota_media_docente": """
00441 SELECT
00442 l.curso_aca,
00443 l.asignatura,
00444 AVG(l.calif_numerica) AS media_calif
00445 FROM
00446 lineas_actas l
00447 WHERE
00448 l.asignatura IN :asignaturas AND
00449 l.curso_aca IN :curso_academico
00450 GROUP BY
00451 l.curso_aca,
00452 l.asignatura
00453 ORDER BY
00454 l.curso_aca,
00455 l.asignatura;
00456 """
00457 # Consulta para obtener la nota cualitativa de los alumnos por asignatura y curso académico.
00458           "alumnos/nota_cualitativa_docente": """
00459 SELECT DISTINCT

```

```

00460 l.curso_aca,
00461 l.calif,
00462 COUNT(*) AS num_alumnos
00463 FROM
00464 lineas_actas l
00465 WHERE
00466 l.asignatura = :asignaturas AND
00467 l.curso_aca IN :curso_academico
00468 GROUP BY
00469 l.curso_aca,
00470 l.calif
00471 ORDER BY
00472 l.curso_aca,
00473 l.calif;
00474 """",
00475 },
00476 "general": {
00477         # Consulta para obtener el número de calificaciones cualitativas
00478         # de los alumnos por asignatura y curso académico.
00479         "calif_all_cualitativa_asignaturas": """
00480 SELECT
00481 li.curso_aca,
00482 li.asignatura,
00483 li.calif,
00484 COUNT(DISTINCT li.id) AS n_alumnos
00485 FROM lineas_actas li
00486 JOIN docentes ON li.cod_plan = docentes.cod_plan AND li.cod_asig = docentes.cod_asignatura
00487 WHERE
00488 docentes.titulacion = :titulacion AND
00489 li.curso_aca = :curso_academico AND
00490 li.asignatura IN :asignaturas
00491 GROUP BY li.asignatura, li.curso_aca, li.calif
00492 ORDER BY li.asignatura, li.curso_aca, li.calif;
00493 """
00494 """,
00495 "calif_media_asignaturas": """
00496 SELECT
00497 subquery.asignatura,
00498 AVG(subquery.calif_numerica) AS media_calif
00499 FROM (
00500 SELECT
00501 l.asignatura,
00502 l.id,
00503 l.curso_aca,
00504 l.calif_numerica
00505 FROM
00506 lineas_actas l
00507 JOIN
00508 matricula m ON l.id = m.id AND l.cod_plan = m.cod_plan
00509 WHERE
00510 l.asignatura IN :asignaturas AND
00511 l.curso_aca = :curso_academico AND
00512 m.titulacion = :titulacion
00513 GROUP BY
00514 l.asignatura,
00515 l.id,
00516 l.curso_aca,
00517 l.calif_numerica
00518 ) subquery
00519 GROUP BY
00520 subquery.asignatura
00521 ORDER BY
00522 subquery.asignatura;
00523 """",
00524 },
00525 },
00526 },
00527 "gestor": {
00528         "common": {
00529             # Consulta para obtener los gestores.
00530             "gestores_all": """
00531 SELECT DISTINCT gestor_id FROM gestores
00532 LIMIT 20;
00533 """",
00534 # Consulta para obtener el número de alumnos matriculados en una universidad.
00535         "numero_alumnos_matriculados_universidad": """
00536 SELECT COUNT(alumnos.id) AS n_alumnos_universidad
00537 FROM alumnos
00538 WHERE universidad = :universidad AND abandona = 'no';
00539 """
00540 # Consulta que muestra la universidad de un gestor.
00541         "universidades_gestor": """
00542 SELECT DISTINCT cod_universidad, universidad
00543 FROM gestores
00544 WHERE gestor_id = :gestor_id;
00545 """",
00546 "cursos_academicos_de_egresados": """

```

```

00547 SELECT DISTINCT curso_aca
00548 FROM egresados
00549 WHERE cod_universidad = :cod_universidad
00550 """",
00551 },
00552 "filters": {
00553     # Consulta para obtener los cursos académicos de una universidad.
00554     "curso_academico_universidad": """
00555 SELECT DISTINCT curso_aca
00556 FROM matricula
00557 WHERE cod_universidad = :cod_universidad
00558 ORDER BY curso_aca;
00559 """",
00560 # Consulta para obtener las titulaciones de una universidad.
00561     "titulaciones_universidad_gestor": """
00562 SELECT DISTINCT titulacion
00563 FROM matricula
00564 WHERE cod_universidad = :cod_universidad AND curso_aca = :curso_academico;
00565 """",
00566 },
00567 "graphs": {
00568     "indicadores": {
00569         # Consulta para obtener el número de alumnos de nuevo ingreso por género y titulación.
00570         "alumnos_nuevo_ingreso_genero_titulacion": """
00571 SELECT curso_aca, titulacion, sexo, COUNT(*) as num_alumnos
00572     FROM matricula
00573     WHERE curso_aca = :curso_academico
00574     AND titulacion IN :titulaciones
00575     AND nuevo_ingreso = 'si'
00576     AND cod_universidad = :cod_universidad
00577     GROUP BY curso_aca, titulacion, sexo
00578     ORDER BY titulacion;
00579 """
00580 # Consulta para obtener el número de alumnos egresados por género y titulación.
00581     "alumnos_egresados_genero_titulacion": """
00582 SELECT
00583 m.titulacion AS Titulacion,
00584 m.sexo AS Genero,
00585 COUNT(DISTINCT m.id) AS Cantidad
00586 FROM public.egresados e
00587 JOIN public.matricula m ON e.cod_plan = m.cod_plan AND
00588 e.curso_aca = m.curso_aca AND e.id = m.id
00589 WHERE e.cod_universidad = :cod_universidad AND
00590 m.curso_aca = :curso_academico AND
00591 m.titulacion IN :titulaciones
00592 GROUP BY m.titulacion, m.sexo
00593 ORDER BY m.titulacion, m.sexo;
00594 """",
00595 # Consulta para obtener el número de alumnos de nuevo ingreso por nacionalidad y titulación.
00596     "alumnos_nuevo_ingreso_nacionalidad_titulacion": """
00597 SELECT titulacion, nacionalidad, COUNT(*) as num_alumnos
00598     FROM matricula
00599     WHERE curso_aca = :curso_academico
00600     AND titulacion IN :titulaciones
00601     AND nuevo_ingreso = 'si'
00602     AND cod_universidad = :cod_universidad
00603     GROUP BY titulacion, nacionalidad
00604 """
00605 # Consulta para obtener el número de alumnos egresados por nacionalidad y titulación.
00606     "alumnos_egresados_nacionalidad_titulacion": """
00607 SELECT
00608 m.titulacion AS Titulacion,
00609 m.nacionalidad,
00610 COUNT(DISTINCT m.id) AS Cantidad
00611 FROM public.egresados e
00612 JOIN public.matricula m ON e.cod_plan = m.cod_plan AND
00613 e.curso_aca = m.curso_aca AND e.id = m.id
00614 WHERE e.cod_universidad = :cod_universidad AND
00615 m.curso_aca = :curso_academico AND
00616 m.titulacion IN :titulaciones
00617 GROUP BY m.titulacion, m.nacionalidad
00618 ORDER BY m.titulacion, m.nacionalidad;
00619 """",
00620 },
00621 "resultados": {
00622     # Consulta para obtener la nota media de acceso de cada titulación.
00623     "nota_media_acceso_titulacion": """
00624 SELECT ma.curso_aca, ma.titulacion,
00625 AVG(eb.nota_prue) AS nota_ebau_media
00626 FROM matricula ma
00627 JOIN ebau_prueba eb ON ma.id = eb.id
00628 WHERE nuevo_ingreso = 'si' AND ma.cod_universidad = :cod_universidad
00629     GROUP BY ma.curso_aca, ma.titulacion
00630     ORDER BY ma.curso_aca, ma.titulacion;
00631 """
00632 # Consulta para obtener la duración media de los estudios con respecto a la nota media, por titulación
y curso académico.

```

```

00633             "duracion_media_estudios_nota_gestor": """
00634     WITH matriculas_por_estudiante AS (
00635         SELECT
00636             id,
00637             titulacion,
00638             COUNT(*) AS numero_matriculas
00639         FROM matricula
00640         GROUP BY id, titulacion
00641     ),
00642     media_notas_y_duracion AS (
00643         SELECT
00644             e.curso_aca,
00645             m.titulacion,
00646             m.rama,
00647             AVG(e.nota_media) AS nota_media_curso_aca,
00648             ROUND(AVG(mpe.numero_matriculas)) AS duracion_media_estudios,
00649             COUNT(DISTINCT e.id) AS numero_alumnos
00650         FROM egresados e
00651         JOIN matricula m ON e.id = m.id AND e.cod_plan = m.cod_plan AND e.cod_universidad = m.cod_universidad
00652         JOIN matriculas_por_estudiante mpe ON mpe.id = m.id AND mpe.titulacion = m.titulacion
00653         WHERE e.cod_universidad = :cod_universidad
00654         GROUP BY e.curso_aca, m.titulacion, m.rama
00655     )
00656     SELECT
00657         nota_media_curso_aca,
00658         titulacion,
00659         rama,
00660         curso_aca,
00661         duracion_media_estudios,
00662         numero_alumnos
00663     FROM media_notas_y_duracion
00664     ORDER BY curso_aca;
00665
00666     """,
00667 },
00668     "riesgo_abandono": {
00669         "# Consulta para obtener la tasa de abandono de una titulación.
00670         "tasa_abandono_titulacion_gestor": """
00671     WITH ultima_matricula AS (
00672         SELECT
00673             m.id,
00674             m.curso_aca,
00675             m.titulacion,
00676             t.abandona,
00677             ROW_NUMBER() OVER (PARTITION BY m.id ORDER BY m.curso_aca DESC) AS row_num
00678         FROM
00679             public.matricula m
00680         JOIN
00681             public.alumnos t ON m.id = t.id
00682     )
00683
00684     SELECT
00685         m.curso_aca,
00686         m.titulacion,
00687         COUNT(DISTINCT m.id) AS numero_matriculados,
00688         COUNT(DISTINCT CASE WHEN le.abandona = 'si' AND le.row_num = 1 THEN le.id END) AS numero_abandono
00689         FROM
00690             public.matricula m
00691             LEFT JOIN ultima_matricula le ON m.id = le.id AND m.curso_aca =
00692                 le.curso_aca
00693             WHERE
00694                 m.cod_universidad = :cod_universidad
00695                 AND m.curso_aca IN :curso_academico
00696             GROUP BY
00697                 m.curso_aca,
00698                 m.titulacion
00699             ORDER BY
00700                 m.curso_aca,
00701                 m.titulacion;
00702
00703     # Consulta para obtener la tasa de graduación de una titulación.
00704     "tasa_graduacion_titulacion_gestor": """
00705     SELECT
00706         COUNT(matricula.id) AS numero_matriculados,
00707         COALESCE(COUNT( DISTINCT egresados.id), 0) AS numero_egresados,
00708         egresados.curso_aca,
00709         matricula.titulacion
00710     FROM
00711         matricula
00712     LEFT JOIN
00713         egresados ON matricula.id = egresados.id
00714     WHERE
00715         matricula.cod_universidad = :cod_universidad AND
00716         matricula.curso_aca IN :curso_academico
00717     GROUP BY
00718         egresados.curso_aca,

```

```

00719 matricula.titulacion
00720 ORDER BY
00721 titulacion,
00722 curso_aca;
00723 """ ,
00724 },
00725 },
00726 },
00727 }

```

6.247. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/layouts/alumno_layout.py**

Namespaces

- namespace [layouts](#)
- namespace [layouts.alumno_layout](#)

Funciones

- def [layouts.alumno_layout.alumno_layout\(\)](#)

6.248. alumno_layout.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file alumno_layout.py
00003 # @brief Este archivo contiene el código del layout del perfil "Alumno".
00004 # @version 1.0
00005 # @date 01/05/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 import components.alumnado.utils.tabs_alumnado as tabs_alumnado
00013
00014
00015 def alumno_layout():
00016     #
00017     # @brief Retorna el layout del perfil "Alumno"
00018     # @return Layout del alumno
00019     #
00020     return html.Div([tabs_alumnado.tabs_alumnado()])

```

6.249. Referencia del Archivo

**C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_←
Universities_TFG/src/layouts/docente_layout.py**

Namespaces

- namespace [layouts](#)
- namespace [layouts.docente_layout](#)

Funciones

- def `layouts.docente_layout.docente_layout()`

6.250. docente_layout.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file docente_layout.py
00003 # @brief Este archivo contiene el código del layout del perfil "Docente".
00004 # @version 1.0
00005 # @date 01/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 import components.docente.utils.tabs_docente as tabs_docente
00013
00014
00015 def docente_layout():
00016     #
00017     # @brief Retorna el layout del perfil "Docente"
00018     # @return Layout del docente
00019     #
00020     return html.Div([tabs_docente.tabs_docente()])
```

6.251. Referencia del Archivo

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/layouts/gestor_layout.py

Namespaces

- namespace `layouts`
- namespace `layouts.gestor_layout`

Funciones

- def `layouts.gestor_layout.gestor_layout()`

6.252. gestor_layout.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file gestor_layout.py
00003 # @brief Este archivo contiene el código del layout del perfil "Gestor".
00004 # @version 1.0
00005 # @date 01/05/2024
00006 # @license MIT License
00007 # @author Fabrizzio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from dash import html
00012 import components.gestor.utils.tabs_gestor as tabs_gestor
00013
00014
00015 def gestor_layout():
00016     #
00017     # @brief Retorna el layout del perfil "Gestor"
00018     # @return Layout del gestor
00019     #
00020     return html.Div([tabs_gestor.tabs_gestor()])
```

6.253. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/model.py

Namespaces

- namespace [model](#)

Variables

- [model.data](#) = load_data_for_model()
- [model.current_year](#) = datetime.now().year
- list [model.columnas_categoricas](#) = ['nacionalidad', 'sexo', 'titulacion']
- list [model.columnas_numericas](#) = ['nota_def_acceso', 'nota_media', 'edad_actual']
- [model.preprocessor](#)
- [model.X](#) = data[columnas_categoricas + columnas_numericas]
- [model.y](#) = data['abandona']
- [model.ids](#) = data['id']
- [model.X_train](#) = preprocessor.fit_transform(X_train)
- [model.X_test](#) = preprocessor.transform(X_test)
- [model.y_train](#)
- [model.y_test](#)
- [model.id_train](#)
- [model.id_test](#)
- [model.test_size](#)
- [model.random_state](#)
- [model.stratify](#)
- [model.smote](#) = SMOTE(random_state=42)
- [model.poly](#) = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
- dictionary [model.modelos](#)
- dictionary [model.param_grids](#)
- dictionary [model.best_estimators](#) = {}
- dictionary [model.metrics](#) = {}
- [model.total](#)
- [model.desc](#)
- [model.grid_search](#) = GridSearchCV(model, param_grids[key], cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)
- [model.y_pred](#) = grid_search.predict(X_test)
- [model.mejores_modelos](#) = sorted(metrics.items(), key=lambda item: item[1]['f1_score'], reverse=True)[:3]
- list [model.mejores_estimadores](#) = [(key, best_estimators[key]) for key, _ in mejores_modelos]
- [model.voting_clf](#)
- string [model.joblib_file](#) = "src/trained_model.pkl"
- [model.y_pred_voting](#) = voting_clf.predict(X_test)
- [model.cross_val_scores_voting](#) = cross_val_score(voting_clf, poly.transform(preprocessor.transform(X)), y, cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)

6.254. model.py

[Ir a la documentación de este archivo.](#)

```
00001 #
00002 # @file model.py
00003 # @brief Este archivo contiene el código del modelo de aprendizaje automático.
00004 # @version 1.0
00005 # @date 13/06/2024
00006 # @license MIT License
00007 # @author Fabrizio Daniell Perilli Martín
00008 # @email alu0101138589@ull.edu.es
00009 #
00010
00011 from datetime import datetime
00012 from sklearn.impute import SimpleImputer
00013 from sklearn.preprocessing import StandardScaler, OneHotEncoder, PolynomialFeatures
00014 from sklearn.compose import ColumnTransformer
00015 from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold, cross_val_score
00016 from sklearn.ensemble import RandomForestClassifier, VotingClassifier, AdaBoostClassifier
00017 from sklearn.linear_model import LogisticRegression
00018 from sklearn.neighbors import KNeighborsClassifier
00019 from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, f1_score,
roc_auc_score
00020 from sklearn.pipeline import Pipeline
00021 import joblib
00022 from tqdm import tqdm
00023 from imblearn.over_sampling import SMOTE
00024 import xgboost as xgb
00025 import lightgbm as lgb
00026 from util import load_data_for_model
00027
00028 data = load_data_for_model()
00029
00030
00031 current_year = datetime.now().year
00032 data['edad_actual'] = current_year - data['anio_nac']
00033
00034 # Codificar variable objetivo
00035 data['abandona'] = data['abandona'].map({'si': 1, 'no': 0})
00036
00037
00038 columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
00039 columnas_numericas = ['nota_def_acceso', 'nota_media', 'edad_actual']
00040
00041 # Preprocesamiento
00042 preprocessor = ColumnTransformer(
00043     transformers=[
00044         ('num', Pipeline([('imputer', SimpleImputer(strategy='mean')), ('scaler', StandardScaler())]),
00045         ('cat', Pipeline([('imputer', SimpleImputer(strategy='constant', fill_value='Desconocido')),
00046                         ('encoder', OneHotEncoder(handle_unknown='ignore'))]), columnas_categoricas)
00047     ])
00048
00049 # Dividir datos en características y variable objetivo
00050 X = data[columnas_categoricas + columnas_numericas]
00051 y = data['abandona']
00052 ids = data['id']
00053
00054 # Dividir los datos en conjuntos de entrenamiento y prueba
00055 X_train, X_test, y_train, y_test, id_train, id_test = train_test_split(X, y, ids, test_size=0.2,
random_state=42, stratify=y)
00056
00057
00058 X_train = preprocessor.fit_transform(X_train)
00059 X_test = preprocessor.transform(X_test)
00060
00061 smote = SMOTE(random_state=42)
00062 X_train, y_train = smote.fit_resample(X_train, y_train)
00063
00064 # Crear interacciones polinómicas
00065 poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
00066 X_train = poly.fit_transform(X_train)
00067 X_test = poly.transform(X_test)
00068
00069 # Pipelines para cada modelo
00070 modelos = {
00071     'RandomForest': RandomForestClassifier(random_state=42, n_jobs=-1),
00072     'LogisticRegression': LogisticRegression(max_iter=1000, random_state=42),
00073     'KNeighbors': KNeighborsClassifier(n_jobs=-1),
00074     'AdaBoost': AdaBoostClassifier(random_state=42),
00075     'XGBoost': xgb.XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss'),
00076     'LightGBM': lgb.LGBMClassifier(random_state=42, verbose=-1, force_row_wise=True)
00077 }
00078
00079 # Hiperparámetros para búsqueda
```

```

00080 param_grids = {
00081     'RandomForest': {
00082         'n_estimators': [100, 200],
00083         'max_depth': [10, 20],
00084         'min_samples_split': [2, 5],
00085         'min_samples_leaf': [1, 2]
00086     },
00087     'LogisticRegression': {
00088         'C': [0.01, 0.1, 1.0],
00089         'solver': ['liblinear', 'lbfgs']
00090     },
00091     'KNeighbors': {
00092         'n_neighbors': [3, 5, 7],
00093         'weights': ['uniform', 'distance']
00094     },
00095     'AdaBoost': {
00096         'n_estimators': [50, 100],
00097         'learning_rate': [0.01, 0.1]
00098     },
00099     'XGBoost': {
00100         'n_estimators': [100, 200],
00101         'learning_rate': [0.01, 0.1],
00102         'max_depth': [3, 5]
00103     },
00104     'LightGBM': {
00105         'n_estimators': [100, 200],
00106         'learning_rate': [0.01, 0.1],
00107         'max_depth': [3, 5]
00108     }
00109 }
00110
00111 # Buscar los mejores hiperparámetros y entrenar modelos
00112 best_estimators = {}
00113 metrics = {}
00114
00115 with tqdm(total=len(modelos), desc="Entrenando modelos") as pbar:
00116     for key, model in modelos.items():
00117         grid_search = GridSearchCV(model, param_grids[key], cv=StratifiedKFold(n_splits=5),
00118             scoring='accuracy', n_jobs=-1)
00119         grid_search.fit(X_train, y_train)
00120         best_estimators[key] = grid_search.best_estimator_
00121
00122         # Evaluar modelo
00123         y_pred = grid_search.predict(X_test)
00124         metrics[key] = {
00125             'accuracy': accuracy_score(y_test, y_pred),
00126             'precision': precision_score(y_test, y_pred),
00127             'recall': recall_score(y_test, y_pred),
00128             'f1_score': f1_score(y_test, y_pred),
00129             'roc_auc': roc_auc_score(y_test, y_pred)
00130         }
00131         print("")
00132         print(f"Resultados para {key}:")
00133         print(f"Accuracy: {metrics[key]['accuracy']:.2f}")
00134         print(f"Precision: {metrics[key]['precision']:.2f}")
00135         print(f"Recall: {metrics[key]['recall']:.2f}")
00136         print(f"F1 Score: {metrics[key]['f1_score']:.2f}")
00137         print(f"ROC AUC: {metrics[key]['roc_auc']:.2f}")
00138         print("Matriz de Confusión:")
00139         print(confusion_matrix(y_test, y_pred))
00140         print("")
00141         pbar.update(1)
00142
00143
00144 # Selecciona los mejores modelos para el ensemble
00145 mejores_modelos = sorted(metrics.items(), key=lambda item: item[1]['f1_score'], reverse=True)[:3]
00146 mejores_estimadores = [(key, best_estimators[key]) for key, _ in mejores_modelos]
00147
00148 # Crear VotingClassifier con los mejores modelos
00149 voting_clf = VotingClassifier(
00150     estimators=mejores_estimadores,
00151     voting='soft',
00152     n_jobs=-1
00153 )
00154
00155
00156 voting_clf.fit(X_train, y_train)
00157
00158 # Guardar el modelo entrenado
00159 joblib_file = "src/trained_model.pkl"
00160 joblib.dump(voting_clf, joblib_file)
00161 print(f"Modelo guardado en {joblib_file}")
00162
00163 # Métricas del ensemble
00164 y_pred_voting = voting_clf.predict(X_test)
00165 print("Resultados del VotingClassifier:")

```

```

00166 print(f"Modelos utilizados: {[name for name, _ in mejores_estimadores]}")
00167 print(f"Accuracy: {accuracy_score(y_test, y_pred_voting):.2f}")
00168 print(f"Precision: {precision_score(y_test, y_pred_voting):.2f}")
00169 print(f"Recall: {recall_score(y_test, y_pred_voting):.2f}")
00170 print(f"F1 Score: {f1_score(y_test, y_pred_voting):.2f}")
00171 print(f"ROC AUC: {roc_auc_score(y_test, y_pred_voting):.2f}")
00172 print("Matriz de Confusión:")
00173 print(confusion_matrix(y_test, y_pred_voting))
00174
00175 # Validación cruzada del ensemble
00176 cross_val_scores_voting = cross_val_score(voting_clf, poly.transform(preprocessor.transform(X)), y,
cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)
00177 print(f"Cross-Validation Accuracy: {cross_val_scores_voting.mean():.2f}")

```

6.255. Referencia del Archivo C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_TFG/src/util.py

Namespaces

- namespace [util](#)

Funciones

- def [util.list_to_tuple](#) (value)
- def [util.random_color](#) (size)
- def [util.load_data_for_model](#) ()
- def [util.load_model](#) ()

Variables

- dictionary [util.config_mode_bar_buttons_gestor](#)

6.256. util.py

[Ir a la documentación de este archivo.](#)

```

00001 #
00002 # @file util.py
00003 # @brief Este archivo contiene funciones de utilidad para la aplicación.
00004 # @details Se definen funciones para convertir listas en tuplas, generar colores aleatorios y cargar
    datos y modelos.
00005 # @version 1.0
00006 # @date 13/06/2024
00007 # @license MIT License
00008 # @author Fabrizio Daniell Perilli Martin
00009 # @email alu0101138589@ull.edu.es
00010 #
00011
00012 import random
00013 import pandas as pd
00014 import joblib
00015 from datetime import datetime
00016 from sklearn.impute import SimpleImputer
00017 from sklearn.preprocessing import StandardScaler, OneHotEncoder, PolynomialFeatures
00018 from sklearn.compose import ColumnTransformer
00019 from sklearn.pipeline import Pipeline
00020 from data.queries import data_for_model
00021
00022 def list_to_tuple(value):
00023     """
00024     Convierte una lista en una tupla.
00025     Args:
00026         value: Lista a convertir.
00027     Returns:

```

```

00028 Tupla con los elementos de la lista.
00029 """
00030 if isinstance(value, str):
00031     return (value,)
00032 elif isinstance(value, list):
00033     return tuple(value)
00034 else:
00035     return ValueError(
00036         "Tipo de dato no soportado, se esperaba un list o str. {}".format(
00037             type(value)
00038         )
00039     )
00040
00041
00042 def random_color(size):
00043     """
00044 Genera una lista de colores aleatorios en formato hexadecimal.
00045
00046 Args:
00047 size: Número de colores a generar.
00048 Returns:
00049 Lista de colores aleatorios.
00050 """
00051 return ["#%06X" % random.randint(0, 0xFFFFFF) for _ in range(size)]
00052
00053
00054 # Configuración de los botones de la barra de herramientas de Plotly
00055 config_mode_bar_buttons_gestor = {
00056     "displayModeBar": True,
00057     "displaylogo": False,
00058     "scrollZoom": True,
00059     "modeBarButtonsToRemove": ["zoom2d", "lasso2d", "resetScale2d"],
00060     "modeBarButtonsToAdd": [
00061         "drawline",
00062         "drawcircle",
00063         "drawrect",
00064         "eraseshape",
00065         "toggleSpikelines",
00066     ],
00067 }
00068
00069
00070 def load_data_for_model():
00071     """
00072 Carga los datos necesarios para entrenar el modelo.
00073
00074 Returns:
00075 pd.DataFrame: Datos para entrenar el modelo.
00076 """
00077 data = data_for_model()
00078 df = pd.DataFrame(data)
00079
00080 # Ajustar los tipos de datos de las columnas
00081 df = df.astype({
00082     'id': str,
00083     'anio_nac': int,
00084     'nacionalidad': str,
00085     'sexo': str,
00086     'titulacion': str,
00087     'nota_def_acceso': float,
00088     'nota_media': float,
00089     'abandona': str
00090 })
00091
00092     return df
00093
00094
00095 def load_model():
00096     """
00097 Carga el modelo entrenado.
00098
00099 Returns:
00100 Modelo entrenado.
00101 """
00102 model_file = "src/trained_model.pkl"
00103 voting_clf = joblib.load(model_file)
00104 data = load_data_for_model()
00105
00106 current_year = datetime.now().year
00107 data['edad_actual'] = current_year - data['anio_nac']
00108
00109 # Codificar variable objetivo
00110 data['abandona'] = data['abandona'].map({'si': 1, 'no': 0})
00111
00112 # Definir columnas
00113 columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
00114 columnas_numericas = ['nota_def_acceso', 'nota_media', 'edad_actual']

```

```
00115 # Imputación de datos
00116 num_imputer = SimpleImputer(strategy='mean')
00117 cat_imputer = SimpleImputer(strategy='constant', fill_value='Desconocido')
00118
00119 # Preprocesamiento
00120 preprocessor = ColumnTransformer(
00121     transformers=[
00122         ('num', Pipeline([('imputer', num_imputer), ('scaler', StandardScaler())]),
00123          columnas_numericas),
00124         ('cat', Pipeline([('imputer', cat_imputer), ('encoder',
00125            OneHotEncoder(handle_unknown='ignore'))]), columnas_categoricas)
00126     ]
00127 )
00128
00129 # Dividir datos en características y variable objetivo
00130 X = data[columnas_categoricas + columnas_numericas]
00131 y = data['abandona']
00132 ids = data['id']
00133
00134 X_non_abandon = X[y == 0]
00135 data_non_abandon = data[y == 0].copy()
00136 X_non_abandon_preprocessed = preprocessor.fit_transform(X_non_abandon)
00137
00138 poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
00139 X_non_abandon_poly = poly.fit_transform(X_non_abandon_preprocessed)
00140
00141 # Predecir probabilidades de abandono
00142 probabilidad_abandono = voting_clf.predict_proba(X_non_abandon_poly)[:, 1]
00143 data_non_abandon['probabilidad_abandono'] = probabilidad_abandono
00144
00145 return data_non_abandon
```


Índice alfabético

__init__
 data.db_connector.DatabaseConnector, 223
_favicon
 app, 13

alumno_layout
 layouts.alumno_layout, 207

alumnos_all
 data.queries, 169

alumnos_df
 data.generate_synthetic_data, 165

alumnos_egresados_genero_titulacion
 data.queries, 170

alumnos_egresados_nacionalidad_titulacion
 data.queries, 171

alumnos_genero_docente
 data.queries, 172

alumnos_nota_cualitativa_docente
 data.queries, 172

alumnos_nota_media_docente
 data.queries, 173

alumnos_nuevo_ingreso_genero_titulacion
 data.queries, 174

alumnos_nuevo_ingreso_nacionalidad_titulacion
 data.queries, 175

alumnos_repetidores_nuevos
 data.queries, 176

app, 13
 /favicon, 13
 app, 13
 debug, 13
 layout, 14
 server, 14
 title, 14

asignaturas_actas_titulacion
 data.queries, 176

asignaturas_dict
 data.generate_synthetic_data, 165

asignaturas_docente
 data.queries, 177

asignaturas_matriculadas
 data.queries, 178

asignaturas_matriculadas_df
 data.generate_synthetic_data, 165

asignaturas_matriculadas_y_superadas
 data.queries, 178

asignaturas_superadas
 data.queries, 179

asignaturas_superadas_media_abandono

 data.queries, 180

best_estimators
 model, 210

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_227

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_228

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_228

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_228

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_235, 236

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_237

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_238

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_229

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_229

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_239

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_241

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_243

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_229

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_245

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_246, 247

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_248

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_249, 250

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_229

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_251

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_252

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_253

C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BI_Universities_229

- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_dollares/
255
231
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_layout/
255, 256
279
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_layout/B1_universities/
256, 257
282, 283
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/B1_universities/
230
285, 286
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/B1_universities/
230
288, 289
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_tareas/B1_universities/
257, 258
232
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_tareas/B1_universities/
259
291, 292
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_tareas/B1_universities/
260
294, 295
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_tareas/B1_universities/
261
232
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/Bal_tareas/B1_universities/
262
297, 298
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/B1_universities/
230
300, 301
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/general/B1_universities/
230
232
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/general/B1_universities/
263
303
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/general/B1_universities/
265
304, 305
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/personal/B1_universities/
230
305, 306
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/personal/B1_universities/
266
232
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/personal/B1_universities/
267, 268
232
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/personal/B1_universities/
269
233
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/personal/B1_universities/
270, 271
307
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/S_cursos/stadad/6_tutorios/B1_universities/
231
308
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/stadad/6_tutorios/B1_universities/
272
309
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/stadad/6_tutorios/B1_universities/
273
233
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/stadad/6_tutorios/B1_universities/
274
310
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/stadad/6_tutorios/B1_universities/
231
311
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/obligaciones/6_tutoriales/stadad/6_tutorios/B1_universities/
231
233
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/gestion/fitas/datos/B1_universities/
275, 276
312
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/gestion/fitas/datos/B1_universities/
277
313
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/gestion/fitas/datos/B1_universities/
278
314
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_University/Documentos/Fabrics/gestion/graficas/6_tutorios/B1_universities/
231
314, 315

- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universities_233
334
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universities_315, 316
336
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universitys_316
337
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universitys_317
335
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universitys_318
338
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universitys_318, 319
339
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universitys_319, 320
340
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universitys_320
340, 341
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_Universitys_233
341
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universities_234
342
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universities_321
343, 344
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universidades_322
349, 350
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universidades_323
357
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universidades_324
325
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universidades_325
366
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universidades_234
366, 367
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universidades_326
367
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BI_universidades_327
368, 369
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BC_Universitys_TFG/src/components/Facultad/Universidad/6itcurso/BpyUniversities_234
371
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/docente/utils/recomendador_docente_data.queries, 181
328
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/docente/utils/resumen_docente.py, 181
329
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/docente/utils/select_docente.py, 182
329, 330
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/docente/utils/tabs_docente.py, 183
330
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/gestor/_init__.py, 184
234
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/gestor/filters/_init__.py, 185
234
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/gestor/filters/filter_all_curso_academico_185
331
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/gestor/filters/filter_curso_academico_14
332
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/gestor/filters/filter_titulaciones_gestor_callbacks.alumnado, 15
333
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/gestor/graphs/_init__.py, 15
235
- C:/Users/fabri/Documents/Fabrizio/Universidad/6_curso/BtalUniversities_TFG/src/components/gestor/graphs/_init__.py, 15
update_filter_asignaturas_matri_alumnado, 15

callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado
 update_filter_asignaturas_docente, 43
 update_filter_curso_academico_alumnado, 16
 callbacks.alumnado.filters.callback_filter_titulacion_alumnado, 17
 update_filter_titulacion_alumnado, 18
 callbacks.alumnado.graphs, 19
 callbacks.alumnado.graphs.general, 19
 callbacks.alumnado.graphs.general.callback_graph_asig
 19
 update_graph_alumnado, 19
 callbacks.alumnado.graphs.general.callback_graph_calif_cual
 21
 update_graph_alumnado, 21
 callbacks.alumnado.graphs.general.callback_graph_calif_media
 23
 update_graph_alumnado, 23
 callbacks.alumnado.graphs.personal, 25
 callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa
 25
 update_graph_alumnado, 25
 callbacks.alumnado.graphs.personal.callback_graph_calif_numerica
 27
 update_graph_alumnado, 27
 callbacks.alumnado.graphs.personal.callback_graph_progreso
 29
 update_graph_alumnado, 29
 callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento
 30
 update_graph_alumnado, 30
 callbacks.alumnado.utils, 31
 callbacks.alumnado.utils.callback_resumen_alumnado,
 32
 not_data, 32
 update_resumen_alumnado, 32
 callbacks.alumnado.utils.callback_select_alumnado, 34
 store_selected_alumnado, 34
 callbacks.alumnado.utils.callback_tabs_alumnado, 35
 render_content, 35
 callbacks.common, 36
 callbacks.common.callback_sidebarCollapse, 36
 sidebarCollapse, 37
 callbacks.common.callback_update_layout, 37
 update_layout, 37
 callbacks.common.callback_user_role, 38
 initialize_dropdown, 39
 preventInitialCall, 40
 updateRole, 39
 callbacks.docente, 40
 callbacks.docente.filters, 40
 callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente,
 40
 updateFilterAsignaturasDocente, 40
 callbacks.docente.filters.callback_filter_all_curso_academico,
 42
 updateFilterAllCursosAcademicosDocente, 42
 callbacks.docente.filters.callback_filter_asignaturas_docente
 43
 updateFilterAsignaturasDocente, 43
 updateFilterTitulacionDocente, 44
 callbacks.docente.filters.callback_filter_titulacion_docente,
 45
 updateFilterTitulacionDocente, 45
 callbacks.docente.graphs, 46
 callbacks.docente.graphs.general, 46
 callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente
 46
 updateGraphDocente, 46
 callbacks.docente.graphs.general.callback_graph_all_calif_media_docente
 48
 updateGraphDocente, 48
 callbacks.docente.graphs.personal, 49
 callbacks.docente.graphs.personal.callback_graph_alu_genero_docente,
 50
 updateGraphAluDocente, 50
 callbacks.docente.graphs.personal.callback_graph_alu_media_docente,
 51
 updateGraphAluDocente, 51
 callbacks.docente.graphs.personal.callback_graph_alu_notas_cualitativas
 52
 updateGraphAluDocente, 52
 callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos
 54
 updateGraphAluDocente, 54
 callbacks.docente.utils, 56
 callbacks.docente.utils.callback_resumen_docente, 56
 notData, 56
 updateResumenDocente, 56
 callbacks.docente.utils.callback_select_docente, 57
 storeSelectedDocente, 58
 callbacks.docente.utils.callback_tabs_docente, 58
 renderContent, 59
 callbacks.gestor, 60
 callbacks.gestor.filters, 60
 callbacks.gestor.filters.callback_filter_all_curso_academico_gestor,
 61
 updateFilterAllCursoAcademicoGestor, 61
 callbacks.gestor.filters.callback_filter_curso_academico_gestor,
 62
 updateFilterCursoAcademicoGestor, 62
 callbacks.gestor.filters.callback_filter_titulaciones_gestor,
 63
 updateFilterTitulacionesGestor, 63
 callbacks.gestor.graphs, 65
 callbacks.gestor.graphs.indicadores, 65
 callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_g
 66
 generateCsv, 65
 getData, 66
 toggleModal, 67
 updateGraphGestor, 68
 updateTable, 69
 callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad
 71

generate_csv, 71
get_data, 72
toggle_modal, 73
update_graph_gestor, 74
update_table, 75
callbacks.gestor.graphs.indicadores.callback_graph_nuevocod_asignaturas_gestor,
76
generate_csv, 76
get_data, 77
toggle_modal, 78
update_graph_gestor, 79
update_table, 81
callbacks.gestor.graphs.indicadores.callback_graph_nuevocomponentes_academico_gestor,
82
generate_csv, 82
get_data, 83
toggle_modal, 84
update_graph_gestor, 85
update_table, 86
callbacks.gestor.graphs.resultados, 87
callbacks.gestor.graphs.resultados.callback_graph_duracion_estudios,
88
generate_csv, 88
get_data, 89
toggle_modal, 90
update_graph_gestor, 91
update_table, 92
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor,
93
generate_csv, 93
get_data, 94
toggle_modal, 95
update_grph_gestor, 96
update_table, 97
callbacks.gestor.graphs.riesgo_abandono, 98
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor,
98
generate_csv, 98
get_data, 99
toggle_modal, 101
update_graph_gestor, 101
update_table, 103
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor,
104
generate_csv, 104
get_data, 105
toggle_modal, 106
update_graph_gestor, 107
update_table, 108
callbacks.gestor.utils, 109
callbacks.gestor.utils.callback_resumen_gestor, 109
not_data, 109
update_resumen_gestor, 110
callbacks.gestor.utils.callback_select_gestor, 111
store_selected_gestor, 111
callbacks.gestor.utils.callback_tabs_gestor, 112
render_content, 112
check_data
data.queries, 186
close
data.db_connector.DatabaseConnector, 224
cod_asignaturas_dict
data.generate_synthetic_data, 165
columnas_categoricas
model, 210
columnas_numericas
model, 210
components, 114
components.alumnado, 164
components.alumnado.filters, 114
components.alumnado.filters.filter_asignaturas_matri_alumnado,
115
filter_asignaturas_matri_alumnado, 115
components.alumnado.filters.filter_curso_academico_alumnado,
116
filter_curso_academico_alumnado, 116
components.alumnado.estudios, 117
components.alumnado.estudios.admitidos, 117
filter_titulacion_alumnado, 117
components.alumnado.graphs, 118
components.alumnado.graphs.graphs_general_alumnado,
118
graphs_general_alumnado, 118
components.alumnado.graphs.personal_alumnado,
119
graphs_personal_alumnado, 119
components.alumnado.utils, 120
components.alumnado.utils.recomendador_alumnado,
120
recomendador_alumnado, 120
components.alumnado.utils.resumen_alumnado, 122
components.alumnado.utils.resumen_alumnado, 122
components.alumnado.utils.select_alumnado, 122
select_alumnado, 122
components.alumnado.utils.tabs_alumnado, 123
tabs_alumnado, 123
components.common, 124
components.common.create_graph, 124
components.common.create_graph, 125
components.common.create_graph_with_table, 125
create_graph_with_table, 126
components.common.filters, 127
filters, 127
components.common.footer, 128
footer, 128
components.common.header, 128
header, 129
components.common.modal_data, 129
create_modal, 129
components.common.sidebar, 130
sidebar, 131
components.docente, 131
components.docente.filters, 132

components.docente.filters.filter_all_asignaturas_titulacion
 132
 filter_all_asignaturas_titulacion_docente, 132
 components.docente.filters.filter_all_curso_academico,
 133
 filter_all_curso_academico, 133
 components.docente.filters.filter_asignaturas_docente,
 134
 filter_asignaturas_docente, 134
 components.docente.filters.filter_curso_academico_docente,
 135
 filter_curso_academico_docente, 135
 components.docente.filters.filter_titulacion_docente,
 136
 filter_titulacion_docente, 136
 components.docente.graphs, 137
 components.docente.graphs.graphs_general_docente,
 137
 graphs_general_docente, 137
 components.docente.graphs.graphs_personal_docente,
 138
 graphs_personal_docente, 139
 components.docente.utils, 140
 components.docente.utils.recomendador_docente, 140
 recomendador_docente, 140
 components.docente.utils.resumen_docente, 141
 resumen_docente, 141
 components.docente.utils.select_docente, 142
 select_docente, 142
 components.docente.utils.tabs_docente, 143
 tabs_docente, 143
 components.gestor, 143
 components.gestor.filters, 144
 components.gestor.filters.filter_all_curso_academico_gestor,
 144
 filter_all_curso_academico_gestor, 144
 components.gestor.filters.filter_curso_academico_gestor,
 145
 filter_curso_academico_gestor, 145
 components.gestor.filters.filter_titulaciones_gestor, 146
 filter_titulaciones_gestor, 146
 components.gestor.graphs, 147
 components.gestor.graphs.graphs_indicadores_gestor,
 147
 graphs_indicadores_gestor, 147
 components.gestor.graphs.graphs_resultados_gestor,
 149
 graphs_resultados_gestor, 149
 components.gestor.graphs.graphs_riesgo_abandono_gestor,
 150
 graphs_riesgo_abandono_gestor, 150
 components.gestor.utils, 152
 components.gestor.utils.recomendador_gestor, 152
 recomendador_gestor, 152
 components.gestor.utils.resumen_gestor, 153
 resumen_gestor, 153
 components.gestor.utils.select_gestor, 154
 select_gestor, 154
 components.gestor.utils.tabs_gestor, 155
 tabs_gestor, 155
 config
 data.db_connector, 156
 config_mode_bar_buttons_gestor
 util, 221
 create_graph
 components.common.create_graph, 125
 create_graph_with_table
 components.common.create_graph_with_table,
 126
 create_modal
 components.common.modal_data, 129
 cross_val_scores_voting
 model, 210
 current_year
 model, 210
 curso_academico_actas_titulacion
 data.queries, 188
 curso_academico_alumnado
 data.queries, 189
 curso_academico_docente
 data.queries, 190
 curso_academico_universidad
 data.queries, 190
 cursos_academicos
 data.generate_synthetic_data, 166
 cursos_academicos_egresados
 data.queries, 191
 data, 156
 model, 211
 data.db_connector, 156
 config, 156
 db, 157
 data.db_connector.DatabaseConnector, 223
 __init__, 223
 close, 224
 db_url, 225
 engine, 225
 execute_query, 224
 data.generate_synthetic_data, 157
 alumnos_df, 165
 asignaturas_dict, 165
 asignaturas_matriculadas_df, 165
 cod_asignaturas_dict, 165
 cod_plan_dict, 166
 cursos_academicos, 166
 docentes_df, 166
 ebau_prueba_df, 166
 educacion, 166
 egresados_df, 166
 fake, 167
 generate_alumnos, 158
 generate_asignaturas, 158
 generate_asignaturas_matriculadas, 159
 generate_curso_aca, 159
 generate_docentes, 160
 generate_ebau_prueba, 161

generate_egresados, 161
generate_gestores, 162
generate_lineas_actas, 163
generate_matricula, 163
generate_unique_cod_asignaturas, 164
gestores_df, 167
index, 167
lineas_actas_df, 167
matricula_df, 167
num_alumnos, 167
num_docentes, 168
num_gestores, 168
universidades_dict, 168
data.queries, 168
alumnos_all, 169
alumnos_egresados_genero_titulacion, 170
alumnos_egresados_nacionalidad_titulacion, 171
alumnos_genero_docente, 172
alumnos_nota_cualitativa_docente, 172
alumnos_nota_media_docente, 173
alumnos_nuevo_ingreso_genero_titulacion, 174
alumnos_nuevo_ingreso_nacionalidad_titulacion, 175
alumnos_repetidores_nuevos, 176
asignaturas_actas_titulacion, 176
asignaturas_docente, 177
asignaturas_matriculadas, 178
asignaturas_matriculadas_y_superadas, 178
asignaturas_superadas, 179
asignaturas_superadas_media_abandono, 180
cache_query, 181
calif_all_cualitativa_asignaturas, 181
calif_cualitativa_alumno_asignaturas, 182
calif_cualitativa_asignatura, 183
calif_cualitativa_comparativa, 184
calif_media_asignaturas, 185
calif_numerica_asignatura, 185
check_data, 186
curso_academico_actas_titulacion, 188
curso_academico_alumnado, 189
curso_academico_docente, 190
curso_academico_universidad, 190
cursos_academicos_egresados, 191
data_for_model, 192
docentes_all, 193
duracion_media_estudios nota_gestor, 193
gestores_all, 194
nota_media_acceso_titulacion, 195
nota_media_alumno_titulacion, 196
nota_media_general_mi_nota, 196
numero_alumnos_matriculados_universidad, 197
resumen_alumno, 198
resumen_docente, 199
resumen_gestor, 199
tasa_abandono_titulacion_gestor, 200
tasa_graduacion_titulacion_gestor, 201
titulacion_alumnado, 201
titulacion_docente, 202
titulaciones_universidad_gestor, 203
universidad_alumno, 203
universidades_docente, 204
universidades_gestor, 205
data.queries_dictionary, 206
queries, 207
data_for_model
 data.queries, 192
db
 data.db_connector, 157
db_url
 data.db_connector.DatabaseConnector, 225
debug
 app, 13
desc
 model, 211
docente_layout
 layouts.docente_layout, 208
docentes_all
 data.queries, 193
docentes_df
 data.generate_synthetic_data, 166
duracion_media_estudios_nota_gestor
 data.queries, 193
ebau_prueba_df
 data.generate_synthetic_data, 166
educacion
 data.generate_synthetic_data, 166
egresados_df
 data.generate_synthetic_data, 166
engine
 data.db_connector.DatabaseConnector, 225
execute_query
 data.db_connector.DatabaseConnector, 224
fake
 data.generate_synthetic_data, 167
filter_all_asignaturas_titulacion_docente
 components.docente.filters.filter_all_asignaturas_titulacion_docente, 132
filter_all_curso_academico
 components.docente.filters.filter_all_curso_academico, 133
filter_all_curso_academico_gestor
 components.gestor.filters.filter_all_curso_academico_gestor, 144
filter_asignaturas_docente
 components.docente.filters.filter_asignaturas_docente, 134
filter_asignaturas_matri_alumnado
 components.alumnado.filters.filter_asignaturas_matri_alumnado, 115
filter_curso_academico_alumnado
 components.alumnado.filters.filter_curso_academico_alumnado, 116
filter_curso_academico_docente
 components.docente.filters.filter_curso_academico_docente, 135

filter_curso_academico_gestor
 components.gestor.filters.filter_curso_academico_gestor, 145

filter_titulacion_alumnado
 components.alumnado.filters.filter_titulacion_alumnado, 117

filter_titulacion_docente
 components.docente.filters.filter_titulacion_docente, 136

filter_titulaciones_gestor
 components.gestor.filters.filter_titulaciones_gestor, 146

filters
 components.common.filters, 127

footer
 components.common.footer, 128

generate_alumnos
 data.generate_synthetic_data, 158

generate_asignaturas
 data.generate_synthetic_data, 158

generate_asignaturas_matriculadas
 data.generate_synthetic_data, 159

generate_csv
 callbacks.gestor.graphs.indicadores.callback_graph_egresados, 65

callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad, 71

callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso, 76

callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad, 82

callbacks.gestor.graphs.resultados.callback_graph_duracion_media, 88

callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion, 93

callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono, 98

callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion, 104

generate_curso_aca
 data.generate_synthetic_data, 159

generate_docentes
 data.generate_synthetic_data, 160

generate_ebau_prueba
 data.generate_synthetic_data, 161

generate_egresados
 data.generate_synthetic_data, 161

generate_gestores
 data.generate_synthetic_data, 162

generate_lineas_actas
 data.generate_synthetic_data, 163

generate_matricula
 data.generate_synthetic_data, 163

generate_unique_cod_asignaturas
 data.generate_synthetic_data, 164

gestor_layout
 layouts.gestor_layout, 208

gestores_all

data.queries, 194

gestores_df
 data.generate_synthetic_data, 167

get_data
 callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad, 66

callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad, 72

callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad, 77

callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad, 83

callbacks.gestor.graphs.resultados.callback_graph_duracion_media, 89

callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion, 94

callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono, 99

callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion, 105

graphs_general_alumnado
 components.alumnado.graphs.graphs_general_alumnado, 118

graphs_general_docente
 components.docente.graphs.graphs_general_docente, 137

graphs_nivelados_gestor
 components.gestor.graphs.graphs_nivelados_gestor, 149

graphs_personal_alumnado
 components.alumnado.graphs.graphs_personal_alumnado, 119

graphs_personal_docente
 components.docente.graphs.graphs_personal_docente, 137

graphs_resultados_gestor
 components.gestor.graphs.graphs_resultados_gestor, 149

graphs_tasa_abandono_gestor
 components.gestor.graphs.graphs_tasa_abandono_gestor, 150

grid_search
 model, 211

header
 components.common.header, 129

id_test
 model, 211

id_train
 model, 211

ids
 model, 211

index
 data.generate_synthetic_data, 167

initialize_dropdown
 callbacks.common.callback_user_role, 39

joblib_file

model, 212
layout
 app, 14
layouts, 207
layouts.alumno_layout, 207
 alumno_layout, 207
layouts.docente_layout, 208
 docente_layout, 208
layouts.gestor_layout, 208
 gestor_layout, 208
lineas_actas_df
 data.generate_synthetic_data, 167
list_to_tuple
 util, 216
load_data_for_model
 util, 218
load_model
 util, 219

matricula_df
 data.generate_synthetic_data, 167
mejores_estimadores
 model, 212
mejores_modelos
 model, 212
metrics
 model, 212
model, 209
 best_estimators, 210
 columnas_categoricas, 210
 columnas_numericas, 210
 cross_val_scores_voting, 210
 current_year, 210
 data, 211
 desc, 211
 grid_search, 211
 id_test, 211
 id_train, 211
 ids, 211
 joblib_file, 212
 mejores_estimadores, 212
 mejores_modelos, 212
 metrics, 212
 modelos, 212
 param_grids, 212
 poly, 213
 preprocessor, 213
 random_state, 213
 smote, 214
 stratify, 214
 test_size, 214
 total, 214
 voting_clf, 214
X, 214
X_test, 215
X_train, 215
y, 215
y_pred, 215
y_pred_voting, 215
y_test, 215
y_train, 216
modelos
 model, 212

not_data
 callbacks.alumnado.utils.callback_resumen_alumnado,
 32
 callbacks.docente.utils.callback_resumen_docente,
 56
 callbacks.gestor.utils.callback_resumen_gestor,
 109
nota_media_acceso_titulacion
 data.queries, 195
nota_media_alumno_titulacion
 data.queries, 196
nota_media_general_mi_nota
 data.queries, 196
num_alumnos
 data.generate_synthetic_data, 167
num_docentes
 data.generate_synthetic_data, 168
num_gestores
 data.generate_synthetic_data, 168
numero_alumnos_matriculados_universidad
 data.queries, 197

param_grids
 model, 212
poly
 model, 213
preprocessor
 model, 213
prevent_initial_call
 callbacks.common.callback_user_role, 40

queries
 data.queries_dictionary, 207

random_color
 util, 221
random_state
 model, 213
recomendador_alumnado
 components.alumnado.utils.recomendador_alumnado,
 120
recomendador_docente
 components.docente.utils.recomendador_docente,
 140
recomendador_gestor
 components.gestor.utils.recomendador_gestor,
 152
render_content
 callbacks.alumnado.utils.callback_tabs_alumnado,
 35
 callbacks.docente.utils.callback_tabs_docente, 59
 callbacks.gestor.utils.callback_tabs_gestor, 112
resumen_alumnado

components.alumnado.utils.resumen_alumnado,
 122
 resumen_alumno
 data.queries, 198
 resumen_docente
 components.docente.utils.resumen_docente, 141
 data.queries, 199
 resumen_gestor
 components.gestor.utils.resumen_gestor, 153
 data.queries, 199

 select_alumnado
 components.alumnado.utils.select_alumnado, 122
 select_docente
 components.docente.utils.select_docente, 142
 select_gestor
 components.gestor.utils.select_gestor, 154
 server
 app, 14
 sidebar
 components.common.sidebar, 131
 sidebarCollapse
 callbacks.common.callback_sidebarCollapse, 37
 smote
 model, 214
 store_selected_alumnado
 callbacks.alumnado.utils.callback_select_alumnado,
 34
 store_selected_docente
 callbacks.docente.utils.callback_select_docente,
 58
 store_selected_gestor
 callbacks.gestor.utils.callback_select_gestor, 111
 stratify
 model, 214

 tabs_alumnado
 components.alumnado.utils.tabs_alumnado, 123
 tabs_docente
 components.docente.utils.tabs_docente, 143
 tabs_gestor
 components.gestor.utils.tabs_gestor, 155
 tasa_abandono_titulacion_gestor
 data.queries, 200
 tasa_graduacion_titulacion_gestor
 data.queries, 201
 test_size
 model, 214
 title
 app, 14
 titulacion_alumnado
 data.queries, 201
 titulacion_docente
 data.queries, 202
 titulaciones_universidad_gestor
 data.queries, 203
 toggle_modal
 callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionales,
 67

 callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionales,
 73
 callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacional
 78
 callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacional
 84
 callbacks.gestor.graphs.resultados.callback_graph_duracion_media
 90
 callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulo
 95
 callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono
 101
 callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_grado
 106
 total
 model, 214

 universidad_alumno
 data.queries, 203
 universidades_dict
 data.generate_synthetic_data, 168
 universidades_docente
 data.queries, 204
 universidades_gestor
 data.queries, 205
 update_filter_all_curso_academico_gestor
 callbacks.gestor.filters.callback_filter_all_curso_academico_gestor,
 61
 update_filter_all_cursos_academicos_docente
 callbacks.docente.filters.callback_filter_all_curso_academico,
 42
 update_filter_asignaturas_docente
 callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente,
 40
 callbacks.docente.filters.callback_filter_asignaturas_docente,
 43
 update_filter_asignaturas_matri_alumnado
 callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado,
 15
 update_filter_curso_academico_alumnado
 callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado,
 16
 update_filter_curso_academico_docente
 callbacks.docente.filters.callback_filter_curso_academico_docente,
 44
 update_filter_curso_academico_gestor
 callbacks.gestor.filters.callback_filter_curso_academico_gestor,
 62
 update_filter_titulacion_alumnado
 callbacks.alumnado.filters.callback_filter_titulacion_alumnado,
 18
 update_filter_titulacion_docente
 callbacks.docente.filters.callback_filter_titulacion_docente,
 45
 update_filter_titulaciones_gestor
 callbacks.gestor.filters.callback_filter_titulaciones_gestor,
 63

callbacks.alumnado.graphs.general.callback_graph_asignaciones_media,
19
callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor,
callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa,
21
callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor,
callbacks.alumnado.graphs.general.callback_graph_calif_media_t55i_nota,
23
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_egresados,
callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado,
25
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_egresados,
callbacks.alumnado.graphs.personal.callback_graph_calif_num_t56ca_alumnado,
27
callbacks.gestor.graphs.resultados.callback_graph_duracion_media_egresados,
callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado,
29
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulo_egresados,
callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado,
30
callbacks.gestor.graphs riesgo_abandono.callback_graph_tasa_abandono,
update_graph_docente
103
callbacks.docente.graphs.general.callback_graph_all_calif_t57calif_media_docente,
46
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion,
108
callbacks.docente.graphs.general.callback_graph_all_t58media_docente,
48
callbacks.gestor.graphs.config_mode_bar_buttons_gestor, 221
callbacks.docente.graphs.personal.callback_graph_alu_gehert_t59t59media_docente,
50
callbacks.gestor.graphs.load_data_for_model, 218
callbacks.docente.graphs.personal.callback_graph_alu_medium_t59media_docente,
51
callbacks.gestor.graphs.random_color, 221
callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente,
53
callbacks.gestor.graphs.voting_clf
callbacks.docente.graphs.personal.callback_graph_alu_repetidores_t59nuevos_docente,
54
X
update_graph_gestor
callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor,
model, 214
68
X_test
callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor,
model, 215
74
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_t59ingreso_genero_gestor,
79
model, 215
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_t59ingreso_nacionalidad_gestor,
85
model, 215
callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor,
model, 215
91
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor,
model, 215
101
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor,
model, 215
107
y_train
update_grph_t_gestor
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor,
model, 216
96
y_train
update_layout
callbacks.common.callback_update_layout, 37
update_resumen_alumnado
callbacks.alumnado.utils.callback_resumen_alumnado,
32
update_resumen_docente
callbacks.docente.utils.callback_resumen_docente,
56
update_resumen_gestor
callbacks.gestor.utils.callback_resumen_gestor,
110
update_role
callbacks.common.callback_user_role, 39