Business Intelligence Universidades

Generado por Doxygen 1.9.3

1 Indice de namespaces 1
1.1 Lista de 'namespaces' . . . 1

2 Índice de clases 5
2.1 Lista de clases . . . 5

3 Indice 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_sidebar_collapse . . . 36
4.23.1 Documentación de las funciones . . . 37
4.23.1.1 sidebar_collapse() . . . 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_genero_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_genero_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_grpht_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 . . . 139
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 . . . 141
4.102.1 Documentación de las funciones . . . 141
4.102.1.1 select_docente() . . . 142
4.103 Referencia del Namespace components.docente.utils.tabs_docente . . . 142
4.103.1 Documentación de las funciones . . . 142
4.103.1.1 tabs_docente() . . . 143
4.104 Referencia del Namespace components.gestor . . . 143
4.105 Referencia del Namespace components.gestor.filters . . . 143
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() . . . 147
4.111 Referencia del Namespace components.gestor.graphs.graphs_resultados_gestor . . . 148
4.111.1 Documentación de las funciones . . . 148
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() . . . 150
4.113 Referencia del Namespace components.gestor.utils . . . 151
4.114 Referencia del Namespace components.gestor.utils.recomendador_gestor . . . 151
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 . . . 152
4.115.1 Documentación de las funciones . . . 153
4.115.1.1 resumen_gestor() . . . 153
4.116 Referencia del Namespace components.gestor.utils.select_gestor . . . 153
4.116.1 Documentación de las funciones . . . 153
4.116.1.1 select_gestor() . . . 154
4.117 Referencia del Namespace components.gestor.utils.tabs_gestor . . . 154
4.117.1 Documentación de las funciones . . . 154
4.117.1.1 tabs_gestor() . . . 155
4.118 Referencia del Namespace data . . . 155
4.119 Referencia del Namespace data.db_connector . . . 155
4.119.1 Documentación de las variables . . . 156
4.119.1.1 config . . . 156
4.119.1.2 db . . . 156
4.120 Referencia del Namespace data.generate_synthetic_data . . . 156
4.120.1 Documentación de las funciones . . . 157
4.120.1.1 generate_alumnos() . . . 157
4.120.1.2 generate_asignaturas() . . . 158
4.120.1.3 generate_asignaturas_matriculadas() . . . 158
4.120.1.4 generate_curso_aca() . . . 158
4.120.1.5 generate_docentes() . . . 159
4.120.1.6 generate_ebau_prueba() . . . 159
4.120.1.7 generate_egresados() . . . 160
4.120.1.8 generate_gestores() . . . 160
4.120.1.9 generate_lineas_actas() . . . 160
4.120.1.10 generate_matricula() . . . 161
4.120.1.11 generate_unique_cod_asignaturas() . . . 162
4.120.2 Documentación de las variables . . . 162
4.120.2.1 alumnos_df . . . 162
4.120.2.2 asignaturas_dict . . . 162
4.120.2.3 asignaturas_matriculadas_df . . . 162
4.120.2.4 cod_asignaturas_dict . . . 163
4.120.2.5 cod_plan_dict . . . 163
4.120.2.6 cursos_academicos . . . 163
4.120.2.7 docentes_df . . . 163
4.120.2.8 ebau_prueba_df . . . 163
4.120.2.9 educacion . . . 163
4.120.2.10 egresados_df . . . 164
4.120.2.11 fake . . . 164
4.120.2.12 gestores_df . . . 164
4.120.2.13 index . . . 164
4.120.2.14 lineas_actas_df . . . 164

4.120.2.15 matricula_df . . . 164
4.120.2.16 num_alumnos . . . 165
4.120.2.17 num_docentes . . . 165
4.120.2.18 num_gestores . . . 165
4.120.2.19 universidades_dict . . . 165
4.121 Referencia del Namespace data.queries . . . 165
4.121.1 Documentación de las funciones . . . 166
4.121.1.1 alumnos_all() . . . 167
4.121.1.2 alumnos_egresados_genero_titulacion() . . . 167
4.121.1.3 alumnos_egresados_nacionalidad_titulacion() . . . 168
4.121.1.4 alumnos_genero_docente() . . . 169
4.121.1.5 alumnos_nota_cualitativa_docente() . . . 170
4.121.1.6 alumnos_nota_media_docente() . . . 170
4.121.1.7 alumnos_nuevo_ingreso_genero_titulacion() . . . 171
4.121.1.8 alumnos_nuevo_ingreso_nacionalidad_titulacion() . . . 172
4.121.1.9 alumnos_repetidores_nuevos() . . . 173
4.121.1.10 asignaturas_actas_titulacion() . . . 173
4.121.1.11 asignaturas_docente() . . . 174
4.121.1.12 asignaturas_matriculadas() . . . 175
4.121.1.13 asignaturas_matriculadas_y_superadas() . . . 176
4.121.1.14 asignaturas_superadas() . . . 176
4.121.1.15 asignaturas_superadas_media_abandono() . . . 177
4.121.1.16 cache_query() . . . 178
4.121.1.17 calif_all_cualitativa_asignaturas() . . . 178
4.121.1.18 calif_cualitativa_alumno_asignaturas() . . . 179
4.121.1.19 calif_cualitativa_asignatura() . . . 180
4.121.1.20 calif_cualitativa_comparativa() . . . 181
4.121.1.21 calif_media_asignaturas() . . . 182
4.121.1.22 calif_numerica_asignatura() . . . 182
4.121.1.23 check_data() . . . 183
4.121.1.24 curso_academico_actas_titulacion() . . . 184
4.121.1.25 curso_academico_alumnado() . . . 185
4.121.1.26 curso_academico_docente() . . . 186
4.121.1.27 curso_academico_universidad() . . . 187
4.121.1.28 cursos_academicos_egresados() . . . 187
4.121.1.29 data_for_model() . . . 188
4.121.1.30 docentes_all() . . . 189
4.121.1.31 duracion_media_estudios_nota_gestor() . . . 190
4.121.1.32 gestores_all() . . . 190
4.121.1.33 nota_media_acceso_titulacion() . . . 191
4.121.1.34 nota_media_alumno_titulacion() . . . 192
4.121.1.35 nota_media_general_mi_nota() . . . 193

4.121.1.36 numero_alumnos_matriculados_universidad() . . . 193
4.121.1.37 resumen_alumno() . . . 194
4.121.1.38 resumen_docente() . . . 195
4.121.1.39 resumen_gestor() . . . 195
4.121.1.40 tasa_abandono_titulacion_gestor() . . . 196
4.121.1.41 tasa_graduacion_titulacion_gestor() . . . 197
4.121.1.42 titulacion_alumnado() . . . 197
4.121.1.43 titulacion_docente() . . . 198
4.121.1.44 titulaciones_universidad_gestor() . . . 199
4.121.1.45 universidad_alumno() . . . 200
4.121.1.46 universidades_docente() . . . 200
4.121.1.47 universidades_gestor() . . . 201
4.122 Referencia del Namespace data.queries_dictionary . . . 202
4.122.1 Documentación de las variables . . . 203
4.122.1.1 queries . . . 203
4.123 Referencia del Namespace layouts . . . 203
4.124 Referencia del Namespace layouts.alumno_layout . . . 203
4.124.1 Documentación de las funciones . . . 203
4.124.1.1 alumno_layout() . . . 203
4.125 Referencia del Namespace layouts.docente_layout . . . 204
4.125.1 Documentación de las funciones . . . 204
4.125.1.1 docente_layout() . . . 204
4.126 Referencia del Namespace layouts.gestor_layout . . . 204
4.126.1 Documentación de las funciones . . . 204
4.126.1.1 gestor_layout() . . . 205
4.127 Referencia del Namespace model . . . 205
4.127.1 Documentación de las variables . . . 206
4.127.1.1 best_estimators . . . 206
4.127.1.2 columnas_categoricas . . . 206
4.127.1.3 columnas_numericas . . . 206
4.127.1.4 cross_val_scores_voting . . . 206
4.127.1.5 current_year . . . 207
4.127.1.6 data . . . 207
4.127.1.7 desc . . . 207
4.127.1.8 grid_search . . . 207
4.127.1.9 id_test . . . 207
4.127.1.10 id_train . . . 207
4.127.1.11 ids . . . 208
4.127.1.12 joblib_file . . . 208
4.127.1.13 modelos . . . 208
4.127.1.14 param_grids . . . 208
4.127.1.15 poly . . . 209

4.127.1.16 predicciones . . . 209
4.127.1.17 preprocessor . . . 209
4.127.1.18 random_state . . . 209
4.127.1.19 smote . . . 209
4.127.1.20 stratify . . . 210
4.127.1.21 test_size . . . 210
4.127.1.22 total . . . 210
4.127.1.23 voting_clf . . . 210
4.127.1.24 X . . . 210
4.127.1.25 X_test . . . 210
4.127.1.26 X_train . . . 211
4.127.1.27 y . . . 211
4.127.1.28 y_pred_voting . . . 211
4.127.1.29 y_test . . . 211
4.127.1.30 y_train . . . 211
4.128 Referencia del Namespace util . . . 211
4.128.1 Documentación de las funciones . . . 212
4.128.1.1 list_to_tuple() . . . 212
4.128.1.2 load_data_for_model() . . . 213
4.128.1.3 load_model() . . . 214
4.128.1.4 random_color() . . . 216
4.128.2 Documentación de las variables . . . 216
4.128.2.1 config_mode_bar_buttons_gestor . . . 216

5 Documentación de las clases **217**
5.1 Referencia de la Clase data.db_connector.DatabaseConnector . . . 217
5.1.1 Descripción detallada . . . 217
5.1.2 Documentación del constructor y destructor . . . 217
5.1.2.1 __init__() . . . 218
5.1.3 Documentación de las funciones miembro . . . 218
5.1.3.1 close() . . . 218
5.1.3.2 execute_query() . . . 218
5.1.4 Documentación de los datos miembro . . . 219
5.1.4.1 db_url . . . 219
5.1.4.2 engine . . . 219

6 Documentación de archivos **221**
6.1 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/app.py . . . 221
6.2 app.py . . . 221
6.3 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/__init__.py . . . 222
6.4 __init__.py . . . 222

6.5 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/__init__.py . . . 222
6.6 __init__.py . . . 222
6.7 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/filters/__init__.py . . . 222
6.8 __init__.py . . . 222
6.9 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/__init__.py . . . 222
6.10 __init__.py . . . 222
6.11 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/general/__init__.py . . . 223
6.12 __init__.py . . . 223
6.13 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/personal/__init__.py . . . 223
6.14 __init__.py . . . 223
6.15 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/utils/__init__.py . . . 223
6.16 __init__.py . . . 223
6.17 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/common/__init__.py . . . 223
6.18 __init__.py . . . 223
6.19 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/docente/__init__.py . . . 223
6.20 __init__.py . . . 223
6.21 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/docente/filters/__init__.py . . . 224
6.22 __init__.py . . . 224
6.23 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/docente/graphs/__init__.py . . . 224
6.24 __init__.py . . . 224
6.25 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/docente/graphs/general/__init__.py . . . 224
6.26 __init__.py . . . 224
6.27 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/docente/graphs/personal/__init__.py . . . 224
6.28 __init__.py . . . 224
6.29 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/docente/utils/__init__.py . . . 224
6.30 __init__.py . . . 224
6.31 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/gestor/__init__.py . . . 225
6.32 __init__.py . . . 225
6.33 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/gestor/filters/__init__.py . . . 225
6.34 __init__.py . . . 225

6.35 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/gestor/graphs/__init__.py . . . 225
6.36 __init__.py . . . 225
6.37 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/gestor/graphs/indicadores/__init__.py . . . 225
6.38 __init__.py . . . 225
6.39 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/gestor/graphs/resultados/__init__.py . . . 225
6.40 __init__.py . . . 225
6.41 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/gestor/graphs/riesgo_abandono/__init__.py . . . 226
6.42 __init__.py . . . 226
6.43 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/gestor/utils/__init__.py . . . 226
6.44 __init__.py . . . 226
6.45 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/__init__.py . . . 226
6.46 __init__.py . . . 226
6.47 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/alumnado/__init__.py . . . 226
6.48 __init__.py . . . 226
6.49 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/alumnado/filters/__init__.py . . . 226
6.50 __init__.py . . . 226
6.51 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/alumnado/graphs/__init__.py . . . 227
6.52 __init__.py . . . 227
6.53 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/alumnado/utils/__init__.py . . . 227
6.54 __init__.py . . . 227
6.55 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/common/__init__.py . . . 227
6.56 __init__.py . . . 227
6.57 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/docente/__init__.py . . . 227
6.58 __init__.py . . . 227
6.59 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/docente/filters/__init__.py . . . 227
6.60 __init__.py . . . 227
6.61 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/docente/graphs/__init__.py . . . 228
6.62 __init__.py . . . 228
6.63 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/docente/utils/__init__.py . . . 228
6.64 __init__.py . . . 228

6.65 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/gestor/__init__.py . . . 228
6.66 __init__.py . . . 228
6.67 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/gestor/filters/__init__.py . . . 228
6.68 __init__.py . . . 228
6.69 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/gestor/graphs/__init__.py . . . 228
6.70 __init__.py . . . 228
6.71 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/components/gestor/utils/__init__.py . . . 229
6.72 __init__.py . . . 229
6.73 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/data/__init__.py . . . 229
6.74 __init__.py . . . 229
6.75 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/layouts/__init__.py . . . 229
6.76 __init__.py . . . 229
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 . . . 229
6.78 callback_filter_asignaturas_matri_alumnado.py . . . 230
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 . . . 230
6.80 callback_filter_curso_cademico_alumnado.py . . . 231
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 . . . 231
6.82 callback_filter_titulacion_alumnado.py . . . 232
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 . . . 232
6.84 callback_graph_asig_superadas_media.py . . . 233
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 . . . 234
6.86 callback_graph_calif_cual_comparativa.py . . . 234
6.87 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/general/callback_graph_calif_media_mi_nota.py . . . 236
6.88 callback_graph_calif_media_mi_nota.py . . . 236
6.89 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/personal/callback_graph_calif_cualitativa_alumnado.py . . . 238
6.90 callback_graph_calif_cualitativa_alumnado.py . . . 238
6.91 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/personal/callback_graph_calif_numerica_alumnado.py . . . 239
6.92 callback_graph_calif_numerica_alumnado.py . . . 239
6.93 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/personal/callback_graph_progreso_academico_alumnado.py . . . 241
6.94 callback_graph_progreso_academico_alumnado.py . . . 241

6.95 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/graphs/personal/callback_graph_tasa_rendimiento_alumnado.py . . . 242
6.96 callback_graph_tasa_rendimiento_alumnado.py . . . 242
6.97 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/utils/callback_resumen_alumnado.py . . . 243
6.98 callback_resumen_alumnado.py . . . 244
6.99 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/alumnado/utils/callback_select_alumnado.py . . . 245
6.100 callback_select_alumnado.py . . . 245
6.101 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/alumnado/utils/callback_tabs_alumnado.py . . . 245
6.102 callback_tabs_alumnado.py . . . 246
6.103 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/common/callback_sidebar_collapse.py . . . 247
6.104 callback_sidebar_collapse.py . . . 247
6.105 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/common/callback_update_layout.py . . . 247
6.106 callback_update_layout.py . . . 248
6.107 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/common/callback_user_role.py . . . 248
6.108 callback_user_role.py . . . 249
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 . . . 249
6.110 callback_filter_all_asignaturas_titulacion_docente.py . . . 250
6.111 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/filters/callback_filter_all_curso_academico.py . . . 250
6.112 callback_filter_all_curso_academico.py . . . 251
6.113 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/filters/callback_filter_asignaturas_docente.py . . . 251
6.114 callback_filter_asignaturas_docente.py . . . 251
6.115 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/filters/callback_filter_curso_academico_docente.py . . . 252
6.116 callback_filter_curso_academico_docente.py . . . 252
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 . . . 253
6.118 callback_filter_titulacion_docente.py . . . 253
6.119 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/graphs/general/callback_graph_all_calif_cualitativa_docente.py . . . 254
6.120 callback_graph_all_calif_cualitativa_docente.py . . . 254
6.121 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/graphs/general/callback_graph_all_calif_media_docente.py . . . 256
6.122 callback_graph_all_calif_media_docente.py . . . 256
6.123 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_genero_docente.py . . . 257
6.124 callback_graph_alu_genero_docente.py . . . 257

6.125 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_media_docente.py 258
6.126 callback_graph_alu_media_docente.py 259
6.127 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_nota_cualitativa_docente.py . . . 259
6.128 callback_graph_alu_nota_cualitativa_docente.py 260
6.129 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/graphs/personal/callback_graph_alu_repetidores_nuevos_docente.py 261
6.130 callback_graph_alu_repetidores_nuevos_docente.py 261
6.131 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/utils/callback_resumen_docente.py 262
6.132 callback_resumen_docente.py 263
6.133 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/utils/callback_select_docente.py 263
6.134 callback_select_docente.py 264
6.135 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/docente/utils/callback_tabs_docente.py 264
6.136 callback_tabs_docente.py 264
6.137 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/filters/callback_filter_all_curso_academico_gestor.py 265
6.138 callback_filter_all_curso_academico_gestor.py 266
6.139 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/filters/callback_filter_curso_academico_gestor.py 266
6.140 callback_filter_curso_academico_gestor.py 267
6.141 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/filters/callback_filter_titulaciones_gestor.py 267
6.142 callback_filter_titulaciones_gestor.py 268
6.143 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_egresados_genero_gestor.py 268
6.144 callback_graph_egresados_genero_gestor.py 269
6.145 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_egresados_nacionalidad_gestor.py . 271
6.146 callback_graph_egresados_nacionalidad_gestor.py 272
6.147 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_nuevo_ingreso_genero_gestor.py . 274
6.148 callback_graph_nuevo_ingreso_genero_gestor.py 275
6.149 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/indicadores/callback_graph_nuevo_ingreso_nacionalidad_↩ gestor.py 277
6.150 callback_graph_nuevo_ingreso_nacionalidad_gestor.py 278
6.151 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/resultados/callback_graph_duracion_media_estudios_nota_↩ gestor.py 280
6.152 callback_graph_duracion_media_estudios_nota_gestor.py 281
6.153 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/resultados/callback_graph_nota_acceso_titulacion_gestor.py . . 283
6.154 callback_graph_nota_acceso_titulacion_gestor.py 284

6.155 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/riesgo_abandono/callback_graph_tasa_abandono_gestor.py . . 286
6.156 callback_graph_tasa_abandono_gestor.py 286
6.157 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/graphs/riesgo_abandono/callback_graph_tasa_graduacion_gestor.py . . 288
6.158 callback_graph_tasa_graduacion_gestor.py 289
6.159 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/utils/callback_resumen_gestor.py 291
6.160 callback_resumen_gestor.py 291
6.161 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/utils/callback_select_gestor.py 292
6.162 callback_select_gestor.py 293
6.163 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/callbacks/gestor/utils/callback_tabs_gestor.py 293
6.164 callback_tabs_gestor.py 293
6.165 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/filters/filter_asignaturas_matri_alumnado.py 295
6.166 filter_asignaturas_matri_alumnado.py 295
6.167 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/filters/filter_curso_academico_alumnado.py 295
6.168 filter_curso_academico_alumnado.py 296
6.169 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/filters/filter_titulacion_alumnado.py 296
6.170 filter_titulacion_alumnado.py 297
6.171 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/graphs/graphs_general_alumnado.py 297
6.172 graphs_general_alumnado.py 297
6.173 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/graphs/graphs_personal_alumnado.py 298
6.174 graphs_personal_alumnado.py 298
6.175 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/utils/recomendador_alumnado.py 299
6.176 recomendador_alumnado.py 299
6.177 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/utils/resumen_alumnado.py 300
6.178 resumen_alumnado.py 300
6.179 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/utils/select_alumnado.py 300
6.180 select_alumnado.py 301
6.181 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/alumnado/utils/tabs_alumnado.py 301
6.182 tabs_alumnado.py 301
6.183 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/common/create_graph.py 302
6.184 create_graph.py 302

6.185 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/common/create_graph_with_table.py . . . 303
6.186 create_graph_with_table.py . . . 303
6.187 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/common/filters.py . . . 303
6.188 filters.py . . . 304
6.189 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/common/footer.py . . . 304
6.190 footer.py . . . 304
6.191 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/common/header.py . . . 304
6.192 header.py . . . 305
6.193 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/common/modal_data.py . . . 305
6.194 modal_data.py . . . 306
6.195 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/common/sidebar.py . . . 306
6.196 sidebar.py . . . 306
6.197 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/filters/filter_all_asignaturas_titulacion_docente.py . . . 307
6.198 filter_all_asignaturas_titulacion_docente.py . . . 307
6.199 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/filters/filter_all_curso_academico.py . . . 308
6.200 filter_all_curso_academico.py . . . 308
6.201 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/filters/filter_asignaturas_docente.py . . . 308
6.202 filter_asignaturas_docente.py . . . 309
6.203 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/filters/filter_curso_academico_docente.py . . . 309
6.204 filter_curso_academico_docente.py . . . 310
6.205 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/filters/filter_titulacion_docente.py . . . 310
6.206 filter_titulacion_docente.py . . . 310
6.207 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/graphs/graphs_general_docente.py . . . 311
6.208 graphs_general_docente.py . . . 311
6.209 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/graphs/graphs_personal_docente.py . . . 312
6.210 graphs_personal_docente.py . . . 312
6.211 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/utils/recomendador_docente.py . . . 312
6.212 recomendador_docente.py . . . 313
6.213 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/utils/resumen_docente.py . . . 313
6.214 resumen_docente.py . . . 314

6.215 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/utils/select_docente.py . . . 314
6.216 select_docente.py . . . 314
6.217 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/docente/utils/tabs_docente.py . . . 315
6.218 tabs_docente.py . . . 315
6.219 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/filters/filter_all_curso_academico_gestor.py . . . 315
6.220 filter_all_curso_academico_gestor.py . . . 316
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 . . . 316
6.222 filter_curso_academico_gestor.py . . . 317
6.223 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/filters/filter_titulaciones_gestor.py . . . 317
6.224 filter_titulaciones_gestor.py . . . 317
6.225 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/graphs/graphs_indicadores_gestor.py . . . 318
6.226 graphs_indicadores_gestor.py . . . 318
6.227 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/graphs/graphs_resultados_gestor.py . . . 319
6.228 graphs_resultados_gestor.py . . . 320
6.229 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/graphs/graphs_riesgo_abandono_gestor.py . . . 320
6.230 graphs_riesgo_abandono_gestor.py . . . 321
6.231 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/utils/recomendador_gestor.py . . . 321
6.232 recomendador_gestor.py . . . 322
6.233 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/utils/resumen_gestor.py . . . 322
6.234 resumen_gestor.py . . . 322
6.235 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/utils/select_gestor.py . . . 323
6.236 select_gestor.py . . . 323
6.237 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/components/gestor/utils/tabs_gestor.py . . . 324
6.238 tabs_gestor.py . . . 324
6.239 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/data/db_connector.py . . . 324
6.240 db_connector.py . . . 325
6.241 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/data/generate_synthetic_data.py . . . 326
6.242 generate_synthetic_data.py . . . 327
6.243 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/data/queries.py . . . 331
6.244 queries.py . . . 332

6.245 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/data/queries_dictionary.py . . . 338
6.246 queries_dictionary.py . . . 338
6.247 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/layouts/alumno_layout.py . . . 347
6.248 alumno_layout.py . . . 347
6.249 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/layouts/docente_layout.py . . . 348
6.250 docente_layout.py . . . 348
6.251 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/layouts/gestor_layout.py . . . 348
6.252 gestor_layout.py . . . 348
6.253 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/model.py . . . 349
6.254 model.py . . . 349
6.255 Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities↩ _TFG/src/util.py . . . 351
6.256 util.py . . . 352

Índice alfabético **355**

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_sidebar_collapse . . . 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 . . . 139
components.docente.utils.recomendador_docente . . . 140
components.docente.utils.resumen_docente . . . 141
components.docente.utils.select_docente . . . 141
components.docente.utils.tabs_docente . . . 142
components.gestor . . . 143
components.gestor.filters . . . 143
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 . . . 148
components.gestor.graphs.graphs_riesgo_abandono_gestor . . . 150
components.gestor.utils . . . 151
components.gestor.utils.recomendador_gestor . . . 151
components.gestor.utils.resumen_gestor . . . 152
components.gestor.utils.select_gestor . . . 153
components.gestor.utils.tabs_gestor . . . 154
data . . . 155
data.db_connector . . . 155
data.generate_synthetic_data . . . 156
data.queries . . . 165
data.queries_dictionary . . . 202
layouts . . . 203
layouts.alumno_layout . . . 203
layouts.docente_layout . . . 204
layouts.gestor_layout . . . 204
model . . . 205
util . . . 211

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 217

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/app.py 221
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/model.py 349
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/util.py 351
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/__init__.py 222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/__init__.py 222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/__init__.py 222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/callback_filter_asigr 229
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/callback_filter_curso 230
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/filters/callback_filter_titula 231
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/__init__.py 222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/__init__.py 223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/callback_g 232
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/callback_g 234
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/general/callback_g 236
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/__init__.p 223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_ 238
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_ 239
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_ 241
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/graphs/personal/callback_ 242
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/__init__.py 223

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/callback_resumen_al
243
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/callback_select_alum
245
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/alumnado/utils/callback_tabs_alumn
245
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/__init__.py
223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/callback_sidebar_collapse.
247
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/callback_update_layout.py
247
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/common/callback_user_role.py
248
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/__init__.py
223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/__init__.py
224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/callback_filter_all_asig
249
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/callback_filter_all_cur
250
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/callback_filter_asigna
251
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/callback_filter_curso_
252
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/filters/callback_filter_titulaci
253
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/__init__.py
224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/general/__init__.py
224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/general/callback_gra
254
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/general/callback_gra
256
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/__init__.py
224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/callback_g
257
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/callback_g
258
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/callback_g
259
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/graphs/personal/callback_g
261
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/__init__.py
224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/callback_resumen_doc
262
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/callback_select_docent
263
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/docente/utils/callback_tabs_docente
264
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/__init__.py
225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/__init__.py
225

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/callback_filter_all_curso
265
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/callback_filter_curso_ac
266
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/filters/callback_filter_titulacion
267
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/__init__.py
225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/__init__.p
225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/callback_
268
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/callback_
271
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/callback_
274
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/indicadores/callback_
277
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/__init__.py
225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/callback_g
280
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/resultados/callback_g
283
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo↩
_abandono/__init__.py 226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo↩
_abandono/callback_graph_tasa_abandono_gestor.py 286
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/graphs/riesgo↩
_abandono/callback_graph_tasa_graduacion_gestor.py 288
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/__init__.py
226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/callback_resumen_gesto
291
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/callback_select_gestor.p
292
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/callbacks/gestor/utils/callback_tabs_gestor.py
293
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/__init__.py
226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/__init__.py
226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/__init__.py
226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/filter_asignaturas
295
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/filter_curso_acad
295
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/filters/filter_titulacion_a
296
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/graphs/__init__.py
227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/graphs/graphs_genera
297
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/graphs/graphs_person
298
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/__init__.py
227

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/recomendador_alu
299
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/resumen_alumnad
300
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/select_alumnado.p
300
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/alumnado/utils/tabs_alumnado.py
301
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/__init__.py
227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/create_graph.py
302
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/create_graph_with_table
303
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/filters.py
303
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/footer.py
304
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/header.py
304
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/modal_data.py
305
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/common/sidebar.py
306
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/__init__.py
227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/__init__.py
227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_all_asignatura
307
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_all_curso_aca
308
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_asignaturas_
308
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_curso_acade
309
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/filters/filter_titulacion_doc
310
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/graphs/__init__.py
228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/graphs/graphs_general_
311
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/graphs/graphs_personal_
312
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/__init__.py
228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/recomendador_doce
312
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/resumen_docente.p
313
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/select_docente.py
314
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/tabs_docente.py
315
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/__init__.py
228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/__init__.py
228

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_all_curso_acad
315
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_curso_academ
316
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_titulaciones_ge
317
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/__init__.py
228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_indicadores
318
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_resultados_
319
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/graphs_riesgo_aba
320
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/__init__.py
229
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/recomendador_gestor
321
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/resumen_gestor.py
322
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/select_gestor.py
323
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/utils/tabs_gestor.py
324
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/data/__init__.py . . . 229
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/data/db_connector.py 324
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/data/generate_synthetic_data.py
326
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/data/queries.py . . . 331
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/data/queries_dictionary.py
338
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/layouts/__init__.py . . 229
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/layouts/alumno_layout.py
347
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/layouts/docente_layout.py
348
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/layouts/gestor_layout.py
348

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 23 del archivo app.py.

4.1.1.2. app

```
app.app
```

Valor inicial:

```
=  dash.Dash(
    __name__,
    external_stylesheets=[
        'src/assets/css/styles.css',
        dbc.themes.BOOTSTRAP
    ],
    suppress_callback_exceptions=True
)
```

Definición en la línea 13 del archivo app.py.

4.1.1.3. debug

```
app.debug
```

Definición en la línea 37 del archivo app.py.

4.1.1.4. layout

```
app.layout
```

Definición en la línea 26 del archivo app.py.

4.1.1.5. server

```
app.server = app.server
```

Definición en la línea 24 del archivo app.py.

4.1.1.6. title

```
app.title
```

Definición en la línea 22 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_cademico_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 14 del archivo callback_filter_asignaturas_matri_alumnado.py.

```
00014 def update_filter_asignaturas_matri_alumnado(alumno_id, curso_academico, titulacion, n_clicks,
      existing_options):
00015     """
00016 Actualiza las opciones del dropdown de asignaturas matriculadas por el alumno y
00017 gestiona el evento del botón "Seleccionar todo".
00018
00019     Args:
00020     alumno_id (str): Identificador del alumno.
00021     curso_academico (list): Lista con los cursos académicos
00022     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
    """

ctx = callback_context
trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None

    # Evento de selección de todas las asignaturas matriculadas
    if trigger_id == 'select-all-button' and n_clicks > 0:
        return existing_options, [option['value'] for option in existing_options]

    if not (alumno_id and curso_academico and titulacion):
        return [], None

    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        return [], None

    data = asignaturas_matriculadas(alumno_id, curso_academico, titulacion)

    if not data:
        return [], None

    opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
    value = [option['value'] for option in opciones_dropdown]

    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 13 del archivo callback_filter_curso_cademico_alumnado.py.

```
00013 def update_filter_curso_academico_alumnado(alumno_id, titulacion, n_clicks, existing_options):
00014     """
00015 Actualiza las opciones del dropdown de cursos académicos y
00016 gestiona el evento del botón "Seleccionar todo".
00017
00018     Args:
00019     alumno_id (str): Identificador del alumno.
00020     titulacion (str): Titulación seleccionada
00021     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00022     existing_options (list): Opciones actuales del dropdown
00023
00024     Returns:
00025     list: Opciones del dropdown
00026     list: Valor seleccionado
00027     """
00028
00029 ctx = callback_context
00030 trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None
00031
00032     # Evento de selección de todos los cursos académicos
00033     if trigger_id == 'select-all-cursos-academicos' and n_clicks > 0:
00034         return existing_options, [option['value'] for option in existing_options]
00035
00036     if not (alumno_id and titulacion):
00037         return [], None
00038
00039     # Obtener los cursos académicos desde la base de datos
00040     data = curso_academico_alumnado(alumno_id, titulacion)
00041
00042     if not data:
00043         return [], None
00044
00045     opciones_dropdown = [{'label': curso[0], 'value': curso[0]} for curso in data]
00046     value = [option['value'] for option in opciones_dropdown]
00047
00048     return opciones_dropdown, value
00049
```

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 12 del archivo callback_filter_titulacion_alumnado.py.

```
00012 def update_filter_titulacion_alumnado(alumno_id, selected_value, stored_titulacion):
00013     """
00014 Actualiza las opciones del dropdown de titulaciones del perfil "Alumno".
00015
00016     Args:
00017     alumno_id (str): Identificador del alumno.
00018     selected_value (str): Valor seleccionado en el dropdown
00019     stored_titulacion (str): Titulación almacenada
00020
00021     Returns:
00022     list: Opciones del dropdown
00023     str: Valor seleccionado
00024
00025     """
00026 if not alumno_id:
00027         return [], None, None
00028
00029     data = titulacion_alumnado(alumno_id)
00030
00031     if not data:
00032         return [], None, None
00033
00034     opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
00035
00036     # Si no hay valor seleccionado pero hay un valor almacenado, lo usamos si está en las opciones
00037     if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
00038         return opciones_dropdown, stored_titulacion, stored_titulacion
00039
00040     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 15 del archivo callback_graph_asig_superadas_media.py.

```
def 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

    """
fig = go.Figure()

fig.update_layout(
title={
"text": "Relación nota media y número de asignaturas superadas por alumno",
            "x": 0.5,
        },
        xaxis_title="Nota Media",
        yaxis_title="N° Asignaturas superadas",
        legend_title="Estado del alumno",
        showlegend=True,
        xaxis=dict(range=[0, 10]),
        yaxis=dict(range=[0, 40]),
    )

    if not (curso_academico and alumno_id and asignaturas_matriculadas and titulacion):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
        asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
    except Exception as e:
        print("Error:", e)
        return fig

    data_universidad = universidad_alumno(alumno_id)

    if not data_universidad:
        return fig

    data = asignaturas_superadas_media_abandono(
        curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
    )

    if not data:
        return fig

    df = pd.DataFrame(
        data, columns=["Alumno_id", "Abandono", "Nota_Media", "Asignaturas_Superadas"]
    )

    df["Abandono"] = (
        df["Abandono"]
        .str.strip()
        .str.lower()
        .replace({"si": "Abandona", "no": "No abandona"})
    )
    df["Personal"] = df["Alumno_id"].apply(lambda x: " (Yo)" if x == alumno_id else "")
    df["Key"] = df["Abandono"] + df["Personal"]

    colors = {
        "Abandona": "red",
        "No abandona": "blue",
        "Abandona (Yo)": "yellow",
        "No abandona (Yo)": "yellow",
    }

    for key, group in df.groupby("Key"):
        fig.add_trace(
            go.Scatter(
                x=group["Nota_Media"],
                y=group["Asignaturas_Superadas"],
                mode="markers",
                name=key,
                marker=dict(
                    size=12,
                    line=dict(width=2 if " (Yo)" in key else 1),
                    color=colors[key],
                ),
                opacity=1.0 if " (Yo)" in key else 0.8,
            )
```

```
        )

    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 19 del archivo callback_graph_calif_cual_comparativa.py.

```
def 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
    """
fig = go.Figure()

fig.update_layout(
title={
"text": "Calificaciones cualitativas general por curso académico",
            "x": 0.5,
        },
        barmode="stack",
        xaxis={"title": "Asignatura", "tickangle": 45},
        yaxis={"title": "N° Alumnos matriculados"},
        showlegend=True,
        legend={"title": "Calificación"},
        height=600,
    )

    if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
        asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
    except Exception as e:
        print("Error:", e)
        return fig

    data_universidad = universidad_alumno(alumno_id)

    if not data_universidad:
        return fig

    general_data = calif_cualitativa_comparativa(
        curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
    )
    student_data = calif_cualitativa_alumno_asignaturas(
        alumno_id, curso_academico, asignaturas_matriculadas, titulacion
    )

    if not general_data:
        return fig

    general_df = pd.DataFrame(
        general_data, columns=["Asignatura", "Calificacion", "Numero"]
    )
    student_df = pd.DataFrame(
        student_data, columns=["Asignatura", "Calificacion", "Numero"]
    )

    categories = ["Suspenso", "No presentado", "Aprobado", "Notable", "Sobresaliente"]

    color_mapping = {
        "Sobresaliente": "blue",
        "Notable": "green",
        "Aprobado": "orange",
        "Suspenso": "red",
        "No presentado": "gray",
    }

    general_pivot = general_df.pivot_table(
        index="Asignatura", columns="Calificacion", values="Numero", fill_value=0
    )
    general_pivot = general_pivot.reindex(columns=categories, fill_value=0)

    for category in categories:
        fig.add_trace(
            go.Bar(
                x=general_pivot.index,
                y=general_pivot[category],
                name=category,
                marker=dict(color=color_mapping[category]),
                opacity=0.7,
            )
        )
```

```
    for category in categories:
        y = [
            1 if (
                (subject in student_df['Asignatura'].values) and
                (student_df[student_df['Asignatura'] == subject]['Calificacion'].values[0] == category)
            ) else 0 for subject in general_pivot.index
        ]
        fig.add_trace(
            go.Bar(
                x=general_pivot.index,
                y=y,
                name=f"{category} (Yo)",
                marker=dict(color=color_mapping[category], line=dict(color='black', width=2)),
                opacity=1
            )
        )

    return fig
```

Gráfico de llamadas para esta función:

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

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_nota.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 15 del archivo callback_graph_calif_media_mi_nota.py.

```
00015 def update_graph_alumnado(curso_academico, asignaturas_matriculadas, alumno_id, titulacion):
00016     """
00017 Actualiza el gráfico de relación calificaciones del alumno con nota media general
00018 del perfil "Alumno" de la pestaña "Rendimiento académico general".
00019
00020     Args:
00021     curso_academico (list): Lista con los cursos académicos
00022     asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00023     alumno_id (str): Identificador del alumno
00024     titulacion (str): Titulación seleccionada
00025
00026     Returns:
00027     go.Figure: Figura con el gráfico
00028     """
00029 fig = go.Figure()
00030
00031 fig.update_layout(
00032 title={
00033 "text": "Relación calificaciones del alumno con nota media general",
00034             "x": 0.5,
00035         },
00036         xaxis={"title": "Asignatura", "tickangle": 45},
00037         yaxis={"title": "Nota"},
00038         barmode="group",
00039         height=600,
00040     )
00041
00042     if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
00043         return fig
00044
00045     try:
00046         curso_academico = list_to_tuple(curso_academico)
00047         asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00048     except Exception as e:
00049         print("Error:", e)
00050         return fig
00051
00052     data_universidad = universidad_alumno(alumno_id)
00053
00054     if not data_universidad:
00055         return fig
00056
00057     data = nota_media_general_mi_nota(
00058         curso_academico,
00059         asignaturas_matriculadas,
00060         alumno_id,
00061         titulacion,
00062         data_universidad[0][0],
00063     )
00064
00065     if not data:
00066         return fig
00067
00068     df = pd.DataFrame(data, columns=["Asignatura", "NotaMediaGeneral", "MiNota"])
00069
00070     fig.add_trace(
00071         go.Bar(
00072             x=df["Asignatura"],
00073             y=df["MiNota"],
00074             name="Mi nota",
00075             marker_color="blue",
00076             opacity=0.7,
00077         )
00078     )
00079
00080     fig.add_trace(
00081         go.Bar(
00082             x=df["Asignatura"],
00083             y=df["NotaMediaGeneral"],
00084             name="Nota media general",
```

```
            marker_color="grey",
            opacity=0.7,
        )
    )

    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 14 del archivo callback_graph_calif_cualitativa_alumnado.py.

```
00014 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00015     """
00016 Actualiza el gráfico de calificaciones cualitativas de las asignaturas matriculadas
00017 del perfil "Alumno" de la pestaña "Expediente académico personal".
00018
00019     Args:
00020     alumno_id (str): Identificador del alumno.
00021     curso_academico (list): Lista con los cursos académicos
00022     titulacion (str): Titulación seleccionada
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Calificaciones cualitativas", "x": 0.5},
00032         barmode="stack",
00033         xaxis={"title": "Curso académico"},
00034         yaxis={"title": "N° Asignaturas matriculadas"},
00035         showlegend=True,
00036         legend={"title": "Calificación"},
00037     )
00038
00039     if not (alumno_id and curso_academico and titulacion):
00040         return fig
00041
00042     try:
00043         curso_academico = list_to_tuple(curso_academico)
00044     except Exception as e:
00045         print("Error:", e)
00046         return fig
00047
00048     data = calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion)
00049
00050     if not data:
00051         return fig
00052
00053     df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "cantidad"])
00054     df_pivot = df.pivot_table(
00055         index="curso_academico", columns="calificacion", values="cantidad", fill_value=0
00056     )
00057
00058     categories = ["No presentado", "Suspenso", "Aprobado", "Notable", "Sobresaliente"]
00059     for category in categories:
00060         if category not in df_pivot.columns:
00061             df_pivot[category] = 0
00062
00063     df_pivot = df_pivot[categories]
00064     df_pivot = df_pivot.sort_index()
00065
00066     color_mapping = {
00067         "Sobresaliente": "blue",
00068         "Notable": "green",
00069         "Aprobado": "orange",
00070         "Suspenso": "red",
00071         "No presentado": "gray",
```

```
    }

    for category in categories:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot[category],
                name=category,
                marker_color=color_mapping[category],
                opacity=0.7,
            )
        )

    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 14 del archivo callback_graph_calif_numerica_alumnado.py.

```
00014 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00015     """
00016 Actualiza el gráfico de calificaciones cuantitativas de las asignaturas matriculadas
00017 del perfil "Alumno" de la pestaña "Expediente académico personal".
00018
00019     Args:
00020     alumno_id (str): Identificador del alumno.
00021     curso_academico (list): Lista con los cursos académicos
00022     titulacion (str): Titulación seleccionada
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Calificaciones cuantitativas", "x": 0.5},
00032         xaxis={"title": "Asignatura", "tickangle": 45},
00033         yaxis={"title": "Calificación"},
00034         height=600,
00035         showlegend=True,
00036         legend=dict(
00037             x=1,
00038             y=1,
00039             traceorder="normal",
00040             font=dict(
00041                 size=10,
00042             ),
00043             title="Asignaturas",
00044         ),
00045     )
00046
00047     if not (alumno_id and curso_academico and titulacion):
00048         return fig
00049
00050     try:
00051         curso_academico = list_to_tuple(curso_academico)
00052     except Exception as e:
00053         print("Error:", e)
00054         return fig
00055
00056     data = calif_numerica_asignatura(alumno_id, curso_academico, titulacion)
00057
00058     if not data:
00059         return fig
00060
00061     df = pd.DataFrame(data, columns=["Asignatura", "Calificacion"])
00062     colors = random_color(len(df))
00063
00064     for index, row in df.iterrows():
00065         fig.add_trace(
00066             go.Bar(
00067                 x=[row["Asignatura"]],
00068                 y=[row["Calificacion"]],
00069                 name=row["Asignatura"],
00070                 marker_color=colors[index],
00071                 opacity=0.7,
00072             )
00073         )
00074
00075     return fig
```

Gráfico de llamadas para esta función:

4.16. Referencia del Namespace callbacks.alumnado.graphs.personal.↩ 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 14 del archivo callback_graph_progreso_academico_alumnado.py.

```
00014 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00015     """
00016 Actualiza el gráfico de evolución del progreso académico del perfil "Alumno"
00017     de la pestaña "Expediente académico personal".
00018
00019     Args:
00020     alumno_id (str): Identificador del alumno.
00021     curso_academico (list): Lista con los cursos académicos
00022     titulacion (str): Titulación seleccionada
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Evolución del progreso académico", "x": 0.5},
00032         xaxis={"title": "Curso académico"},
00033         yaxis={"title": "Nº Asignaturas de superadas"},
00034         showlegend=False,
00035     )
00036
00037     if not (alumno_id and curso_academico and titulacion):
00038         return fig
00039
00040     try:
00041         curso_academico = list_to_tuple(curso_academico)
00042     except Exception as e:
00043         print("Error:", e)
00044         return fig
00045
00046     data = asignaturas_superadas(alumno_id, curso_academico, titulacion)
00047
```

```
    if not data:
        return fig

    data = pd.DataFrame(data)
    academic_years = data["curso_academico"]
    subjects_passed = data["n_asig_superadas"]

    cumulative_passed = []
    cumulative_total = 0
    for count in subjects_passed:
        cumulative_total += count
        cumulative_passed.append(cumulative_total)

    fig.add_trace(
        go.Bar(x=academic_years, y=cumulative_passed, marker_color="blue", opacity=0.7)
    )

    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 14 del archivo callback_graph_tasa_rendimiento_alumnado.py.

```
00014 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00015     """
00016 Actualiza el gráfico de tasa de rendimiento del perfil "Alumno"
00017     de la pestaña "Expediente académico personal".
00018
00019     Args:
00020     alumno_id (str): Identificador del alumno.
00021     curso_academico (list): Lista con los cursos académicos
00022     titulacion (str): Titulación seleccionada
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026
00027     """
00028
00029 fig = go.Figure()
00030
00031 fig.update_layout(
00032 title={"text": "Tasa de rendimiento por curso académico", "x": 0.5},
00033         xaxis={"title": "Tasa de rendimiento (%)"},
00034         yaxis={"title": "Curso académico"},
00035         showlegend=False,
00036     )
00037
00038     if not (alumno_id and curso_academico and titulacion):
00039         return fig
00040
00041     try:
00042         curso_academico = list_to_tuple(curso_academico)
00043     except Exception as e:
00044         print("Error:", e)
00045         return fig
00046
00047     data = asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion)
00048
00049     if not data:
00050         return fig
00051
00052     df = pd.DataFrame(data, columns=["curso_academico", "matriculadas", "superadas"])
00053     df["tasa_rendimiento"] = (df["superadas"] / df["matriculadas"]) * 100
00054
00055     fig.add_trace(
00056         go.Bar(
00057             x=df["tasa_rendimiento"],
00058             y=df["curso_academico"],
00059             orientation="h",
00060             marker_color="blue",
00061             opacity=0.7,
00062             width=0.7,
00063         )
00064     )
00065
00066     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 63 del archivo callback_resumen_alumnado.py.

```
00063 def not_data():
00064     return html.Div(
00065         [
00066             html.H2("Resumen"),
00067             html.P("Universidad:", className="resumen-label"),
00068             html.P("No disponible"),
00069             html.P("Titulación:", className="resumen-label"),
00070             html.P("No disponible"),
00071             html.P("Alumno:", className="resumen-label"),
00072             html.P("No disponible"),
00073             html.P("Nota Media:", className="resumen-label"),
00074             html.P("No disponible"),
00075             html.Hr(),
00076         ]
00077     )
```

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 11 del archivo callback_resumen_alumnado.py.

```
00011 def update_resumen_alumnado(alumno_id, titulacion):
00012     """
00013 Actualiza el resumen del perfil "Alumno".
00014
00015     Args:
00016     alumno_id (str): Identificador del alumno.
00017     titulacion (str): Titulación seleccionada
00018
00019     Returns:
00020     list: Componentes con el resumen del perfil "Alumno"
00021     """
00022 if not (alumno_id and titulacion):
00023         return not_data()
00024
00025     data = resumen_alumno(alumno_id, titulacion)
00026
00027     data = pd.DataFrame(data, columns=["id_alumno", "universidad", "titulacion", "abandona"])
00028
00029     if data.empty:
00030         return not_data()
00031
00032     componentes = [
00033         html.H2("Resumen"),
00034         html.P("Universidad:", className="resumen-label"),
00035         html.P(data["universidad"].iloc[0]),
00036         html.P("Titulación:", className="resumen-label"),
00037         html.P(data["titulacion"].iloc[0]),
00038         html.P("Alumno:", className="resumen-label"),
00039         html.P(alumno_id),
00040         html.P("Nota Media:", className="resumen-label"),
00041         html.P(nota_media_alumno_titulacion(alumno_id, titulacion))
00042     ]
00043
00044     if data["abandona"].iloc[0] == 'no':
00045         componentes.append(html.P("Estado:", className="resumen-label"))
00046         componentes.append(html.P("Activo"))
00047         componentes.append(html.P("Probabilidad de abandono:", className="resumen-label"))
00048         data_model = load_model()
00049
00050         probabilidad_abandono = data_model[data_model['id'] ==
      alumno_id]['probabilidad_abandono'].values[0]
00051         componentes.append(html.P(f"{probabilidad_abandono:.2%}"))
00052     else:
00053         componentes.append(html.P("Estado:", className="resumen-label"))
00054         componentes.append(html.P("Abandona"))
00055
00056
00057     componentes.append(html.Hr())
00058
00059     return html.Div(componentes)
00060
00061
00062
```

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 12 del archivo callback_select_alumnado.py.

```
00012 def store_selected_alumnado(selected_value, stored_value):
00013     """
00014 Almacena el valor seleccionado en el dropdown de alumnos.
00015
00016 Args:
00017 selected_value (str): Valor seleccionado en el dropdown
00018 stored_value (str): Valor almacenado
00019
00020 Returns:
00021 str: Valor seleccionado
00022 list: Opciones del dropdown
00023 str: Valor almacenado
00024 """
00025 data = alumnos_all()
00026
00027 if not data:
00028         return None, [], None
00029
00030     opciones_dropdown = [{"label": alumno[0], "value": alumno[0]} for alumno in data]
00031
00032     if selected_value is None and stored_value in [
00033         op["value"] for op in opciones_dropdown
00034     ]:
00035         return stored_value, opciones_dropdown, stored_value
00036
00037     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 19 del archivo callback_tabs_alumnado.py.

```
00019 def render_content(tab, selected_alumnado):
00020     """
00021 Renderiza el contenido de las pestañas del dashboard del alumnado.
00022
00023 Args:
00024 tab (str): Pestaña seleccionada
00025 selected_alumnado (str): Alumno seleccionado
00026
00027 Returns:
00028 list: Componentes de la pestaña seleccionada
00029 """
00030 if tab == 'expediente-personal-tab':
00031         return html.Div([
00032             select_alumnado(),
00033             html.H2("Dashboard Alumnado", style={'textAlign': 'center'}),
00034             html.Div([
00035                 sidebar([
00036                     resumen_alumnado(),
00037                     filters([
00038                         filter_titulacion_alumnado(),
00039                         filter_curso_academico_alumnado()
00040                         ])
00041                     ]),
00042                 graphs_personal_alumnado()
00043             ], className='content-layout-dashboard')
00044         ])
00045     elif tab == 'rendimiento-academico-tab':
00046         return html.Div([
00047             html.H2("Dashboard Alumnado", style={'textAlign': 'center'}),
00048             html.Div([
00049                 sidebar([
00050                     resumen_alumnado(),
00051                     filters([
00052                         filter_titulacion_alumnado(),
00053                         filter_curso_academico_alumnado(),
00054                         filter_asignaturas_matri_alumnado()
00055                         ])
00056                     ]),
00057                 graphs_general_alumnado()
00058             ], className='content-layout-dashboard')
```

```
        ])
    elif tab == 'recomendador-tab':
        return html.Div([
            recomendador_alumnado()
        ])

```

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 sidebar_collapse (n, is_open)

4.23.1. Documentación de las funciones

4.23.1.1. sidebar_collapse()

```
def callbacks.common.callback_sidebar_collapse.sidebar_collapse (
              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 9 del archivo callback_sidebar_collapse.py.

```
00009 def sidebar_collapse(n, is_open):
00010     """
00011 Callback que permite colapsar y descolapsar el sidebar.
00012
00013 Args:
00014 n (int): Número de clicks
00015 is_open (bool): Estado del sidebar
00016
00017 Returns:
00018 bool: Estado del sidebar
00019 """
00020 if n:
00021         return not is_open
00022     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 10 del archivo callback_update_layout.py.

```
00010 def update_layout(role):
00011     """
00012 Actualiza el layout de la aplicación según el rol del usuario.
00013
00014 Args:
00015 role (str): Rol del usuario
00016
00017 Returns:
00018 str: Layout correspondiente al rol del usuario
00019 """
00020 if role == 'Alumno':
00021         return alumno_layout()
00022     elif role == 'Docente':
00023         return docente_layout()
00024     elif role == 'Gestor':
00025         return gestor_layout()
00026     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 10 del archivo callback_user_role.py.

```
00010 def initialize_dropdown(ts, stored_role):
00011     """
00012 Inicializa el dropdown con el rol almacenado.
00013
00014 Args:
00015 ts (float): Timestamp de la última modificación del store
00016 stored_role (str): Rol almacenado
00017 """
00018 return stored_role if ts is not None else "Alumno"
00019
00020
00021 @callback(
00022     Output("store-role", "data"),
00023     Input("dropdown_role", "value"),
00024     State("store-role", "data"),
00025     prevent_initial_call=True,
00026 )
```

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 27 del archivo callback_user_role.py.

```
00027 def update_role(new_role, current_role):
00028     """
00029 Actualiza el rol almacenado en el store.
00030
00031 Args:
00032 new_role (str): Nuevo rol seleccionado
00033 current_role (str): Rol actual almacenado
00034
00035 Returns:
00036 str: Rol almacenado
00037 """
00038 if new_role != current_role:
00039         return new_role
00040     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 8 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 13 del archivo callback_filter_all_asignaturas_titulacion_docente.py.

```
00013 def update_filter_asignaturas_docente(titulacion, curso_academico, n_clicks, existing_options):
00014     """
00015 Actualiza las opciones del dropdown con todas las asignaturas de la titulación seleccionada y
00016 gestiona el evento del botón que selecciona todas las asignaturas.
00017
00018 Args:
00019 titulacion (str): Titulación seleccionada
00020 curso_academico (list): Cursos académicos seleccionados
00021 n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00022     existing_options (list): Opciones actuales del dropdown
00023
00024     Returns:
00025     list: Opciones del dropdown
00026     list: Valor seleccionado
00027     """
00028 ctx = callback_context
00029 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00030
00031     if trigger_id == "select-all-asignaturas-titulacion-docente" and n_clicks > 0:
00032         return existing_options, [option["value"] for option in existing_options]
00033
00034     if not (titulacion and curso_academico):
00035         return [], None
00036
00037     data = asignaturas_actas_titulacion(titulacion, curso_academico)
00038
00039     if not data:
00040         return [], None
00041
00042     opciones_dropdown = [
00043         {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
00044     ]
00045     value = (
00046         [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
00047     )
00048
00049     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 10 del archivo callback_filter_all_curso_academico.py.

```
00010 def update_filter_all_cursos_academicos_docente(titulacion):
00011     """
00012 Actualiza las opciones del dropdown con todos los cursos académicos de la titulación seleccionada.
00013
00014 Args:
00015 titulacion (str): Titulación seleccionada
00016
00017 Returns:
00018 list: Opciones del dropdown
00019 list: Valor seleccionado
00020 """
00021
00022 if not titulacion:
00023         return [], None
00024
00025     data = curso_academico_actas_titulacion(titulacion)
00026
00027     if not data:
00028         return [], None
00029
00030     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00031     value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00032
00033     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 11 del archivo callback_filter_asignaturas_docente.py.

```
00011 def update_filter_asignaturas_docente(docente_id, titulacion):
00012     """
00013 Actualiza las opciones del dropdown con las asignaturas del docente seleccionado.
00014
00015 Args:
00016 docente_id (str): ID del docente
00017 titulacion (str): Titulación seleccionada
00018
00019 Returns:
00020 list: Opciones del dropdown
00021 str: Valor seleccionado
00022 """
00023
00024 if not (docente_id and titulacion):
00025         return [], None
00026
00027     data = asignaturas_docente(docente_id, titulacion)
00028
00029     if not data:
00030         return [], None
00031
00032     opciones_dropdown = [
00033         {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
00034     ]
00035     value = opciones_dropdown[0]["value"] if opciones_dropdown else None
00036
00037     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 14 del archivo callback_filter_curso_academico_docente.py.

```
00014 def update_filter_curso_academico_docente(docente_id, asignatura, n_clicks, existing_options):
00015     """
00016 Actualiza las opciones del dropdown con los cursos académicos del docente seleccionado y
00017 gestiona el evento del botón que selecciona todos los cursos académicos.
00018
00019 Args:
00020 docente_id (str): ID del docente
00021 asignatura (str): Asignatura seleccionada
00022 n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00023     existing_options (list): Opciones actuales del dropdown
00024
00025     Returns:
00026     list: Opciones del dropdown
00027     list: Valor seleccionado
00028
00029     """
00030 ctx = callback_context
00031 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00032
00033     if trigger_id == "select-all-cursos-academicos" and n_clicks > 0:
00034         return existing_options, [option["value"] for option in existing_options]
00035
00036     if not (docente_id and asignatura):
00037         return [], None
00038
00039     data = curso_academico_docente(docente_id, asignatura)
00040
00041     if not data:
00042         return [], None
00043
```

```
    opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
    value = [option["value"] for option in opciones_dropdown]

    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 12 del archivo callback_filter_titulacion_docente.py.

```
12 def update_filter_titulacion_docente(docente_id, selected_value, stored_titulacion):
13     """
14 Actualiza las opciones del dropdown con las titulaciones del docente seleccionado.
15
16 Args:
17 docente_id (str): ID del docente
18 selected_value (str): Valor seleccionado
19 stored_titulacion (str): Titulación seleccionada
20
21 Returns:
```

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

"""
if not docente_id:
        return [], None, None

    data = titulacion_docente(docente_id)

    if not data:
        return [], None, None

    opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]

    if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
        return opciones_dropdown, stored_titulacion, stored_titulacion

    return opciones_dropdown, selected_value, selected_value


```

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 14 del archivo callback_graph_all_calif_cualitativa_docente.py.

```
00014 def update_graph_docente(titulacion, curso_academico, asignatura):
00015     """
00016 Actualiza el gráfico de calificaciones cualitativas de la titulación
00017 seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00018
00019     Args:
00020     titulacion (str): Titulación seleccionada
00021     curso_academico (list): Lista con los cursos académicos
00022     asignatura (list): Lista con las asignaturas seleccionadas
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 barmode="stack",
00032         title={"text": "Calificaciones cualitativas de la titulación", "x": 0.5},
00033         xaxis={"title": "Asignaturas", "tickangle": 45},
00034         yaxis={"title": "N° Alumnos matriculados"},
00035         legend=dict(
00036             x=1,
00037             y=1,
00038             traceorder="normal",
00039             font=dict(
00040                 size=10
00041             ),
00042             title="Calificaciones"
00043         ),
00044     )
00045
00046     if not (curso_academico and asignatura):
00047         return fig
00048
00049     try:
00050         asignatura = list_to_tuple(asignatura)
00051     except Exception as e:
00052         return fig
00053
00054     data = calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignatura)
00055
00056     if not data:
00057         return fig
00058
00059     df = pd.DataFrame(
00060         data, columns=["titulacion", "asignatura", "calificación", "n_alumnos"]
00061     )
00062
00063     colors_mapping = {
00064         "Sobresaliente": "blue",
00065         "Notable": "green",
00066         "Aprobado": "orange",
00067         "Suspenso": "red",
00068         "No presentado": "gray",
00069     }
00070
00071     for calif, color in colors_mapping.items():
```

```
        df_calif = df[df["calificación"] == calif]
        fig.add_trace(
            go.Bar(
                x=df_calif["asignatura"],
                y=df_calif["n_alumnos"],
                name=calif,
                marker_color=color,
                opacity=0.7,
            )
        )

    fig.update_xaxes(categoryorder="array", categoryarray=asignatura)

    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 14 del archivo callback_graph_all_calif_media_docente.py.

```
00014 def update_graph_docente(titulacion, curso_academico, asignatura):
00015     """
00016 Actualiza el gráfico de nota media por asignatura de la titulación
00017 seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00018
00019     Args:
00020     titulacion (str): Titulación seleccionada
00021     curso_academico (list): Lista con los cursos académicos
00022     asignatura (list): Lista con las asignaturas seleccionadas
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Nota media por asignatura de la titulación", "x": 0.5},
00032         xaxis_title="Asignaturas",
00033         yaxis_title="Nota media",
00034     )
00035
00036     if not (curso_academico and asignatura):
00037         return fig
00038
00039     try:
00040         asignatura = list_to_tuple(asignatura)
00041     except Exception as e:
00042         return fig
00043
00044     data = calif_media_asignaturas(titulacion, curso_academico, asignatura)
00045
00046     if not data:
00047         return fig
00048
00049     df = pd.DataFrame(data, columns=["asignatura", "media_calif"])
00050
00051     fig.add_trace(
00052         go.Bar(
00053             x=df["asignatura"], y=df["media_calif"], marker_color="blue", opacity=0.7
00054         )
00055     )
00056
00057     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 14 del archivo callback_graph_alu_genero_docente.py.

```
00014 def update_graph_docente(asignaturas, curso_academico, docente_id):
00015     """
00016 Actualiza el gráfico de evolución de alumnos matriculados por género
00017 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00018
00019     Args:
00020     asignaturas (list): Lista con las asignaturas seleccionadas
00021     curso_academico (list): Lista con los cursos académicos
00022     docente_id (str): Identificador del docente
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 barmode="stack",
00032         title={"text": "Evolución alumnos matriculados por género", "x": 0.5},
00033         xaxis={"title": "Curso académico"},
00034         yaxis={"title": "N° Alumnos matriculados"},
00035         legend={"title": "Género"},
00036     )
00037
00038     if not (asignaturas and curso_academico and docente_id):
00039         return fig
00040
00041     try:
00042         curso_academico = list_to_tuple(curso_academico)
00043     except Exception as e:
00044         return fig
00045
00046     data = alumnos_genero_docente(docente_id, asignaturas, curso_academico)
00047
00048     if not data:
00049         return fig
```

```
    df = pd.DataFrame(data, columns=["curso_academico", "genero", "cantidad"])

    df_pivot = df.pivot(
        index="curso_academico", columns="genero", values="cantidad"
    ).fillna(0)

    colores = {"Femenino": "red", "Masculino": "blue"}

    for genero in df_pivot.columns:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot[genero],
                name="Mujeres" if genero == "Femenino" else "Hombres",
                marker_color=colores[genero],
                opacity=0.7,
            )
        )

    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 14 del archivo callback_graph_alu_media_docente.py.

```
00014 def update_graph_docente(asignaturas, curso_academico, docente_id):
00015     """
00016 Actualiza el gráfico de evolución de la nota media por asignatura
00017 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00018
00019     Args:
00020     asignaturas (list): Lista con las asignaturas seleccionadas
00021     curso_academico (list): Lista con los cursos académicos
00022     docente_id (str): Identificador del docente
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Evolución nota media por asignatura", "x": 0.5},
00032         xaxis_title="Curso académico",
00033         yaxis_title="Nota media",
00034         xaxis=dict(type="category"),
00035     )
00036
00037     if not (asignaturas and curso_academico and docente_id):
00038         return fig
00039
00040     try:
00041         curso_academico = list_to_tuple(curso_academico)
00042         asignaturas = list_to_tuple(asignaturas)
00043     except Exception as e:
00044         return fig
00045
00046     data = alumnos_nota_media_docente(asignaturas, curso_academico)
00047
00048     if not data:
00049         return fig
00050
00051     df = pd.DataFrame(data, columns=["curso_academico", "asignatura", "nota_media"])
00052
00053     fig.add_trace(
00054         go.Bar(
00055             x=df["curso_academico"],
00056             y=df["nota_media"],
00057             name="Nota media",
00058             marker=dict(color="blue"),
00059             opacity=0.7,
00060         )
00061     )
00062
00063     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 14 del archivo callback_graph_alu_nota_cualitativa_docente.py.

```
00014 def update_graph_docente(asignaturas, curso_academico, docente_id):
00015     """
00016 Actualiza el gráfico de evolución de calificaciones cualitativas
00017 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00018
00019     Args:
00020     asignaturas (list): Lista con las asignaturas seleccionadas
00021     curso_academico (list): Lista con los cursos académicos
00022     docente_id (str): Identificador del docente
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 barmode="stack",
00032         title={"text": "Evolución calificaciones cualitativas", "x": 0.5},
00033         xaxis={"title": "Curso académico"},
00034         yaxis={"title": "Nº Alumnos matriculados"},
00035         legend_title_text="Calificación",
00036     )
00037
00038     if not (asignaturas and curso_academico and docente_id):
00039         return fig
00040
00041     try:
00042         curso_academico = list_to_tuple(curso_academico)
00043         asignaturas = list_to_tuple(asignaturas)
00044     except Exception as e:
00045         return fig
00046
00047     data = alumnos_nota_cualitativa_docente(asignaturas, curso_academico)
00048     if not data:
00049         return fig
00050
00051     df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "n_alumnos"])
00052
00053     # Pivotear el DataFrame para obtener las cantidades por calificación en columnas separadas
00054     df_pivot = df.pivot(
00055         index="curso_academico", columns="calificacion", values="n_alumnos"
00056     ).fillna(0)
00057     catagories = ["No presentado", "Suspenso", "Aprobado", "Notable", "Sobresaliente"]
00058
00059     for calif in catagories:
00060         if calif not in df_pivot:
00061             df_pivot[calif] = 0
00062
```

```
    df_pivot = df_pivot[catagories]

    colors = {
        "Sobresaliente": "blue",
        "Notable": "green",
        "Aprobado": "orange",
        "Suspenso": "red",
        "No presentado": "gray",
    }

    for calif in catagories:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot[calif],
                name=calif,
                marker=dict(color=colors[calif]),
                opacity=0.7,
            )
        )

    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 14 del archivo callback_graph_alu_repetidores_nuevos_docente.py.

```
00014 def update_graph_docente(asignaturas, curso_academico, docente_id):
00015     """
00016 Actualiza el gráfico de evolución de alumnos de nuevo ingreso y repetidores
00017 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00018
00019     Args:
00020     asignaturas (list): Lista con las asignaturas seleccionadas
00021     curso_academico (list): Lista con los cursos académicos
00022     docente_id (str): Identificador del docente
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 barmode="stack",
00032         title={"text": "Evolución alumnos de nuevo ingreso y repetidores", "x": 0.5},
00033         xaxis={"title": "Curso académico"},
00034         yaxis={"title": "N° Alumnos matriculados"},
00035         showlegend=True,
00036         legend={"orientation": "h", "x": 0, "y": 1.1},
00037     )
00038
00039     if not (asignaturas and curso_academico and docente_id):
00040         return fig
00041
00042     try:
00043         curso_academico = list_to_tuple(curso_academico)
00044     except Exception as e:
00045         return fig
00046
00047     data = alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas)
00048
00049     if not data:
00050         return fig
00051
00052     df = pd.DataFrame(
00053         data,
00054         columns=["curso_academico", "alumnos_repetidores", "alumnos_nuevo_ingreso"],
00055     )
00056
00057     fig.add_trace(
00058         go.Bar(
00059             x=df["curso_academico"],
00060             y=df["alumnos_repetidores"],
00061             name="Alumnos repetidores",
00062             marker_color="red",
00063             opacity=0.7,
00064         )
00065     )
00066
00067     fig.add_trace(
00068         go.Bar(
00069             x=df["curso_academico"],
00070             y=df["alumnos_nuevo_ingreso"],
00071             name="Alumnos de primera matrícula",
00072             marker_color="green",
00073             opacity=0.7,
00074         )
00075     )
00076
00077     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 43 del archivo callback_resumen_docente.py.

```
00043 def not_data():
00044     return html.Div(
00045         [
00046             html.H2("Resumen"),
00047             html.P("Universidad:", className="resumen-label"),
00048             html.P("No disponible"),
00049             html.P("Titulación:", className="resumen-label"),
00050             html.P("No disponible"),
00051             html.P("Docente:", className="resumen-label"),
00052             html.P("No disponible"),
00053             html.Hr(),
00054         ]
00055     )
```

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 10 del archivo callback_resumen_docente.py.

```
00010 def update_resumen_docente(docente_id, titulacion):
00011     """
00012 Actualiza el resumen del docente
00013
00014 Args:
00015 docente_id (str): ID del docente
00016 titulacion (str): Titulación seleccionada
00017
00018 Returns:
00019 html.Div: Layout del resumen del docente
00020 """
00021 if not (docente_id and titulacion):
00022         return not_data()
00023
00024     data = universidades_docente(docente_id)
00025
00026     if not data:
00027         return not_data()
00028
00029     return html.Div(
00030         [
00031             html.H2("Resumen"),
00032             html.P("Universidad:", className="resumen-label"),
00033             html.P(data[0][1]),
00034             html.P("Titulación:", className="resumen-label"),
00035             html.P(titulacion),
00036             html.P("Docente:", className="resumen-label"),
00037             html.P(docente_id),
00038             html.Hr(),
00039         ]
00040     )
00041
00042
```

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 12 del archivo callback_select_docente.py.

```
00012 def store_selected_docente(selected_value, stored_value):
00013     """
00014 Almacena el docente seleccionado en el store.
00015
00016 Args:
00017 selected_value (str): Valor seleccionado
00018 stored_value (str): Valor almacenado
00019
00020 Returns:
00021 str: Valor seleccionado
00022 list: Opciones del dropdown
00023 str: Valor almacenado
00024 """
00025
00026 data = docentes_all()
00027
00028 if not data:
00029         return None, [], None
00030
00031     opciones_dropdown = [{"label": docente[0], "value": docente[0]} for docente in data]
00032
00033     if selected_value is None and stored_value in [
00034         op["value"] for op in opciones_dropdown
00035     ]:
00036         return stored_value, opciones_dropdown, stored_value
00037
00038     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:
list: Componentes de la pestaña seleccionada
```

Definición en la línea 21 del archivo callback_tabs_docente.py.

```
00021 def render_content(tab, selected_docente):
00022     """
00023 Renderiza el contenido de las pestañas del dashboard del docente.
00024
00025 Args:
00026 tab (str): Pestaña seleccionada
00027 selected_docente (str): Docente seleccionado
00028
00029 Returns:
00030 list: Componentes de la pestaña seleccionada
00031 """
00032 if tab == 'rendimiento-academico-asignatura-tab':
00033         return html.Div([
00034             select_docente(),
00035             html.H2("Dashboard Docente", style={'textAlign': 'center'}),
00036             html.Div([
00037                 sidebar([
00038                   resumen_docente(),
00039                   filters([
00040                     filter_titulacion_docente(),
00041                     filter_asignaturas_docente(),
00042                     filter_curso_academico_docente()
00043                   ]),
00044                 ]),
00045                 graphs_personal_docente()
00046             ], className='content-layout-dashboard')
00047         ])
00048     elif tab == 'rendimiento-academico-tab':
00049         return html.Div([
00050             html.H2("Dashboard Docente", style={'textAlign': 'center'}),
00051             html.Div([
00052                 sidebar([
00053                   resumen_docente(),
00054                   filters([
00055                     filter_titulacion_docente(),
00056                     filter_all_curso_academico(),
00057                     filter_all_asignaturas_titulacion_docente()
00058                   ]),
00059                 ]),
00060                 graphs_general_docente()
00061             ], className='content-layout-dashboard')
00062         ])
00063     elif tab == 'recomendaciones-tab':
00064         return html.Div([
00065             recomendador_docente()
00066         ])
```

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 12 del archivo callback_filter_all_curso_academico_gestor.py.

```
00012 def update_filter_all_curso_academico_gestor(gestor_id, n_clicks, existing_options):
00013     """
00014 Actualiza las opciones del filtro de todos los cursos académicos del perfil "Gestor"
00015     en la pestaña "Riesgo académico".
00016
00017     Args:
00018     gestor_id (str): ID del gestor seleccionado
00019     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00020     existing_options (list): Opciones actuales del filtro de curso académico
00021
00022     Returns:
00023     list: Opciones del dropdown
00024     list: Valores seleccionados
00025     """
00026
00027 ctx = callback_context
00028 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00029
00030     if trigger_id == "select-all-curso-academico-button" and n_clicks > 0:
00031         return existing_options, [option["value"] for option in existing_options]
00032
00033     if not gestor_id:
00034         return [], None
00035
00036     cod_universidad = universidades_gestor(gestor_id)
00037
00038     if not cod_universidad:
00039         return [], None
00040
00041     data = curso_academico_universidad(cod_universidad[0][0])
00042
00043     if not data:
00044         return [], None
```

```
00045
00046     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00047     value = (
00048         [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
00049     )
00050
00051     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 10 del archivo callback_filter_curso_academico_gestor.py.

```
00010 def update_filter_curso_academico_gestor(gestor_id):
00011     """
00012 Actualiza las opciones del filtro de curso académico del perfil "Gestor"
00013     en la pestaña "Indicadores académicos".
00014
00015     Args:
00016     gestor_id (str): ID del gestor seleccionado
00017
```

```
    Returns:
    list: Opciones del dropdown
    str: Valor seleccionado
    """

if not gestor_id:
        return [], None

    cod_universidad = universidades_gestor(gestor_id)

    if not cod_universidad:
        return [], None

    data = curso_academico_universidad(cod_universidad[0][0])

    if not data:
        return [], None

    opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
    value = opciones_dropdown[0]["value"] if opciones_dropdown else None

    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" em
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 13 del archivo callback_filter_titulaciones_gestor.py.

```
00013 def update_filter_titulaciones_gestor(docente_id, curso_academico, n_clicks, existing_options):
00014     """
00015 Actualiza las opciones del filtro de titulaciones en base a la universidad y curso académico
      seleccionados.
00016 También gestiona el evento que permite seleccionar todas las opciones del filtro del perfil "Gestor"
      em
00017     la pestaña "Indicadores académicos".
00018
00019     Args:
00020     docente_id (str): ID del docente seleccionado
00021     curso_academico (str): Curso académico seleccionado
00022     n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00023     existing_options (list): Opciones actuales del filtro de titulaciones
00024
00025     Returns:
00026     list: Opciones del dropdown
00027     list: Valores seleccionados
00028     """
00029 ctx = callback_context
00030 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00031
00032     if trigger_id == "select-all-titulaciones-gestor" and n_clicks > 0:
00033         return existing_options, [option["value"] for option in existing_options]
00034
00035     if not (docente_id and curso_academico):
00036         return [], None
00037
00038     cod_universidad = universidades_gestor(docente_id)
00039
00040     if not cod_universidad:
00041         return [], None
00042
00043     data = titulaciones_universidad_gestor(cod_universidad[0][0], curso_academico)
00044
00045     if not data:
00046         return [], None
00047
00048     opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
00049     value = (
00050         [option["value"] for option in opciones_dropdown] if opciones_dropdown else None
00051     )
00052
00053     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 151 del archivo callback_graph_egresados_genero_gestor.py.

```
00151 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00152     """
00153 Genera un archivo CSV descargable con los datos de alumnos egresados por género
00154 y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00155
00156     Args:
00157     btn (int): Número de clicks en el botón
00158     gestor_id (str): ID del gestor seleccionado
00159     curso_academico (list): Lista con los cursos académicos
00160     titulaciones (list): Lista con las titulaciones seleccionadas
00161
00162     Returns:
00163     str: Enlace al archivo CSV
00164
00165     """
00166
00167 if not btn:
00168         return ""
00169
00170     df = get_data(gestor_id, curso_academico, titulaciones)
00171
00172     if df.empty:
00173         return ""
00174
00175     csv_string = df.to_csv(index=False, encoding="utf-8")
00176     csv_string = "data:text/csv;charset=utf-8," + csv_string
00177     return csv_string
00178
00179
```

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 180 del archivo callback_graph_egresados_genero_gestor.py.

```
00180 def get_data(gestor_id, curso_academico, titulaciones):
00181     """
00182 Obtiene los datos de la base de datos.
00183
00184 Args:
00185 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
"""
empty = pd.DataFrame()

if not gestor_id or not curso_academico or not titulaciones:
        return empty

    try:
        titulaciones = list_to_tuple(titulaciones)
    except Exception as e:
        return empty

    data_universidad = universidades_gestor(gestor_id)
    if not data_universidad:
        return empty

    data = alumnos_egresados_genero_titulacion(
        data_universidad[0][0], curso_academico, titulaciones
    )

    if not data:
        return empty

    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 )
```

```
Alterna 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 89 del archivo callback_graph_egresados_genero_gestor.py.

```
00089 def toggle_modal(btn, is_open):
00090     """
00091 Alterna la visibilidad del modal con los datos de alumnos egresados por género
00092 y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00093
00094     Args:
00095     btn (int): Número de clicks en el botón
00096     is_open (bool): Estado actual del modal
00097
00098     Returns:
00099     bool: Nuevo estado del modal
00100
00101     """
00102
00103 if btn:
00104         return not is_open
00105     return is_open
00106
00107
00108 @callback(
00109     Output("table-container-egresados-genero", "children"),
00110     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00111     Input("selected-gestor-store", "data"),
00112     Input("curso-academico-gestor", "value"),
00113     Input("titulaciones-gestor", "value"),
00114 )
```

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 15 del archivo callback_graph_egresados_genero_gestor.py.

```
00015 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00016     """
00017 Actualiza el gráfico de alumnos egresados por género y titulación
00018 del perfil "Gestor" de la pestaña "Indicadores académicos".
00019
00020     Args:
00021     gestor_id (str): ID del gestor seleccionado
00022     curso_academico (list): Lista con los cursos académicos
00023     titulaciones (list): Lista con las titulaciones seleccionadas
00024
```

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

    """
fig = go.Figure()

fig.update_layout(
barmode="stack",
        title={"text": "Alumnos egresados por género y titulación", "x": 0.5},
        xaxis=dict(title="Titulaciones"),
        yaxis=dict(title="N° Alumnos"),
        showlegend=True,
        legend={"title": "Género"},
    )

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return fig

    def map_genero(genero):
        if "masculin" in genero.lower():
            return "Hombres"
        elif "fem" in genero.lower():
            return "Mujeres"
        return genero

    df["genero"] = df["genero"].map(map_genero)

    # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
    df_pivot = df.pivot_table(
        index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
    ).fillna(0)

    if "Hombres" in df_pivot.columns:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot["Hombres"],
                name="Hombres",
                marker_color="blue",
                opacity=0.7,
            )
        )

    if "Mujeres" in df_pivot.columns:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot["Mujeres"],
                name="Mujeres",
                marker_color="red",
                opacity=0.7,
            )
        )

    return fig


@callback(
    Output("modal-egresados-genero", "is_open"),
    Input("btn-ver-datos-egresados-genero", "n_clicks"),
    State("modal-egresados-genero", "is_open"),
)
```

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 115 del archivo callback_graph_egresados_genero_gestor.py.

```
00115 def update_table(btn, gestor_id, curso_academico, titulaciones):
00116     """
00117 Actualiza la tabla con los datos de alumnos egresados por género y titulación
00118 del perfil "Gestor" de la pestaña "Indicadores académicos".
00119
00120     Args:
00121     btn (int): Número de clicks en el botón
00122     gestor_id (str): ID del gestor seleccionado
00123     curso_academico (list): Lista con los cursos académicos
00124     titulaciones (list): Lista con las titulaciones seleccionadas
00125
00126     Returns:
00127     dbc.Table: Tabla con los datos
00128
00129     """
00130
00131 if not btn:
00132         return ""
00133
00134     df = get_data(gestor_id, curso_academico, titulaciones)
00135
00136     if df.empty:
00137         return dbc.Alert("No hay datos disponibles", color="info")
00138
00139     return dbc.Table.from_dataframe(
00140         df.head(50), striped=True, bordered=True, hover=True
00141     )
00142
00143
00144 @callback(
00145     Output("btn-descargar-egresados-genero", "href"),
00146     Input("btn-ver-datos-egresados-genero", "n_clicks"),
00147     State("selected-gestor-store", "data"),
00148     State("curso-academico-gestor", "value"),
00149     State("titulaciones-gestor", "value"),
00150 )
```

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 121 del archivo callback_graph_egresados_nacionalidad_gestor.py.

```
00121 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00122     """
00123 Genera un archivo CSV descargable con los datos de alumnos egresados por nacionalidad
00124 y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00125
00126     Args:
00127     btn (int): Número de clicks en el botón
00128     gestor_id (str): ID del gestor seleccionado
00129     curso_academico (list): Lista con los cursos académicos
00130     titulaciones (list): Lista con las titulaciones seleccionadas
00131
00132     Returns:
00133     str: Enlace al archivo CSV
00134     """
00135
00136 if not btn:
00137         return ""
00138
00139     df = get_data(gestor_id, curso_academico, titulaciones)
00140
00141     if df.empty:
00142         return ""
00143
00144     csv_string = df.to_csv(index=False, encoding="utf-8")
00145     csv_string = "data:text/csv;charset=utf-8," + csv_string
```

```
    return csv_string


```

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 149 del archivo callback_graph_egresados_nacionalidad_gestor.py.

```
00149 def get_data(gestor_id, curso_academico, titulaciones):
00150     """
00151 Obtiene los datos de alumnos egresados por nacionalidad y titulación
00152 del perfil "Gestor" de la pestaña "Indicadores académicos".
00153
00154     Args:
00155     gestor_id (str): ID del gestor seleccionado
00156     curso_academico (list): Lista con los cursos académicos
00157     titulaciones (list): Lista con las titulaciones seleccionadas
00158
00159     Returns:
00160     pd.DataFrame: Datos de la consulta
00161     """
00162
00163 empty = pd.DataFrame()
00164
00165 if not gestor_id or not curso_academico or not titulaciones:
00166         return empty
00167
00168     data_universidad = universidades_gestor(gestor_id)
00169     if not data_universidad:
00170         return empty
00171
00172     try:
00173         titulaciones = list_to_tuple(titulaciones)
00174     except Exception as e:
00175         return empty
00176
00177     data = alumnos_egresados_nacionalidad_titulacion(
00178         data_universidad[0][0], curso_academico, titulaciones
00179     )
```

```
    if not data:
        return empty

    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 )
```

```
Alterna 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 63 del archivo callback_graph_egresados_nacionalidad_gestor.py.

```
00063 def toggle_modal(btn, is_open):
00064     """
00065 Alterna la visibilidad del modal con los datos de alumnos egresados por nacionalidad
00066
```

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

Returns:
bool: Nuevo estado del modal
"""

if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-egresados-nacionalidad", "children"),
    Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
```

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 15 del archivo callback_graph_egresados_nacionalidad_gestor.py.

```
15 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
16     """
17 Actualiza el gráfico de alumnos egresados por nacionalidad y titulación
18 del perfil "Gestor" de la pestaña "Indicadores académicos".
19
20     Args:
21     gestor_id (str): ID del gestor seleccionado
22     curso_academico (list): Lista con los cursos académicos
23     titulaciones (list): Lista con las titulaciones seleccionadas
24
25     Returns:
26     go.Figure: Figura con el gráfico
27
28     """
29
30 fig = go.Figure()
31
32 fig.update_layout(
33 barmode="stack",
34         title={"text": "Alumnos egresados por nacionalidad y titulación", "x": 0.5},
35         xaxis=dict(title="Titulaciones"),
36         yaxis=dict(title="N° Alumnos"),
37         showlegend=True,
38         legend={"title": "Nacionalidad"},
39     )
40
41     df = get_data(gestor_id, curso_academico, titulaciones)
42
43     if df.empty:
44         return fig
45
```

```
    df_pivot = df.pivot_table(
        index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
    ).fillna(0)

    for nacionalidad in df_pivot.columns:
        fig.add_trace(
            go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
        )

    return fig


@callback(
    Output("modal-egresados-nacionalidad", "is_open"),
    Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
    State("modal-egresados-nacionalidad", "is_open"),
)
```

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 87 del archivo callback_graph_egresados_nacionalidad_gestor.py.

```
def 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
    """

if not btn:
```

```
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-egresados-nacionalidad", "href"),
    Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
```

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 ingres
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 149 del archivo callback_graph_nuevo_ingreso_genero_gestor.py.

```
00149 def generate_csv(btn, gestor_id, curso_academico, titulaciones):
00150     """
00151 Genera un archivo CSV descargable con los datos de los alumnos de nuevo ingres
00152 por género y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00153
00154     Args:
00155     btn (int): Número de clicks en el botón
00156     gestor_id (str): ID del gestor seleccionado
00157     curso_academico (list): Lista con los cursos académicos
00158     titulaciones (list): Lista con las titulaciones seleccionadas
00159
00160     Returns:
00161     str: Contenido del archivo CSV
00162     """
00163
00164 if not btn:
00165         return ""
00166
00167     df = get_data(gestor_id, curso_academico, titulaciones)
00168     if df.empty:
00169         return ""
00170
00171     csv_string = df.to_csv(index=False, encoding="utf-8")
00172     csv_string = "data:text/csv;charset=utf-8," + csv_string
00173     return csv_string
00174
00175
```

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 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 176 del archivo callback_graph_nuevo_ingreso_genero_gestor.py.

```
00176 def get_data(gestor_id, curso_academico, titulaciones):
00177     """
00178 Obtiene los datos de la base da datos para el gráfico y tabla
00179
00180 Args:
00181 gestor_id (str): ID del gestor seleccionado
00182 curso_academico (list): Lista con los cursos académicos
00183 titulaciones (list): Lista con las titulaciones seleccionadas
00184
00185 Returns:
00186 pd.DataFrame: Datos para el gráfico y tabla
00187 """
00188
00189 empty = pd.DataFrame()
00190
00191 if not gestor_id or not curso_academico or not titulaciones:
00192         return empty
00193
00194     try:
00195         titulaciones = list_to_tuple(titulaciones)
00196     except Exception as e:
00197         return empty, None
00198
00199     data_universidad = universidades_gestor(gestor_id)
00200     if not data_universidad:
00201         return empty
00202
00203     data = alumnos_nuevo_ingreso_genero_titulacion(
00204         curso_academico, titulaciones, data_universidad[0][0]
00205     )
00206
00207     if not data:
00208         return empty
00209
00210     return pd.DataFrame(
00211         data, columns=["curso_academico", "titulacion", "genero", "cantidad"]
00212     )
```

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 )
```

```
Alterna 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 91 del archivo callback_graph_nuevo_ingreso_genero_gestor.py.

```
00091 def toggle_modal(btn, is_open):
00092     """
00093 Alterna la visibilidad del modal que muestra los datos de los alumnos de nuevo ingreso
00094
00095 Args:
00096 btn (int): Número de clicks en el botón
00097 is_open (bool): Estado actual del modal
00098
00099 Returns:
00100 bool: Nuevo estado del modal
00101 """
00102 if btn:
00103         return not is_open
00104     return is_open
00105
00106
00107 @callback(
00108     Output("table-container-nuevo-ingreso-genero", "children"),
00109     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00110     Input("selected-gestor-store", "data"),
00111     Input("curso-academico-gestor", "value"),
00112     Input("titulaciones-gestor", "value"),
00113 )
```

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 15 del archivo callback_graph_nuevo_ingreso_genero_gestor.py.

```
00015 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00016     """
00017 Actualiza el gráfico de alumnos de nuevo ingreso por género y titulación
00018 del perfil "Gestor" de la pestaña "Indicadores académicos".
00019
00020     Args:
00021     gestor_id (str): ID del gestor seleccionado
00022     curso_academico (list): Lista con los cursos académicos
00023     titulaciones (list): Lista con las titulaciones seleccionadas
00024
00025     Returns:
00026     go.Figure: Figura con el gráfico
00027
00028     """
00029 fig = go.Figure()
00030
00031 fig.update_layout(
00032 barmode="stack",
00033         title={"text": "Alumnos de nuevo ingreso por género y titulación", "x": 0.5},
00034         xaxis=dict(title="Titulaciones"),
00035         yaxis=dict(title="N° Alumnos"),
00036         showlegend=True,
00037         legend={"title": "Género"},
00038     )
00039
00040     # Asegurarse de que las columnas sean consistentes
00041     def map_genero(genero):
00042         if "masculin" in genero.lower():
00043             return "Hombres"
00044         elif "fem" in genero.lower():
00045             return "Mujeres"
00046         return genero
00047
00048     df = get_data(gestor_id, curso_academico, titulaciones)
00049
00050     if df.empty:
00051         return fig
00052
00053     df["genero"] = df["genero"].map(map_genero)
00054
00055     # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
00056     df_pivot = df.pivot_table(
00057         index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
00058     ).fillna(0)
00059
00060     # Asegurarse de que las columnas existen antes de graficar
00061     if "Hombres" in df_pivot.columns:
00062         fig.add_trace(
00063             go.Bar(
00064                 x=df_pivot.index,
00065                 y=df_pivot["Hombres"],
00066                 name="Hombres",
00067                 marker_color="blue",
00068                 opacity=0.7,
00069             )
00070         )
00071
00072     if "Mujeres" in df_pivot.columns:
00073         fig.add_trace(
00074             go.Bar(
00075                 x=df_pivot.index,
00076                 y=df_pivot["Mujeres"],
00077                 name="Mujeres",
00078                 marker_color="red",
00079                 opacity=0.7,
00080             )
00081         )
00082
00083     return fig
00084
00085
00086 @callback(
00087     Output("modal-nuevo-ingreso-genero", "is_open"),
00088     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00089     State("modal-nuevo-ingreso-genero", "is_open"),
00090 )
```

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 114 del archivo callback_graph_nuevo_ingreso_genero_gestor.py.

```
00114 def update_table(btn, gestor_id, curso_academico, titulaciones):
00115     """
00116 Actualiza la tabla de datos de los alumnos de nuevo ingreso por género y titulación
00117 del perfil "Gestor" de la pestaña "Indicadores académicos".
00118
00119     Args:
00120     btn (int): Número de clicks en el botón
00121     gestor_id (str): ID del gestor seleccionado
00122     curso_academico (list): Lista con los cursos académicos
00123     titulaciones (list): Lista con las titulaciones seleccionadas
00124
00125     Returns:
00126     dbc.Table: Tabla con los datos
00127     """
00128
00129 if not btn:
00130         return ""
00131
00132     df = get_data(gestor_id, curso_academico, titulaciones)
00133
00134     if df.empty:
00135         return dbc.Alert("No hay datos disponibles", color="info")
00136
00137     return dbc.Table.from_dataframe(
00138         df.head(50), striped=True, bordered=True, hover=True
00139     )
00140
00141
00142 @callback(
00143     Output("btn-descargar-nuevo-ingreso-genero", "href"),
00144     Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
00145     State("selected-gestor-store", "data"),
```

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

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 (gesto_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 132 del archivo callback_graph_nuevo_ingreso_nacionalidad_gestor.py.

```
def 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

    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return ""

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


```

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 (
              gesto_id,
              curso_academico,
              titulaciones )


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

Args:
gesto_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 161 del archivo callback_graph_nuevo_ingreso_nacionalidad_gestor.py.

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

```

```
Args:
gesto_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

"""

empty = pd.DataFrame()

if not gesto_id or not curso_academico or not titulaciones:
        return empty

    data_universidad = universidades_gestor(gesto_id)
    if not data_universidad:
        return empty

    try:
        titulaciones = list_to_tuple(titulaciones)
    except Exception as e:
        return empty

    data = alumnos_nuevo_ingreso_nacionalidad_titulacion(
        data_universidad[0][0], curso_academico, titulaciones
    )

    if not data:
        return empty

    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 )
```

```
Alterna 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 70 del archivo callback_graph_nuevo_ingreso_nacionalidad_gestor.py.

```
00070 def toggle_modal(btn, is_open):
00071     """
00072 Alterna la visibilidad del modal con la tabla de datos de alumnos de nuevo ingreso
00073 por nacionalidad y titulación del perfil "Gestor" de la pestaña "Indicadores académicos".
00074
00075     Args:
00076     btn (int): Número de clicks en el botón "Ver datos"
00077     is_open (bool): Estado actual del modal
00078
00079     Returns:
00080     bool: Nuevo estado del modal
00081
00082     """
00083
00084 if btn:
00085         return not is_open
00086     return is_open
00087
00088
00089 @callback(
00090     Output("table-container-nuevo-ingreso-nacionalidad", "children"),
00091     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00092     State("selected-gestor-store", "data"),
00093     Input("curso-academico-gestor", "value"),
00094     Input("titulaciones-gestor", "value"),
00095 )
```

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 18 del archivo callback_graph_nuevo_ingreso_nacionalidad_gestor.py.

```
00018 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00019     """
00020 Actualiza el gráfico de alumnos de nuevo ingreso por nacionalidad y titulación
00021 del perfil "Gestor" de la pestaña "Indicadores académicos".
00022
00023     Args:
00024     gestor_id (str): ID del gestor seleccionado
00025     curso_academico (list): Lista con los cursos académicos
00026     titulaciones (list): Lista con las titulaciones seleccionadas
00027
00028     Returns:
00029     go.Figure: Figura con el gráfico
00030
00031     """
00032
00033 fig = go.Figure()
00034
00035 fig.update_layout(
00036 barmode="stack",
00037         title={
00038             "text": "Alumnos de nuevo ingreso por nacionalidad y titulación",
00039             "x": 0.5,
00040         },
00041         xaxis_title="Titulaciones",
00042         yaxis_title="N° Alumnos",
00043         showlegend=True,
00044         legend={"title": "Nacionalidad"},
00045     )
00046
00047     df = get_data(gestor_id, curso_academico, titulaciones)
00048
00049     if df.empty:
00050         return fig
00051
00052     # Pivotear el DataFrame para obtener las cantidades por nacionalidad en columnas separadas
00053     df_pivot = df.pivot_table(
00054         index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
00055     ).fillna(0)
00056
00057     for nacionalidad in df_pivot.columns:
00058         fig.add_trace(
00059             go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
00060         )
00061
00062     return fig
00063
00064
00065 @callback(
00066     Output("modal-nuevo-ingreso-nacionalidad", "is_open"),
00067     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00068     State("modal-nuevo-ingreso-nacionalidad", "is_open"),
00069 )
```

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 96 del archivo callback_graph_nuevo_ingreso_nacionalidad_gestor.py.

```
00096 def update_table(btn, gestor_id, curso_academico, titulaciones):
00097     """
00098 Actualiza la tabla de datos de alumnos de nuevo ingreso por nacionalidad y titulación
00099 del perfil "Gestor" de la pestaña "Indicadores académicos".
00100
00101     Args:
00102     btn (int): Número de clicks en el botón "Ver datos"
00103     gestor_id (str): ID del gestor seleccionado
00104     curso_academico (list): Lista con los cursos académicos
00105     titulaciones (list): Lista con las titulaciones seleccionadas
00106
00107     Returns:
00108     dbc.Table: Tabla con los datos
00109
00110     """
00111
00112 if not btn:
00113         return ""
00114
00115     df = get_data(gestor_id, curso_academico, titulaciones)
00116
00117     if df.empty:
00118         return dbc.Alert("No hay datos disponibles", color="info")
00119
00120     return dbc.Table.from_dataframe(
00121         df.head(50), striped=True, bordered=True, hover=True
00122     )
00123
00124
00125 @callback(
00126     Output("btn-descargar-nuevo-ingreso-nacionalidad", "href"),
00127     Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
00128     State("selected-gestor-store", "data"),
00129     Input("curso-academico-gestor", "value"),
00130     Input("titulaciones-gestor", "value"),
00131 )
```

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 133 del archivo callback_graph_duracion_media_estudios_nota_gestor.py.

```
00133 def generate_csv(btn, gestor_id):
00134     """
00135 Genera un archivo CSV descargable con los datos de duración media de los estudios
00136 con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".
00137
00138     Args:
00139     btn (int): Número de clicks en el botón
00140     gestor_id (str): ID del gestor seleccionado
00141
00142     Returns:
00143     str: Enlace al archivo CSV
00144     """
00145
00146 if not btn:
00147         return ""
00148
00149     df = get_data(gestor_id)
00150
00151     if df.empty:
00152         return ""
00153
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:

4.58.1.2. 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 159 del archivo callback_graph_duracion_media_estudios_nota_gestor.py.

```
00159 def get_data(gestor_id):
00160     """
00161 Obtiene los datos de la base de datos.
00162
00163 Args:
00164 gestor_id (str): ID del gestor seleccionado
00165
00166 Returns:
00167 pd.DataFrame: Datos de la base de datos
00168 """
00169 empty = pd.DataFrame()
00170
00171 if not gestor_id:
00172         return empty
00173
00174     data_universidad = universidades_gestor(gestor_id)
00175     if not data_universidad:
00176         return empty
00177
00178     data = duracion_media_estudios_nota_gestor(data_universidad[0][0])
00179     if not data:
00180         return empty
00181
00182     df = pd.DataFrame(
00183         data,
00184         columns=[
00185             "nota_media",
00186             "titulacion",
00187             "rama",
00188             "curso_academico",
00189             "duracion_media",
00190             "numero_alumnos",
00191         ],
00192     )
00193     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 )
```

```
Alterna 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 81 del archivo callback_graph_duracion_media_estudios_nota_gestor.py.

```
00081 def toggle_modal(btn, is_open):
00082     """
00083 Alterna la visibilidad del modal con los datos de duración media de los estudios
00084 con respecto a la nota media.
00085
00086 Args:
00087 btn (int): Número de clicks en el botón
00088 is_open (bool): Estado actual del modal
00089
```

```
Returns:
bool: Nuevo estado del modal
"""
if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-duracion-estudios", "children"),
    Input("btn-ver-datos-duracion-estudios", "n_clicks"),
    State("selected-gestor-store", "data"),
)
```

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 13 del archivo callback_graph_duracion_media_estudios_nota_gestor.py.

```
13 def update_graph_gestor(gestor_id):
14     """
15 Actualiza el gráfico de duración media de los estudios con respecto a la nota media
16 del perfil "Gestor" de la pestaña "Resultados académicos".
17
18     Args:
19     gestor_id (str): ID del gestor seleccionado
20
21     Returns:
22     go.Figure: Figura con el gráfico
23     """
24
25 fig = go.Figure()
26
27 if not gestor_id:
28         return fig
29
30     data = get_data(gestor_id)
31
32     if data.empty:
33         return fig
34
35     fig = px.scatter(
36         data,
37         x="nota_media",
38         y="duracion_media",
39         color="rama",
40         size="numero_alumnos",
41         hover_name="titulacion",
42         animation_frame="curso_academico",
43         animation_group="rama",
44         category_orders={
45             "rama": sorted(data["rama"].unique())
46         },
47     )
48
49     fig.update_layout(
50         title={
51             "text": "Duración media de los estudios con respecto a la nota media",
52             "x": 0.5,
53         },
54         xaxis_title="Nota media",
55         yaxis_title="Duración media de los estudios",
```

```
        legend={"title": "Rama de conocimiento"},
    )

    fig.update_traces(
        marker=dict(line=dict(width=1, color="DarkSlateGrey")),
        textposition="top center"
    )

    # Ajustar el rango de los ejes para que las burbujas no se corten
    fig.update_xaxes(range=[data["nota_media"].min() - 1, data["nota_media"].max() + 1])
    fig.update_yaxes(
        range=[data["duracion_media"].min() - 1, data["duracion_media"].max() + 1]
    )

    if len(data["curso_academico"].unique()) > 1:
        fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 1000

    return fig


@callback(
    Output("modal-duracion-estudios", "is_open"),
    Input("btn-ver-datos-duracion-estudios", "n_clicks"),
    State("modal-duracion-estudios", "is_open"),
)
```

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 103 del archivo callback_graph_duracion_media_estudios_nota_gestor.py.

```
103 def update_table(btn, gestor_id):
104     """
105 Actualiza la tabla con los datos de duración media de los estudios
106 con respecto a la nota media del perfil "Gestor" de la pestaña "Resultados académicos".
107
108     Args:
109     btn (int): Número de clicks en el botón
110     gestor_id (str): ID del gestor seleccionado
111
112     Returns:
113     dbc.Table: Tabla con los datos
114     """
```

```
if not btn:
        return ""

    df = get_data(gestor_id)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-csv-duracion-estudios", "href"),
    Input("btn-descargar-csv-duracion-estudios", "n_clicks"),
    State("selected-gestor-store", "data"),
)
```

Gráfico de llamadas para esta función:

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

Funciones

- def update_grpht_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 112 del archivo callback_graph_nota_acceso_titulacion_gestor.py.

```
def 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

    """

if not btn:
        return ""

    df = get_data(gestor_id)

    if df.empty:
        return ""

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


```

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 139 del archivo callback_graph_nota_acceso_titulacion_gestor.py.

```
def 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
"""
empty = pd.DataFrame()

if not gestor_id:
        return empty

    data_universidad = universidades_gestor(gestor_id)
```

```
    if not data_universidad:
        return empty

    data = nota_media_acceso_titulacion(data_universidad[0][0])

    if not data:
        return empty

    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 57 del archivo callback_graph_nota_acceso_titulacion_gestor.py.

```
def 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

    """
if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-nota-acceso", "children"),
    Input("btn-ver-datos-nota-acceso", "n_clicks"),
    State("selected-gestor-store", "data"),
)
```

4.59.1.4. update_grpht_gestor()

```
def callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor.update_↩
grpht_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 12 del archivo callback_graph_nota_acceso_titulacion_gestor.py.

```
def update_grpht_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

    """

fig = go.Figure()

fig.update_layout(
title={"text": "Evolución nota de acceso por titulación", "x": 0.5},
        xaxis_title="Curso académico",
        yaxis_title="Nota media de acceso",
        legend={"title": "Titulaciones"},
    )

    df = get_data(gestor_id)

    if df.empty:
        return fig

    for titulation, group in df.groupby("titulacion"):
        fig.add_trace(
            go.Scatter(
                x=group["curso_academico"],
                y=group["nota"],
```

```
                mode="lines+markers",
                name=titulation,
            )
        )

    return fig


@callback(
    Output("modal-nota-acceso", "is_open"),
    Input("btn-ver-datos-nota-acceso", "n_clicks"),
    State("modal-nota-acceso", "is_open"),
)
```

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 80 del archivo callback_graph_nota_acceso_titulacion_gestor.py.

```
def 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

    """

if not btn:
        return ""

    df = get_data(gestor_id)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )
```

```
@callback(
    Output("btn-descargar-csv-nota-acceso", "href"),
    Input("btn-ver-datos-nota-acceso", "n_clicks"),
    State("selected-gestor-store", "data"),
)
```

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 120 del archivo callback_graph_tasa_abandono_gestor.py.

```
00120 def generate_csv(btn, gestor_id, curso_academico):
00121     """
00122 Genera un archivo CSV descargable con los datos de la tasa de abandono por titulación
00123 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00124
00125     Args:
00126     btn (int): Número de clicks en el botón
00127     gestor_id (str): ID del gestor seleccionado
00128     curso_academico (list): Lista con los cursos académicos
00129
00130     Returns:
00131     str: Enlace al archivo CSV
00132     """
00133
00134 if not btn:
00135         return ""
00136     df = get_data(gestor_id, curso_academico)
00137
00138     if df.empty:
00139         return ""
00140
00141     df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00142     csv_string = df.to_csv(index=False, encoding="utf-8")
00143     csv_string = "data:text/csv;charset=utf-8," + csv_string
00144     return csv_string
00145
00146
```

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 147 del archivo callback_graph_tasa_abandono_gestor.py.

```
00147 def get_data(gestor_id, curso_academico):
00148     """
00149 Obtiene los datos de la base de datos.
00150
00151 Args:
00152 gestor_id (str): ID del gestor seleccionado
00153 curso_academico (list): Lista con los cursos académicos
00154
00155 Returns:
00156 pd.DataFrame: Datos de la tasa de abandono por titulación
00157 """
00158 empty = pd.DataFrame()
00159
00160 if not gestor_id or not curso_academico:
00161         return empty
00162
00163     data_universidad = universidades_gestor(gestor_id)
00164     if not data_universidad:
00165         return empty
00166     try:
00167         curso_academico = list_to_tuple(curso_academico)
00168     except Exception as e:
00169         print("Error:", e)
00170         return empty
00171
00172     data = tasa_abandono_titulacion_gestor(data_universidad[0][0], curso_academico)
00173
00174     if not data:
00175         return empty
00176
00177     return pd.DataFrame(
00178         data,
00179         columns=[
00180             "curso_academico",
00181             "titulacion",
00182             "numero_matriculados",
00183             "numero_abandonos",
00184         ],
00185     )
```

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 62 del archivo callback_graph_tasa_abandono_gestor.py.

```
00062 def toggle_modal(btn, is_open):
00063     """
00064 Alternar la visibilidad del modal de datos de la tasa
00065 de abandono por titulación del perfil "Gestor" de la pestaña "Riesgo de abandono".
00066
00067     Args:
00068     btn (int): Número de clicks en el botón
00069     is_open (bool): Estado actual del modal
00070
00071     Returns:
00072     bool: Nuevo estado del modal
00073     """
00074
00075 if btn:
00076         return not is_open
00077     return is_open
00078
00079
00080 @callback(
00081     Output("table-container-tasa-abandono", "children"),
00082     Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00083     State("selected-gestor-store", "data"),
00084     Input("curso-all-academico-gestor", "value"),
00085 )
```

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 14 del archivo callback_graph_tasa_abandono_gestor.py.

```
00014 def update_graph_gestor(curso_academico, gestor_id):
00015     """
00016 Actualiza el gráfico de la tasa de abandono por titulación
00017 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00018
00019     Args:
00020     curso_academico (list): Lista con los cursos académicos
00021     gestor_id (str): ID del gestor seleccionado
00022
00023     Returns:
00024     go.Figure: Figura con el gráfico
00025     """
00026 fig = go.Figure()
00027
00028 fig.update_layout(
00029 title={"text": "Tasa de abandono por titulación ", "x": 0.5},
00030         xaxis_title="Curso académico",
00031         yaxis_title="Tasa de abandono (%)",
00032         showlegend=True,
00033         legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
00034     )
00035
00036     df = get_data(gestor_id, curso_academico)
00037
00038     if df.empty:
00039         return fig
00040
00041     df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00042
00043     for titulacion in df["titulacion"].unique():
00044         df_titulacion = df[df["titulacion"] == titulacion]
00045         fig.add_trace(
00046             go.Scatter(
00047                 x=df_titulacion["curso_academico"],
00048                 y=df_titulacion["tasa_abandono"],
00049                 mode="lines+markers",
00050                 name=titulacion,
00051             )
00052         )
00053
00054     return fig
00055
00056
00057 @callback(
00058     Output("modal-tasa-abandono", "is_open"),
00059     Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00060     State("modal-tasa-abandono", "is_open"),
00061 )
```

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 86 del archivo callback_graph_tasa_abandono_gestor.py.

```
00086 def update_table(btn, gestor_id, curso_academico):
00087     """
00088 Actualiza la tabla con los datos de la tasa de abandono por titulación
00089 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00090
00091     Args:
00092     btn (int): Número de clicks en el botón
00093     gestor_id (str): ID del gestor seleccionado
00094     curso_academico (list): Lista con los cursos académicos
00095
00096     Returns:
00097     dbc.Table: Tabla con los datos
00098     """
00099
00100 if not btn:
00101         return ""
00102
00103     df = get_data(gestor_id, curso_academico)
00104
00105     if df.empty:
00106         return dbc.Alert("No hay datos disponibles", color="info")
00107
00108     df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
00109     return dbc.Table.from_dataframe(
00110         df.head(50), striped=True, bordered=True, hover=True, responsive=True
00111     )
00112
00113
00114 @callback(
00115     Output("btn-descargar-tasa-abandono", "href"),
00116     Input("btn-ver-datos-tasa-abandono", "n_clicks"),
00117     State("selected-gestor-store", "data"),
00118     Input("curso-all-academico-gestor", "value"),
00119 )
```

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 113 del archivo callback_graph_tasa_graduacion_gestor.py.

```
00113 def generate_csv(n, curso_academico, gestor_id):
00114     """
00115 Genera un archivo CSV descargable con los datos de la tasa de graduación por titulación
00116 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00117
00118     Args:
00119     n (int): Número de clicks en el botón
00120     curso_academico (list): Lista con los cursos académicos
00121     gestor_id (str): ID del gestor seleccionado
00122     """
00123
00124 if not n:
00125         return None
00126
00127     df = get_data(gestor_id, curso_academico)
00128
00129     if df is None:
00130         return None
00131
00132     csv_string = df.to_csv(index=False, encoding="utf-8")
00133     csv_string = "data:text/csv;charset=utf-8," + csv_string
00134
00135     return csv_string
00136
00137
```

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 138 del archivo callback_graph_tasa_graduacion_gestor.py.

```
00138 def get_data(gestor_id, curso_academico):
00139     """
00140 Obtiene los datos de la tasa de graduación por titulación
00141 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00142
00143     Args:
00144     gestor_id (str): ID del gestor seleccionado
00145     curso_academico (list): Lista con los cursos académicos
00146
00147     Returns:
00148     pd.DataFrame: Datos de la tasa de graduación
00149     """
00150
00151 empty = pd.DataFrame()
00152
00153 if not gestor_id or not curso_academico:
00154         return empty
00155
00156     try:
00157         curso_academico = list_to_tuple(curso_academico)
00158     except Exception as e:
00159         print("Error:", e)
00160         return empty
00161
00162     data_universidad = universidades_gestor(gestor_id)
00163
00164     if not data_universidad:
00165         return empty
00166
00167     data = tasa_graduacion_titulacion_gestor(data_universidad[0][0], curso_academico)
00168
00169     if not data:
00170         return empty
00171
00172     df = pd.DataFrame(
00173         data,
00174         columns=[
00175             "numero_matriculados",
```

```
            "numero_egresados",
            "curso_academico",
            "titulacion",
        ],
    )
    df["tasa_graduacion"] = (df["numero_egresados"] / df["numero_matriculados"]) * 100

    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 60 del archivo callback_graph_tasa_graduacion_gestor.py.

```
def 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
    """

if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-tasa-graduacion", "children"),
    Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
    State("curso-all-academico-gestor", "value"),
    State("selected-gestor-store", "data"),
)
```

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 14 del archivo callback_graph_tasa_graduacion_gestor.py.

```
def 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
    """

fig = go.Figure()

fig.update_layout(
title={"text": "Tasa de graduación por titulación ", "x": 0.5},
        xaxis_title="Curso académico",
        yaxis_title="Tasa de graduación (%)",
        showlegend=True,
        legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
    )

    df = get_data(gestor_id, curso_academico)

    if df.empty:
        return fig

    for titulacion, group in df.groupby("titulacion"):
        fig.add_trace(
            go.Scatter(
                x=group["curso_academico"],
```

```
                y=group["tasa_graduacion"],
                mode="lines+markers",
                name=titulacion,
            )
        )

    return fig


@callback(
    Output("modal-tasa-graduacion", "is_open"),
    Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
    State("modal-tasa-graduacion", "is_open"),
)
```

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 81 del archivo callback_graph_tasa_graduacion_gestor.py.

```
00081 def update_table(btn, curso_academico, gestor_id):
00082     """
00083 Actualiza la tabla de datos de la tasa de graduación por titulación
00084 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00085
00086     Args:
00087     btn (int): Número de clicks en el botón
00088     curso_academico (list): Lista con los cursos académicos
00089     gestor_id (str): ID del gestor seleccionado
00090
00091     Returns:
00092     dbc.Table: Tabla con los datos
00093     """
00094 if not btn:
00095         return ""
00096
00097     df = get_data(gestor_id, curso_academico)
00098
00099     if df.empty:
```

```
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True, responsive=True
    )


@callback(
    Output("btn-descargar-tasa-graduacion", "href"),
    Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
    State("curso-all-academico-gestor", "value"),
    State("selected-gestor-store", "data"),
)
```

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 47 del archivo callback_resumen_gestor.py.

```
00047 def not_data():
00048     return html.Div(
00049         [
00050             html.H2("Resumen"),
00051             html.P("Universidad:", className="resumen-label"),
00052             html.P("No disponible"),
00053             html.P("Gestor:", className="resumen-label"),
00054             html.P("No disponible"),
00055             html.P("Número de alumnos matriculados:", className="resumen-label"),
00056             html.P("No disponible"),
00057             html.Hr(),
00058         ]
00059     )
```

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 9 del archivo callback_resumen_gestor.py.

```
00009 def update_resumen_gestor(gestor_id):
00010     """
00011 Actualiza el resumen del gestor seleccionado.
00012
00013 Args:
00014 gestor_id (str): ID del gestor seleccionado
00015
00016 Returns:
00017 html.Div: Resumen del gestor seleccionado
00018 """
00019
00020 if not gestor_id:
00021         return not_data()
00022
00023     data = universidades_gestor(gestor_id)
00024
00025     if not data:
00026         return not_data()
00027
00028     data_alumnos = numero_alumnos_matriculados_universidad(data[0][1])
00029
```

```
    if not data_alumnos:
        return not_data()

    return html.Div(
        [
            html.H2("Resumen"),
            html.P("Universidad:", className="resumen-label"),
            html.P(data[0][1]),
            html.P("Gestor:", className="resumen-label"),
            html.P(gestor_id),
            html.P("Número de alumnos matriculados:", className="resumen-label"),
            html.P(data_alumnos[0][0]),
            html.Hr(),
        ]
    )


```

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 12 del archivo callback_select_gestor.py.

```
00012 def store_selected_gestor(selected_value, stored_value):
00013     """
00014 Almacena el gestor seleccionado en un store.
00015
00016 Args:
00017 selected_value (str): Valor seleccionado en el dropdown
00018 stored_value (dict): Datos almacenados en el store
00019
00020 Returns:
00021 str: Valor seleccionado en el dropdown
00022 list: Opciones del dropdown
00023 str: Valor almacenado en el store
00024 """
00025
00026 data = gestores_all()
00027
00028 if not data:
00029         return None, [], None
00030
00031     opciones_dropdown = [{"label": gestor[0], "value": gestor[0]} for gestor in data]
00032
00033     if selected_value is None and stored_value in [
00034         op["value"] for op in opciones_dropdown
00035     ]:
00036         return stored_value, opciones_dropdown, stored_value
00037
00038     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 19 del archivo callback_tabs_gestor.py.

```
00019 def render_content(tab, selected_gestor):
00020     """
00021 Renderiza el contenido de las pestañas de la sección "Gestor".
00022
00023     Args:
00024     tab (str): Pestaña seleccionada
00025     selected_gestor (dict): Datos del gestor seleccionado
00026
00027     Returns:
00028     html.Div: Contenido de la pestaña seleccionada
00029     """
00030
00031 if tab == 'indicadores-academicos-tab':
00032         return html.Div([
00033             select_gestor(),
00034             html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
00035             html.Div([
00036                 sidebar([
00037                     resumen_gestor(),
00038                     filters([
00039                         filter_curso_academico_gestor(),
00040                         filter_titulaciones_gestor()
00041                     ]),
00042                 ]),
00043                 graphs_indicadores_gestor()
00044             ], className='content-layout-dashboard')
00045
00046         ])
00047     elif tab == 'resultados-academicos-tab':
00048         return html.Div([
00049             html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
00050              html.Div([
00051                 sidebar([
00052                     resumen_gestor(),
00053                 ]),
00054                 graphs_resultados_gestor()
00055             ], className='content-layout-dashboard')
00056         ])
00057     elif tab == 'riesgo-abandono-tab':
00058         return html.Div([
00059            html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
00060              html.Div([
00061                 sidebar([
00062                     resumen_gestor(),
00063                     filters([
00064                         filter_all_curso_academico_gestor()
00065                     ]),
00066                 ]),
00067                 graphs_riesgo_abandono_gestor()
00068             ], className='content-layout-dashboard')
00069         ])
00070     elif tab == 'recomendaciones-tab':
00071         return html.Div([
00072             recomendador_gestor()
00073         ])
00074
00075
00076
```

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 5 del archivo filter_asignaturas_matri_alumnado.py.

```
00005 def filter_asignaturas_matri_alumnado():
00006     """
00007 Crea un componente con un dropdown que contiene las asignaturas matriculadas por el alumno.
00008
00009 Returns:
00010 html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
00011 """
00012 return html.Div(
00013         [
00014             html.Br(),
00015             html.Label("Asignaturas matriculadas"),
00016             dcc.Dropdown(
00017                 id="asignaturas-matriculadas",
00018                 searchable=True,
00019                 multi=True,
00020                 value=None,
00021                 clearable=True,
00022                 options=[],
00023                 maxHeight=300,
00024                 optionHeight=50,
00025                 placeholder="Seleccione una opción",
00026             ),
00027             html.Button(
00028                 "Seleccionar todo",
00029                 id="select-all-button",
00030                 className="button-select-all-filter",
00031                 n_clicks=0,
00032             ),
00033         ]
00034     )
```

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 5 del archivo filter_curso_academico_alumnado.py.

```
00005 def filter_curso_academico_alumnado():
00006     """
00007 Crea un componente con un dropdown que contiene los cursos académicos.
00008
00009 Returns:
00010 html.Div: Componente con un dropdown y un botón para seleccionar todos los cursos académicos
00011 """
00012 return html.Div(
00013         [
00014             html.Br(),
00015             html.Label("Curso académico"),
00016             dcc.Dropdown(
00017                 id="curso-academico",
00018                 searchable=False,
00019                 multi=True,
00020                 clearable=True,
00021                 options=[],
00022                 value=None,
00023                 maxHeight=300,
00024                 placeholder="Seleccione una opción",
00025             ),
00026             html.Button(
00027                 "Seleccionar todo",
00028                 id="select-all-cursos-academicos",
00029                 className="button-select-all-filter",
00030                 n_clicks=0,
00031             ),
00032         ]
00033     )
```

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 5 del archivo filter_titulacion_alumnado.py.

```
00005 def filter_titulacion_alumnado():
00006     """
00007 Crea un componente con un dropdown que contiene las titulaciones del perfil "Alumno".
00008
00009     Returns:
00010     html.Div: Componente con un dropdown y almacenamiento de la titulación seleccionada
00011     """
00012 return html.Div(
00013         [
00014             html.Label("Titulación"),
00015             dcc.Dropdown(
00016                 id="titulacion-alumnado",
00017                 options=[],
00018                 value=None,
00019                 searchable=False,
00020                 clearable=False,
00021                 optionHeight=50,
00022                 maxHeight=300,
00023                 placeholder="Seleccione una opción",
00024             ),
00025             dcc.Store(id="selected-titulacion-alumnado-store", storage_type="local"),
00026         ]
00027     )
```

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 "Rendimento Académico general" del perfil "Alumnado"

Returns:
html.Div: Gráficos
```

Definición en la línea 8 del archivo graphs_general_alumnado.py.

```
00008 def graphs_general_alumnado():
00009     """
00010 Retona los gráficos de la pestaña "Rendimento Académico general" del perfil "Alumnado"
00011
00012     Returns:
00013     html.Div: Gráficos
00014     """
00015 graph_ids = [
00016 "asignaturas-superadas-general-mi-nota",
00017         "nota-cualitativa-general-mi-nota",
00018         "nota-media-general-mi-nota",
00019     ]
00020
00021     config = {"displayModeBar": False}
00022     item_class = "graph-item-general-alumnado"
00023
00024     return html.Div(
00025         [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
00026         className="graphs-container-general-alumnado",
00027     )
```

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 9 del archivo graphs_personal_alumnado.py.

```
00009 def graphs_personal_alumnado():
00010     """
00011 Retorna los gráficos de la pestaña "Expediente académico Personal" del perfil "Alumnado"
00012
00013     Returns:
00014     html.Div: Gráficos
00015     """
00016 graph_ids = [
00017 "graph-evolucion-progreso-academico",
00018         "graph-bar-evolucion-asignaturas-matriculadas",
00019         "graph-bar-calificaciones-por-asignatura",
00020         "graph-bar-tasa-rendimiento",
00021     ]
00022
00023     config = {"displayModeBar": False}
00024     item_class = "graph-item-personal-alumnado"
00025
00026     return html.Div(
00027         [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
00028         className="graphs-container-personal-alumnado",
00029     )
```

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 3 del archivo recomendador_alumnado.py.

```
00003 def recomendador_alumnado():
00004     """
00005 Esta función retorna el contenido de la pestaña "Recomendador" del perfil "Alumnado".
00006
00007     Returns:
00008     html.Div: Layout de la pestaña "Recomendador" del perfil "Alumnado"
00009     """
00010
00011 return html.Div([
00012         html.H1("Deserción universitaria: ¿Cuáles son las razones y cómo prevenirla?",
      className='titulo-recomendador-alumnado'),
00013         html.Img(src='assets/images/abandono_academico.jpg',
      className='imagen-recomendador-alumnado'),
00014         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'),
00015         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-alumnado'),
00016         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'),
00017         html.Br(),
00018         html.Br(),
00019         html.H2("¿Cuáles son las razones de la deserción universitaria?",
      className='sb-recomendador-alumnado'),
00020         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'),
00021         html.Ul([
00022             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'),
00023             html.Li(html.Span([html.Strong("Problemas académicos:"), " Las dificultades para aprobar
      asignaturas pueden hacer que te sientas frustrado."]), className='li-recomendador-alumnado'),
00024             html.Li(html.Span([html.Strong("Problemas personales:"), " Situaciones como problemas de
      salud, familiares o relaciones personales pueden afectarte."]), className='li-recomendador-alumnado'),
00025             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'),
00026             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'),
00027         ], className='ul-recomendador-alumnado'),
00028         html.H2("¿Cómo prevenir la deserción universitaria?", className='sb-recomendador-alumnado'),
00029         html.P("Para prevenir la deserción, puedes buscar ayuda en los siguientes recursos que la
      universidad ofrece:", className='p-recomendador-alumnado'),
00030         html.Ul([
00031             html.Li("Solicitar becas y ayudas económicas si tienes dificultades financieras.",
      className='li-recomendador-alumnado'),
00032             html.Li("Participar en programas de tutoría y apoyo académico si tienes problemas con tus
      asignaturas.", className='li-recomendador-alumnado'),
00033             html.Li("Aprovechar los programas de orientación y asesoramiento para adaptarte mejor a la
      vida universitaria.", className='li-recomendador-alumnado'),
00034             html.Li("Asistir a actividades y talleres que mejoren tu motivación y habilidades de
      estudio.", className='li-recomendador-alumnado'),
00035         ], className='ul-recomendador-alumnado'),
00036     ])
```

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 5 del archivo resumen_alumnado.py.

```
00005 def resumen_alumnado():
00006     """
00007 Crea el layout del resumen del alumnado.
00008
00009 Returns:
00010 html.Div: Layout del resumen del alumnado
00011 """
00012 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 5 del archivo select_alumnado.py.

```
00005 def select_alumnado():
00006     """
00007 Crea el desplegable para seleccionar un alumno.
00008
00009 Returns:
00010 html.Div: Desplegable para seleccionar un alumno
00011 """
00012 return html.Div(
00013         [
00014             html.Div(
00015                 [
00016                     dcc.Dropdown(
00017                         options=[],
00018                         value=None,
00019                         id="alumnado-dropdown",
00020                         clearable=False,
00021                         placeholder="Seleccione un alumno",
00022                     )
00023                 ],
00024                 className="select-alumnado",
00025             ),
00026         ]
00027     )
```

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.tabs_alumnado ( )


Contiene las pestañas de la vista del alumnado.

Returns:
html.Div: Pestañas del alumnado
```

Definición en la línea 5 del archivo tabs_alumnado.py.

```
00005 def tabs_alumnado():
00006     """
00007 Contiene las pestañas de la vista del alumnado.
00008
00009 Returns:
00010 html.Div: Pestañas del alumnado
00011 """
00012 return html.Div(
00013         [
00014             dcc.Tabs(
00015                 id="tabs-alumnado",
00016                 value="expediente-personal-tab",
00017                 children=[
00018                     dcc.Tab(
00019                         label="Expediente académico personal",
00020                         value="expediente-personal-tab",
00021                     ),
00022                     dcc.Tab(
00023                         label="Rendimiento académico general",
00024                         value="rendimiento-academico-tab",
00025                     ),
00026                     dcc.Tab(label="Recomendaciones", value="recomendador-tab"),
00027                 ],
00028                 className="tabs",
00029             ),
00030             html.Div(id="tabs-alumnado-content"),
00031             dcc.Store(id="selected-alumnado-store", storage_type="local"),
00032             dcc.Location(id="url", refresh=False),
00033         ]
00034     )
```

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 4 del archivo create_graph.py.

```
00004 def create_graph(graph_id, item_class, config):
00005     """
00006 Crea un gráfico con un id, una clase y una configuración específica.
00007
00008 Args:
00009 graph_id (str): ID del gráfico
00010 item_class (str): Clase del gráfico
00011 config (dict): Configuración del gráfico
00012
00013 Returns:
00014 html.Div: Gráfico
00015 """
00016 return html.Div(
00017         [dcc.Loading(children=[dcc.Graph(id=graph_id, figure={}, config=config)])],
00018         className=item_class,
00019     )
```

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 6 del archivo create_graph_with_table.py.

```
00006 def create_graph_with_table(graph_id, item_class, config, modal_config):
00007     """
00008 Crea un gráfico con un modal para mostrar los datos y descargarlos.
00009
00010 Args:
00011 graph_id (str): ID del gráfico
00012 item_class (str): Clase del gráfico
00013 config (dict): Configuración del gráfico
00014 modal_config (dict): Configuración del modal
00015
00016 Returns:
00017 html.Div: Gráfico con modal para mostrar los datos
00018 """
00019 return html.Div(
00020         [create_graph(graph_id, item_class, config), create_modal(**modal_config)],
00021         className=item_class,
00022     )
```

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 4 del archivo filters.py.

```
00004 def filters(filters):
00005     """
00006 Crea un componente que contiene los filtros de la aplicación.
00007
00008 Args:
00009 filters (list): Lista de filtros
00010
00011 Returns:
00012 html.Div: Componente que contiene los filtros
00013 """
00014 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 3 del archivo footer.py.

```
00003 def footer():
00004     """
00005 Retorna el footer de la aplicación
00006
00007 Returns:
00008 html.Div: Footer de la aplicación
00009 """
00010 return html.Div([
00011         html.P('2023-2024 Universidad de la Laguna - Visualización de datos académicos'),
00012         html.P('Pabellón de Gobierno, C/ Padre Herrera s/n Apartado Postal 456 38200, San Cristóbal de
      La Laguna'),
00013         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 5 del archivo header.py.

```
00005 def header(store_role):
00006     """
00007 Contiene el header de la aplicación y el desplegable para seleccionar el rol.
00008
00009 Args:
00010 store_role: str: ID del store que contiene el rol del usuario
00011
00012 Returns:
00013 html.Div: Header de la aplicación
00014 """
00015 return html.Div(
00016         [
00017             html.Img(src="assets/images/logoULL.png", className="logo"),
00018             html.H1("Visualización de datos académicos", className="title"),
00019             dcc.Dropdown(
00020                 options=[
00021                     {"label": "Alumno", "value": "Alumno"},
00022                     {"label": "Docente", "value": "Docente"},
00023                     {"label": "Gestor", "value": "Gestor"},
00024                 ],
00025                 id="dropdown_role",
00026                 className="dropdown_role",
00027                 clearable=False,
00028                 searchable=False,
00029                 placeholder="Selecciona un rol",
00030                 value="Alumno",
00031             ),
00032             dcc.Store(id=store_role, storage_type="session"),
00033         ],
00034         className="header",
00035     )
```

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

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 5 del archivo modal_data.py.

```
00005 def create_modal(modal_id, table_container_id, download_button_id, view_data_button_id):
00006     """
00007 Crea el modal para mostrar los datos de una tabla. El modal tiene un botón para descargar los datos en
      formato CSV.
00008
00009 Args:
00010 modal_id (str): ID del modal
00011 table_container_id (str): ID del contenedor de la tabla
00012 download_button_id (str): ID del botón para descargar los datos
00013 view_data_button_id (str): ID del botón para abrir el modal
00014
00015 Returns:
00016 dbc.Modal: Modal para mostrar los datos
00017 """
00018 return html.Div([
00019         dbc.Button('Ver datos', id=view_data_button_id, n_clicks=0, color='primary'),
00020         dbc.Modal([
00021             dbc.ModalHeader("Datos"),
00022             dbc.ModalBody(html.Div(id=table_container_id), style={'maxHeight': 'calc(100vh - 200px)',
      'overflowY': 'auto'}),
00023             dbc.ModalFooter(
00024                 dbc.Button("Descargar CSV", id=download_button_id, color='primary')
00025             )
00026         ], id=modal_id, is_open=False, size="lg")
00027     ])
```

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 6 del archivo sidebar.py.

```
00006 def sidebar(components):
00007     """
00008 Sidebar del dashboard donde se encuentra el resumen de la información y los filtros
00009
00010 Args:
00011 components (list): Componentes que se desplegarán en el sidebar
00012
00013 Returns:
00014 html.Div: Sidebar del dashboard
00015
00016 """
00017 sidebar_content = html.Div(components, className="sidebar")
00018     sidebar_collapse = dbc.Collapse(sidebar_content, id="collapse", is_open=True)
00019
00020     return html.Div(sidebar_collapse)
```

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 5 del archivo filter_all_asignaturas_titulacion_docente.py.

```
00005 def filter_all_asignaturas_titulacion_docente():
00006     """
00007 Crea un componente con un dropdown que contiene todas las asignaturas de la titulación del perfil
      "Docente"
00008     en la pestaña "Rendimiento académico general".
00009
00010     Returns:
00011     html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
00012     """
00013 return html.Div(
00014         [
00015             html.Br(),
00016             html.Label("Asignaturas"),
00017             dcc.Dropdown(
00018                 id="all-asignaturas-titulacion-docente",
00019                 searchable=True,
00020                 value=None,
00021                 clearable=True,
00022                 options=[],
00023                 maxHeight=300,
00024                 optionHeight=50,
00025                 persistence=True,
00026                 persistence_type="local",
00027                 multi=True,
00028             ),
```

```
            html.Button(
                "Seleccionar todo",
                id="select-all-asignaturas-titulacion-docente",
                className="button-select-all-filter",
                n_clicks=0,
            ),
        ]
    )
```

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

Definición en la línea 5 del archivo filter_all_curso_academico.py.

```
00005 def filter_all_curso_academico():
00006     """
00007 Crea un componente con un dropdown que contiene todos los cursos académicos del perfil "Docente"
00008     en la pestaña "Rendimiento académico general".
00009
00010     Returns:
00011     html.Div: Componente con un dropdown
00012     """
00013 return html.Div(
00014         [
00015             html.Br(),
00016             html.Label("Curso académico"),
00017             dcc.Dropdown(
00018                 id="all-cursos-academicos-docente",
```

```
                searchable=False,
                multi=False,
                clearable=False,
                options=[],
                value=None,
                maxHeight=300,
            ),
        ]
    )
```

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 5 del archivo filter_asignaturas_docente.py.

```
00005 def filter_asignaturas_docente():
00006     """
00007 Crea un componente con un dropdown que contiene las asignaturas del perfil "Docente"
00008     en la pestaña "Rendimiento académico personal".
00009
00010     Returns:
00011     html.Div: Componente con un dropdown
00012     """
00013
00014 return html.Div([
00015         html.Br(),
00016         html.Label("Asignaturas"),
00017         dcc.Dropdown(
00018             id="asignaturas-docente",
```

```
            searchable=True,
            value=None,
            clearable=True,
            options=[],
            maxHeight=300,
            optionHeight=50,
        )
    ])
```

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 5 del archivo filter_curso_academico_docente.py.

```
00005 def filter_curso_academico_docente():
00006     """
00007 Crea un componente con un dropdown que contiene los cursos académicos del perfil "Docente"
00008     en la pestaña "Rendimiento académico personal".
00009
00010     Returns:
00011     html.Div: Componente con un dropdown
00012     """
00013 return html.Div(
00014         [
00015             html.Br(),
00016             html.Label("Curso académico"),
00017             dcc.Dropdown(
00018                 id="curso-academico-docente",
```

```
                searchable=False,
                multi=True,
                clearable=True,
                options=[],
                value=None,
                maxHeight=300,
            ),
            html.Button(
                "Seleccionar todo",
                id="select-all-cursos-academicos-docente",
                className="button-select-all-filter",
                n_clicks=0,
            ),
        ]
    )
```

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 5 del archivo filter_titulacion_docente.py.

```
00005 def filter_titulacion_docente():
00006     """
00007 Crea un componente con un dropdown que contiene las titulaciones del perfil "Docente"
00008     en la pestaña "Rendimiento académico personal" y "Rendimiento académico general".
00009
00010     Returns:
00011     html.Div: Componente con un dropdown
00012     """
```

```
return html.Div(
        [
            html.Label("Titulación"),
            dcc.Dropdown(
                id="titulacion-docente",
                options=[],
                value=None,
                searchable=False,
                clearable=False,
                optionHeight=50,
                maxHeight=300,
            ),
            dcc.Store(id="selected-titulacion-docente-store", storage_type="local"),
        ]
    )
```

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 8 del archivo graphs_general_docente.py.

```
00008 def graphs_general_docente():
00009     """
00010 Retorna los gráficos de la pestaña "Rendimiento académico general"
00011     del perfil "Docente"
00012
00013     Returns:
00014     html.Div: Gráficos
00015     """
00016
00017 graphs_ids = [
00018 "calificaiones-cuali-all-asig-docente",
00019         "calificaiones-media-all-asig-docente",
00020     ]
00021
00022     item_class = "graph-item-general-docente"
00023     config = config_mode_bar_buttons_gestor
00024
00025     return html.Div(
00026         [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
00027         className="graphs-container-general-docente",
00028     )
```

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"
```

Definición en la línea 10 del archivo graphs_personal_docente.py.

```
00010 def graphs_personal_docente():
00011     """
00012 Retorna los gráficos de la pestaña "Rendimiento académico personal"
00013     del perfil "Docente"
00014     """
00015 graphs_ids = [
00016 "graph-alumnos-repetidores-nuevos",
00017         "graph-alumnos-matri-genero",
00018         "graph-alumnos-nota-media",
00019         "graph-alumnos-nota-cualitativa",
00020     ]
00021
00022     item_class = "graph-item-personal-docente"
00023     config = config_mode_bar_buttons_gestor
00024
00025     return html.Div(
00026         [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
00027         className="graphs-container-personal-docente",
00028     )
00029
```

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.recomendador_docente ( )
```

```
Retorna el componente del recomendador para docentes.

Returns:
html.Div: Componente del recomendador para docentes
```

Definición en la línea 3 del archivo recomendador_docente.py.

```
00003 def recomendador_docente():
00004     """
00005 Retorna el componente del recomendador para docentes.
00006
00007 Returns:
00008 html.Div: Componente del recomendador para docentes
00009 """
00010
00011 return html.Div([
00012         html.H1("Deserción universitaria: ¿Cómo pueden ayudar los docentes a prevenirla?",
      className='titulo-recomendador-docente'),
00013         html.Img(src='assets/images/docente_clase.jpg', className='imagen-recomendador-docente'),
00014         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'),
00015         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-docente'),
00016         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'),
00017         html.Br(),
00018         html.H2("¿Cómo pueden los docentes ayudar a prevenir la deserción universitaria?",
      className='sb-recomendador-docente'),
00019         html.Ul([
00020             html.Li("Identificar y apoyar a los estudiantes con dificultades académicas proporcionando
      tutorías y asesorías.", className='li-recomendador-docente'),
00021             html.Li("Fomentar un ambiente de clase inclusivo y motivador que promueva la participación
      y el interés por el aprendizaje.", className='li-recomendador-docente'),
00022             html.Li("Mantener una comunicación abierta y constante con los estudiantes para entender
      sus problemas y necesidades.", className='li-recomendador-docente'),
00023             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'),
00024             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'),
00025         ], className='ul-recomendador-docente'),
00026     ])
```

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 5 del archivo resumen_docente.py.

```
00005 def resumen_docente():
00006     """
00007 Crea el layout del resumen del docente
00008
00009 Returns:
00010 html.Div: Layout del resumen del docente
00011 """
00012 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.select_docente ( )
```

```
Crea el desplegable para seleccionar un docente.

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

Definición en la línea 5 del archivo select_docente.py.

```
00005 def select_docente():
00006     """
00007 Crea el desplegable para seleccionar un docente.
00008
00009 Returns:
00010 html.Div: Desplegable para seleccionar un docente
00011
00012 """
00013 return html.Div(
00014         [
00015             html.Div(
00016                 [
00017                     dcc.Dropdown(
00018                         options=[],
00019                         value=None,
00020                         id="docente-dropdown",
00021                         clearable=False,
00022                         placeholder="Selecciona un docente",
00023                     )
00024                 ],
00025                 className="select-docente",
00026             ),
00027         ]
00028     )
```

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.tabs_docente ( )
```

```
Crea las pestañas del perfil "Docente"

Returns:
html.Div: Pestañas del docente
```

Definición en la línea 5 del archivo tabs_docente.py.

```
00005 def tabs_docente():
00006     """
00007 Crea las pestañas del perfil "Docente"
00008
00009     Returns:
00010     html.Div: Pestañas del docente
00011     """
00012
00013 return html.Div(
00014         [
00015             dcc.Tabs(
00016                 id="tabs-docente",
00017                 value="rendimiento-academico-asignatura-tab",
00018                 children=[
00019                     dcc.Tab(
00020                         label="Rendimiento académico personal",
00021                         value="rendimiento-academico-asignatura-tab",
00022                     ),
00023                     dcc.Tab(
00024                         label="Rendimiento académico general",
00025                         value="rendimiento-academico-tab",
00026                     ),
00027                     dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
00028                 ],
00029                 className="tabs",
00030             ),
00031             html.Div(id="tabs-docente-content"),
00032             dcc.Store(id="selected-docente-store", storage_type="local"),
00033         ]
00034     )
```

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 5 del archivo filter_all_curso_academico_gestor.py.

```
00005 def filter_all_curso_academico_gestor():
00006     """
00007 Crea el filtro que muestra todos los cursos académicos disponibles
00008 para el perfil "Gestor" de la pestaña "Riesgo académico".
00009
00010     Returns:
00011     html.Div: Componente con el filtro de curso académico
00012     """
00013
00014 return html.Div(
00015         [
00016             html.Label("Curso académico"),
00017             dcc.Dropdown(
00018                 id="curso-all-academico-gestor",
00019                 searchable=True,
00020                 multi=True,
00021                 clearable=True,
00022                 options=[],
00023                 value=None,
00024                 maxHeight=200,
00025                 placeholder="Seleccione una opción",
00026             ),
00027             html.Button(
00028                 "Seleccionar todo",
00029                 id="select-all-curso-academico-button",
00030                 className="button-select-all-filter",
00031                 n_clicks=0,
00032             ),
00033             dcc.Store(id="curso-all-academico-gestor-store", storage_type="local"),
00034         ]
00035     )
```

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 5 del archivo filter_curso_academico_gestor.py.

```
00005 def filter_curso_academico_gestor():
00006     """
00007 Crea el filtro de curso académico para el perfil "Gestor" de la pestaña
00008     "Indicadores académicos".
00009
00010     Returns:
00011     html.Div: Componente con el filtro de curso académico
00012     """
00013 return html.Div(
00014         [
00015             html.Label("Curso académico"),
00016             dcc.Dropdown(
00017                 id="curso-academico-gestor",
00018                 searchable=True,
00019                 multi=False,
00020                 clearable=False,
00021                 options=[],
00022                 value=None,
00023                 maxHeight=300,
00024                 placeholder="Selecciona una opción",
00025             ),
00026         ]
00027     )
```

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 5 del archivo filter_titulaciones_gestor.py.

```
00005 def filter_titulaciones_gestor():
00006     """
00007 Crea el filtro de titulaciones para el perfil "Gestor" de la pestaña
00008     "Indicadores académicos".
00009
00010     Returns:
00011     html.Div: Componente con el filtro de titulaciones
00012     """
00013 return html.Div(
00014         [
00015             html.Br(),
00016             html.Label("Titulaciones"),
00017             dcc.Dropdown(
00018                 id="titulaciones-gestor",
00019                 searchable=True,
00020                 multi=True,
00021                 clearable=True,
00022                 options=[],
00023                 value=None,
00024                 maxHeight=200,
00025                 placeholder="Seleccione una opción",
00026             ),
00027             html.Button(
00028                 "Seleccionar todo",
00029                 id="select-all-titulaciones-gestor",
00030                 className="button-select-all-filter",
00031                 n_clicks=0,
00032             ),
00033         ]
00034     )
```

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 10 del archivo graphs_indicadores_gestor.py.

```
00010 def graphs_indicadores_gestor():
00011     """
00012 Retorna los gráficos de la pestaña "Indicadores académicos" del perfil
00013     "Gestor".Incluye un modal y una tabla para cada gráfico.
00014
00015     Returns:
00016     html.Div: Gráficos
00017     """
00018
00019 graphs_info = [
00020 {
00021 "graph_id": "nuevo-ingreso-genero-gestor",
00022             "modal_id": "modal-nuevo-ingreso-genero",
00023             "table_container_id": "table-container-nuevo-ingreso-genero",
00024             "download_button_id": "btn-descargar-nuevo-ingreso-genero",
00025             "view_data_button_id": "btn-ver-datos-nuevo-ingreso-genero",
00026         },
00027         {
00028             "graph_id": "nuevo_ingreso_nacionalidad-gestor",
00029             "modal_id": "modal-nuevo-ingreso-nacionalidad",
00030             "table_container_id": "table-container-nuevo-ingreso-nacionalidad",
00031             "download_button_id": "btn-descargar-nuevo-ingreso-nacionalidad",
00032             "view_data_button_id": "btn-ver-datos-nuevo-ingreso-nacionalidad",
00033         },
00034         {
00035             "graph_id": "egresados-genero-gestor",
00036             "modal_id": "modal-egresados-genero",
00037             "table_container_id": "table-container-egresados-genero",
00038             "download_button_id": "btn-descargar-egresados-genero",
00039             "view_data_button_id": "btn-ver-datos-egresados-genero",
```

```
        },
        {
            "graph_id": "egresados-nacionalidad-gestor",
            "modal_id": "modal-egresados-nacionalidad",
            "table_container_id": "table-container-egresados-nacionalidad",
            "download_button_id": "btn-descargar-egresados-nacionalidad",
            "view_data_button_id": "btn-ver-datos-egresados-nacionalidad",
        },
    ]

    graph_elements = [
        create_graph_with_table(
            graph_info["graph_id"],
            "graph-item-indicadores-gestor",
            config_mode_bar_buttons_gestor,
            {
                "modal_id": graph_info["modal_id"],
                "table_container_id": graph_info["table_container_id"],
                "download_button_id": graph_info["download_button_id"],
                "view_data_button_id": graph_info["view_data_button_id"],
            },
        )
        for graph_info in graphs_info
    ]

    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 8 del archivo graphs_resultados_gestor.py.

```
00008 def graphs_resultados_gestor():
00009     """
00010 Retorna los gráficos de la pestaña "Resultados académicos" del perfil
00011     "Gestor". Incluye un modal y una tabla para cada gráfico.
00012
00013     Returns:
00014     html.Div: Gráficos
00015     """
00016 graphs_info = [
00017 {
00018 "graph_id": "duración-estudios-nota-media-gestor",
00019             "modal_id": "modal-duracion-estudios",
00020             "table_container_id": "table-container-duracion-estudios",
00021             "download_button_id": "btn-descargar-csv-duracion-estudios",
00022             "view_data_button_id": "btn-ver-datos-duracion-estudios",
00023         },
00024         {
00025             "graph_id": "nota-acceso-titulacion",
00026             "modal_id": "modal-nota-acceso",
00027             "table_container_id": "table-container-nota-acceso",
00028             "download_button_id": "btn-descargar-csv-nota-acceso",
00029             "view_data_button_id": "btn-ver-datos-nota-acceso",
00030         },
00031     ]
00032
00033     graph_elements = [
00034         create_graph_with_table(
00035             graph_info["graph_id"],
00036             "graph-item-resultados-gestor",
00037             config_mode_bar_buttons_gestor,
00038             {
00039                 "modal_id": graph_info["modal_id"],
00040                 "table_container_id": graph_info["table_container_id"],
00041                 "download_button_id": graph_info["download_button_id"],
00042                 "view_data_button_id": graph_info["view_data_button_id"],
00043             },
00044         )
00045         for graph_info in graphs_info
00046     ]
00047
00048     return html.Div(graph_elements, className="graphs-container-resultados-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.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 8 del archivo graphs_riesgo_abandono_gestor.py.

```
00008 def graphs_riesgo_abandono_gestor():
00009     """
00010 Retorna los gráficos de la pestaña "Riesgo de abandono" del perfil
00011     "Gestor". Incluye un modal y una tabla para cada gráfico.
00012
00013     Returns:
00014     html.Div: Gráficos
00015     """
00016 graphs_info = [
00017 {
00018 "graph_id": "tasa-abandono-gestor",
00019             "modal_id": "modal-tasa-abandono",
00020             "table_container_id": "table-container-tasa-abandono",
00021             "download_button_id": "btn-descargar-tasa-abandono",
00022             "view_data_button_id": "btn-ver-datos-tasa-abandono",
00023         },
00024         {
00025             "graph_id": "tasa-graduacion-gestor",
00026             "modal_id": "modal-tasa-graduacion",
00027             "table_container_id": "table-container-tasa-graduacion",
00028             "download_button_id": "btn-descargar-tasa-graduacion",
00029             "view_data_button_id": "btn-ver-datos-tasa-graduacion",
00030         },
00031     ]
00032
```

```
    graph_elements = [
        create_graph_with_table(
            graph_info["graph_id"],
            "graph-item-riesgo-abandono-gestor",
            config_mode_bar_buttons_gestor,
            {
                "modal_id": graph_info["modal_id"],
                "table_container_id": graph_info["table_container_id"],
                "download_button_id": graph_info["download_button_id"],
                "view_data_button_id": graph_info["view_data_button_id"],
            },
        )
        for graph_info in graphs_info
    ]

    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 3 del archivo recomendador_gestor.py.

```
00003 def recomendador_gestor():
00004     """
00005 Crea un contenedor con información y recomendaciones sobre la deserción universitaria
00006 para los gestores de la sección "Recomendaciones" de la pestaña "Gestor".
00007
00008     Returns:
00009     html.Div: Contenedor con información y recomendaciones sobre la deserción universitaria
00010     """
00011 return html.Div([
00012         html.H1("Deserción universitaria", className='titulo-recomendador-gestor'),
00013         html.Img(src='assets/images/abandono-gestores.jpg', className='imagen-recomendador-gestor'),
00014         html.P("La deserción universitaria es un desafío significativo para las instituciones de
      educación superior. Los gestores tienen la responsabilidad de desarrollar e implementar políticas
      efectivas para reducirla.", className='p-recomendador-gestor'),
00015         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-gestor'),
00016         html.P("La deserción universitaria se refiere al abandono de los estudios por parte de los
      estudiantes antes de completar su formación. Este fenómeno tiene implicaciones negativas para la
      universidad y para la sociedad.", className='p-recomendador-gestor'),
00017         html.Br(),
00018         html.H2("Estrategias para prevenir la deserción universitaria",
      className='sb-recomendador-gestor'),
00019         html.Ul([
00020             html.Li("Desarrollar programas de becas y ayudas financieras para apoyar a los estudiantes
      con dificultades económicas.", className='li-recomendador-gestor'),
00021             html.Li("Implementar sistemas de tutoría y mentoría para ofrecer apoyo académico y
      emocional a los estudiantes.", className='li-recomendador-gestor'),
00022             html.Li("Establecer programas de orientación y adaptación para nuevos estudiantes,
      facilitando su integración en la vida universitaria.", className='li-recomendador-gestor'),
00023             html.Li("Promover la formación continua del personal docente en metodologías pedagógicas
      innovadoras.", className='li-recomendador-gestor'),
00024             html.Li("Utilizar sistemas de seguimiento y análisis de datos para identificar a
      estudiantes en riesgo y actuar de manera preventiva.", className='li-recomendador-gestor'),
00025             html.Li("Fomentar un clima institucional inclusivo y de apoyo que motive a los estudiantes
      a continuar con sus estudios.", className='li-recomendador-gestor'),
00026         ], className='ul-recomendador-gestor'),
00027     ])
```

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 5 del archivo resumen_gestor.py.

```
00005 def resumen_gestor():
00006     """
00007 Crea un contenedor para mostrar un resumen de los datos del gestor seleccionado.
00008
00009 Returns:
00010 html.Div: Contenedor para mostrar el resumen
00011 """
00012
00013 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 5 del archivo select_gestor.py.

```
00005 def select_gestor():
00006     """
00007 Crea un dropdown para seleccionar un gestor.
00008
00009 Returns:
00010 html.Div: Dropdown para seleccionar un gestor
00011 """
00012
00013 return html.Div(
00014         [
00015             html.Div(
00016                 [
00017                     dcc.Dropdown(
00018                         options=[],
00019                         value=None,
00020                         id="gestor-dropdown",
00021                         clearable=False,
00022                         placeholder="Seleccione un gestor",
00023                     )
00024                 ],
00025                 className="select-gestor",
00026             ),
00027         ]
00028     )
```

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 5 del archivo tabs_gestor.py.

```
00005 def tabs_gestor():
00006     """
00007 Crea las pestañas de la sección "Gestor". Cada pestaña contiene un conjunto de gráficos
00008     y tablas con los datos de los alumnos del perfil "Gestor".
00009
00010     Returns:
00011     html.Div: Pestañas de la sección "Gestor"
00012     """
00013
00014 return html.Div(
00015         [
00016             dcc.Tabs(
00017                 id="tabs-gestor",
00018                 value="indicadores-academicos-tab",
00019                 children=[
00020                     dcc.Tab(
00021                         label="Indicadores académicos",
00022                         value="indicadores-academicos-tab",
00023                     ),
00024                     dcc.Tab(
00025                         label="Resultados académicos", value="resultados-academicos-tab"
00026                     ),
00027                     dcc.Tab(label="Riesgo de abandono", value="riesgo-abandono-tab"),
00028                     dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
00029                 ],
00030                 className="tabs",
00031             ),
00032             html.Div(id="tabs-gestor-content"),
00033             dcc.Store(id="selected-gestor-store", storage_type="local"),
00034             dcc.Location(id="url", refresh=False),
00035         ]
00036     )
```

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 49 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 52 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 = 5
- int num_docentes = 1
- int num_gestores = 5
- 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 )
```

Definición en la línea 100 del archivo generate_synthetic_data.py.

```
00100 def generate_alumnos(n):
00101     ids = [fake.unique.bothify(text='???#####') for _ in range(n)]
00102     anios_nac = np.random.randint(1980, 2000, n)
00103     universidades = random.choices(list(universidades_dict.keys()), k=n)
00104     cod_universidades = [universidades_dict[uni] for uni in universidades]
00105     abandona = random.choices(['si', 'no'], k=n)
00106     data = {
00107         'id': ids,
00108         'anio_nac': anios_nac,
00109         'universidad': universidades,
00110         'cod_universidad': cod_universidades,
00111         'abandona': abandona
00112     }
00113     return pd.DataFrame(data)
00114
00115 # Función para generar cursos académicos en formato "YYYY/YYYY"
```

4.120.1.2. generate_asignaturas()

```
def data.generate_synthetic_data.generate_asignaturas (
            educacion,
            min_asignaturas = 38 )
```

Definición en la línea 74 del archivo generate_synthetic_data.py.

```
00074 def generate_asignaturas(educacion, min_asignaturas=38):
00075     asignaturas_dict = {}
00076     for rama, titulaciones in educacion.items():
00077         for titulacion, palabras in titulaciones.items():
00078             num_asignaturas = min(min_asignaturas, len(palabras))
00079             asignaturas = random.sample(palabras, num_asignaturas)
00080             asignaturas_dict[titulacion] = asignaturas
00081     return asignaturas_dict
00082
00083 # Generar diccionario de asignaturas
```

4.120.1.3. generate_asignaturas_matriculadas()

```
def data.generate_synthetic_data.generate_asignaturas_matriculadas (
            matricula_df,
            cod_asignaturas )
```

Definición en la línea 174 del archivo generate_synthetic_data.py.

```
00174 def generate_asignaturas_matriculadas(matricula_df, cod_asignaturas):
00175     asignaturas_data = []
00176     for i, row in tqdm(matricula_df.iterrows(), total=len(matricula_df),
      desc="asignaturas_matriculadas"):
00177         num_asignaturas = min(random.randint(1, 20), len(asignaturas_dict[row['titulacion']]))
00178         asignaturas = random.sample(asignaturas_dict[row['titulacion']], num_asignaturas)
00179         for asignatura in asignaturas:
00180             cod_asignatura = cod_asignaturas[asignatura]
00181             asignaturas_data.append({
00182                 'indice': len(asignaturas_data) + 1,
00183                 'asignatura': asignatura,
00184                 'cod_asignatura': cod_asignatura,
00185                 'cod_plan': row['cod_plan'],
00186                 'curso_aca': row['curso_aca'],
00187                 'id': row['id'],
00188                 'cod_tipologia': fake.bothify(text='T##'),
00189                 'plan': row['titulacion'],
00190                 'tipologia': fake.word(),
00191                 'universidad': row['universidad'],
00192                 'cod_universidad': row['cod_universidad']
00193             })
00194     return pd.DataFrame(asignaturas_data)
00195
00196 # Función para generar datos sintéticos para la tabla 'lineas_actas'
```

4.120.1.4. generate_curso_aca()

```
def data.generate_synthetic_data.generate_curso_aca (
            start_year,
            end_year )
```

Definición en la línea 116 del archivo generate_synthetic_data.py.

```
00116 def generate_curso_aca(start_year, end_year):
00117     return [f"{year}/{year+1}" for year in range(start_year, end_year)]
00118
00119 # 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 )
```

Definición en la línea 227 del archivo generate_synthetic_data.py.

```
00227 def generate_docentes(n, universidades, cod_asignaturas, asignaturas_dict):
00228     docentes_data = []
00229
00230     for i in tqdm(range(n), desc="docentes"):
00231         id_docente = fake.bothify(text='D#####')
00232         universidad, cod_universidad = random.choice(list(universidades.items()))
00233
00234         num_asignaturas = random.randint(1, 5)
00235         asignaturas_seleccionadas = random.sample(list(cod_asignaturas.items()), num_asignaturas)
00236
00237         for asignatura, cod_asignatura in asignaturas_seleccionadas:
00238             # Encontrar la titulación correspondiente a la asignatura
00239             titulacion = next(titulacion for titulacion, asignaturas in asignaturas_dict.items() if
      asignatura in asignaturas)
00240             cod_plan = cod_plan_dict[titulacion]
00241             docentes_data.append({
00242                 'indice': len(docentes_data) + 1,
00243                 'id_docente': id_docente,
00244                 'cod_universidad': cod_universidad,
00245                 'universidad': universidad,
00246                 'titulacion': titulacion,
00247                 'cod_plan': cod_plan,
00248                 'cod_asignatura': cod_asignatura,
00249                 'asignatura': asignatura,
00250                 'curso_aca': random.choice(cursos_academicos)
00251             })
00252
00253     return pd.DataFrame(docentes_data)
00254
00255 # Función para generar datos sintéticos para la tabla 'ebau_prueba'
```

4.120.1.6. generate_ebau_prueba()

```
def data.generate_synthetic_data.generate_ebau_prueba (
            n,
            alumnos_ids,
            universidades )
```

Definición en la línea 256 del archivo generate_synthetic_data.py.

```
00256 def generate_ebau_prueba(n, alumnos_ids, universidades):
00257     assert n == len(alumnos_ids), "El número de registros debe ser igual al número de alumnos"
00258
00259     # Generar datos para cada campo
00260     data = {
00261         'indice': range(1, n + 1),
00262         'conv': [fake.bothify(text='C##') for _ in range(n)],
00263         'curso_aca': np.random.choice(cursos_academicos, n),
00264         'especialidad': [fake.word() for _ in range(n)],
00265         'id': alumnos_ids,
00266         'nota_bach': np.round(np.random.uniform(5, 10, n), 2),
00267         'nota_def': np.round(np.random.uniform(5, 10, n), 2),
00268         'nota_prue': np.round(np.random.uniform(5, 10, n), 2),
00269         'universidad': np.random.choice(list(universidades.keys()), n),
00270     }
00271     data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]
00272
00273     return pd.DataFrame(data)
00274
00275 # Función para generar datos sintéticos para la tabla 'egresados'
```

4.120.1.7. generate_egresados()

```
def data.generate_synthetic_data.generate_egresados (
            alumnos_df,
            matricula_df,
            lineas_actas_df )
```

Definición en la línea 276 del archivo generate_synthetic_data.py.

```
00276 def generate_egresados(alumnos_df, matricula_df, lineas_actas_df):
00277     aprobados = lineas_actas_df[lineas_actas_df['calif_numerica'] >= 5].groupby('id').size()
00278     egresados_ids = aprobados[aprobados >= 38].index.tolist()
00279     alumnos_no_abandona = alumnos_df[alumnos_df['abandona'] == 'no']['id'].tolist()
00280     egresados_ids = [id for id in egresados_ids if id in alumnos_no_abandona]
00281
00282     egresados_data = []
00283     for i, id_alumno in tqdm(enumerate(egresados_ids), total=len(egresados_ids), desc="egresados"):
00284         universidad, cod_universidad = alumnos_df.loc[alumnos_df['id'] == id_alumno, ['universidad',
      'cod_universidad']].values[0]
00285         cod_plan = matricula_df.loc[matricula_df['id'] == id_alumno, 'cod_plan'].values[0]
00286         ultimo_curso_aca = matricula_df.loc[matricula_df['id'] == id_alumno, 'curso_aca'].max()
00287         egresados_data.append({
00288             'indice': i + 1,
00289             'cod_plan': cod_plan,
00290             'curso_aca': ultimo_curso_aca,
00291             'id': id_alumno,
00292             'nota_media': round(random.uniform(5, 10), 2),
00293             'plan': fake.word(),
00294             'universidad': universidad,
00295             'cod_universidad': cod_universidad
00296         })
00297     return pd.DataFrame(egresados_data)
00298
00299 # Función para generar datos sintéticos para la tabla 'gestores'
```

4.120.1.8. generate_gestores()

```
def data.generate_synthetic_data.generate_gestores (
            n,
            universidades )
```

Definición en la línea 300 del archivo generate_synthetic_data.py.

```
00300 def generate_gestores(n, universidades):
00301     data = {
00302         'gestor_id': range(1, n + 1),
00303         'universidad': np.random.choice(list(universidades.keys()), n)
00304     }
00305     data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]
00306     return pd.DataFrame(data)
00307
00308
00309 print("Generando datos....")
```

4.120.1.9. generate_lineas_actas()

```
def data.generate_synthetic_data.generate_lineas_actas (
            asignaturas_matriculadas_df )
```

Definición en la línea 197 del archivo generate_synthetic_data.py.

```
00197 def generate_lineas_actas(asignaturas_matriculadas_df):
00198     actas_data = []
00199     for i, row in tqdm(asignaturas_matriculadas_df.iterrows(), total=len(asignaturas_matriculadas_df),
      desc="líneas_actas"):
00200         calif_numerica = round(random.uniform(0, 10), 2)
00201         if calif_numerica < 5:
```

```
            calif = 'Suspenso'
        elif calif_numerica < 7:
            calif = 'Aprobado'
        elif calif_numerica < 9:
            calif = 'Notable'
        else:
            calif = 'Sobresaliente'

        actas_data.append({
            'indice': len(actas_data) + 1,
            'asignatura': row['asignatura'],
            'calif_numerica': calif_numerica,
            'calif': calif,
            'cod_asig': row['cod_asignatura'],
            'cod_plan': row['cod_plan'],
            'conv': fake.bothify(text='C##'),
            'curso_aca': row['curso_aca'],
            'id': row['id'],
            'plan': row['plan'],
            'universidad': row['universidad'],
            'cod_universidad': row['cod_universidad']
        })
    return pd.DataFrame(actas_data)

# Función para generar datos sintéticos para la tabla 'docentes'
```

4.120.1.10. generate_matricula()

```
def data.generate_synthetic_data.generate_matricula (
            alumnos_df )
```

Definición en la línea 123 del archivo generate_synthetic_data.py.

```
def generate_matricula(alumnos_df):
    n = len(alumnos_df)
    num_matriculas = np.random.randint(1, 6, n)
    matricula_data = []
    for i, row in tqdm(alumnos_df.iterrows(), total=n, desc="matrícula"):
        titulacion = random.choice(list(asignaturas_dict.keys()))
        # Encontrar la rama correspondiente a la titulación
        rama = next(rama for rama, titulaciones in educacion.items() if titulacion in titulaciones)

        common_data = {
            'municipio': fake.city(),
            'nacionalidad': fake.country(),
            'provincia': fake.state(),
            'rama': rama,
            'sexo': random.choice(['Masculino', 'Femenino']),
            'titulacion': titulacion,
            'nombre_plan_propio': fake.word(),
            'universidad': row['universidad'],
            'cod_universidad': row['cod_universidad'],
            'cod_tipo_matricula': fake.bothify(text='T##'),
            'tipo_matricula': fake.word(),
            'cod_plan': cod_plan_dict[titulacion],
        }
        cursos_aca_usados = set()
        for j in range(num_matriculas[i]):
            curso_aca = random.choice([ca for ca in cursos_academicos if ca not in cursos_aca_usados])
            cursos_aca_usados.add(curso_aca)
            nuevo_ingreso = 'si' if j == 0 else 'no'
            matricula_data.append({
                'indice': len(matricula_data) + 1,
                'ambito_isced': fake.bothify(text='IS##'),
                'cod_plan': common_data['cod_plan'],
                'cod_mec': fake.bothify(text='MEC##'),
                'curso_aca': curso_aca,
                'municipio': common_data['municipio'],
                'nacionalidad': common_data['nacionalidad'],
                'nuevo_ingreso': nuevo_ingreso,
                'provincia': common_data['provincia'],
                'rama': common_data['rama'],
                'sexo': common_data['sexo'],
                'titulacion': common_data['titulacion'],
                'nombre_plan_propio': common_data['nombre_plan_propio'],
                'universidad': common_data['universidad'],
                'cod_universidad': common_data['cod_universidad'],
                'cod_tipo_matricula': common_data['cod_tipo_matricula'],
                'tipo_matricula': common_data['tipo_matricula'],
```

```
                'id': row['id']
            })
    return pd.DataFrame(matricula_data)

# Función para generar datos sintéticos para la tabla 'asignaturas_matriculadas'
```

4.120.1.11. generate_unique_cod_asignaturas()

```
def data.generate_synthetic_data.generate_unique_cod_asignaturas (
            asignaturas_dict )
```

Definición en la línea 87 del archivo generate_synthetic_data.py.

```
def generate_unique_cod_asignaturas(asignaturas_dict):
    cod_asignaturas = {}
    for titulacion, asignaturas in asignaturas_dict.items():
        for asignatura in asignaturas:
            cod_asignatura = asignatura[:4].upper() + fake.bothify(text='#####')
            cod_asignaturas[asignatura] = cod_asignatura
    return cod_asignaturas

# Generar un diccionario con códigos únicos para cada asignatura
```

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 319 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 84 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 321 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 96 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 97 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 120 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 323 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 324 del archivo generate_synthetic_data.py.

4.120.2.9. educacion

```
dictionary data.generate_synthetic_data.educacion
```

Definición en la línea 12 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 325 del archivo generate_synthetic_data.py.

4.120.2.11. fake

```
data.generate_synthetic_data.fake = Faker('es_ES')
```

Definición en la línea 9 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 326 del archivo generate_synthetic_data.py.

4.120.2.13. index

```
data.generate_synthetic_data.index
```

Definición en la línea 329 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 322 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 320 del archivo generate_synthetic_data.py.

4.120.2.16. num_alumnos

```
int data.generate_synthetic_data.num_alumnos = 5
```

Definición en la línea 310 del archivo generate_synthetic_data.py.

4.120.2.17. num_docentes

```
int data.generate_synthetic_data.num_docentes = 1
```

Definición en la línea 311 del archivo generate_synthetic_data.py.

4.120.2.18. num_gestores

```
int data.generate_synthetic_data.num_gestores = 5
```

Definición en la línea 312 del archivo generate_synthetic_data.py.

4.120.2.19. universidades_dict

```
dictionary data.generate_synthetic_data.universidades_dict
```

Valor inicial:

```
{
    'Universidad de Madrid': 'UDM01',
    'Universidad de Barcelona': 'UDB02',
    'Universidad de La Laguna': 'ULL03',
}
```

Definición en la línea 67 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 cursos_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 27 del archivo queries.py.

```
00027 def alumnos_all():
00028     query = queries["alumnado"]["common"]["alumnos_all"]
00029     return check_data(query, {})
00030
00031
00032 @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 379 del archivo queries.py.

```
00379 def alumnos_egresados_genero_titulacion(cod_universidad, curso_academico, titulaciones):
00380     query = queries["gestor"]["graphs"]["indicadores"][
00381         "alumnos_egresados_genero_titulacion"
00382     ]
00383     params = {
00384         "cod_universidad": cod_universidad,
00385         "curso_academico": curso_academico,
00386         "titulaciones": titulaciones,
00387     }
00388
00389     return check_data(query, params)
00390
00391
00392 @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 393 del archivo queries.py.

```
00395 ):
00396     query = queries["gestor"]["graphs"]["indicadores"][
00397         "alumnos_egresados_nacionalidad_titulacion"
00398     ]
00399     params = {
00400         "cod_universidad": cod_universidad,
00401         "curso_academico": curso_academico,
00402         "titulaciones": titulaciones,
00403     }
00404
00405     return check_data(query, params)
00406
00407
00408 @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 257 del archivo queries.py.

```
00257 def alumnos_genero_docente(id_docente, asignaturas, curso_academico):
00258     query = queries["docente"]["graphs"]["personal"]["alumnos_genero_docente"]
00259     params = {
00260         "id_docente": id_docente,
00261         "asignaturas": asignaturas,
00262         "curso_academico": curso_academico,
00263     }
00264
00265     return check_data(query, params)
00266
00267
00268 @cache_query
```

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:

4.121.1.5. alumnos_nota_cualitativa_docente()

```
def data.queries.alumnos_nota_cualitativa_docente (
              asignaturas,
              curso_academico )
```

Definición en la línea 277 del archivo queries.py.

```
00277 def alumnos_nota_cualitativa_docente(asignaturas, curso_academico):
00278     query = queries["docente"]["graphs"]["personal"]["alumnos_nota_cualitativa_docente"]
00279     params = {"asignaturas": asignaturas, "curso_academico": curso_academico}
00280
00281     return check_data(query, params)
00282
00283
00284 @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 269 del archivo queries.py.

```
00269 def alumnos_nota_media_docente(asignaturas, curso_academico):
00270     query = queries["docente"]["graphs"]["personal"]["alumnos_nota_media_docente"]
00271     params = {"asignaturas": asignaturas, "curso_academico": curso_academico}
00272
00273     return check_data(query, params)
00274
00275
00276 @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 363 del archivo queries.py.

```
00365 ):
00366     query = queries["gestor"]["graphs"]["indicadores"][
00367         "alumnos_nuevo_ingreso_genero_titulacion"
00368     ]
00369     params = {
00370         "curso_academico": curso_academico,
00371         "titulaciones": titulaciones,
00372         "cod_universidad": cod_universidad,
00373     }
00374
00375     return check_data(query, params)
00376
00377
00378 @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 409 del archivo queries.py.

```
00411 ):
00412     query = queries["gestor"]["graphs"]["indicadores"][
00413         "alumnos_nuevo_ingreso_nacionalidad_titulacion"
00414     ]
00415     params = {
00416         "cod_universidad": cod_universidad,
00417         "curso_academico": curso_academico,
00418         "titulaciones": titulaciones,
00419     }
00420
00421     return check_data(query, params)
00422
00423
00424 @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 207 del archivo queries.py.

```
00207 def alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas):
00208     query = queries["docente"]["graphs"]["personal"]["alumnos_repetidores_nuevos"]
00209     params = {
00210         "docente_id": docente_id,
00211         "curso_academico": curso_academico,
00212         "asignaturas": asignaturas,
00213     }
00214
00215     return check_data(query, params)
00216
00217
00218 @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 293 del archivo queries.py.

```
def asignaturas_actas_titulacion(titulacion, curso_academico):
    query = queries["docente"]["filters"]["asignaturas_actas_titulacion"]
    params = {"titulacion": titulacion, "curso_academico": curso_academico}

    return check_data(query, params)


@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 219 del archivo queries.py.

```
def asignaturas_docente(id_docente, titulacion):
    query = queries["docente"]["filters"]["asignaturas_docente"]
    params = {"id_docente": id_docente, "titulacion": titulacion}

    return check_data(query, params)


@cache_query
```

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:

4.121.1.12. asignaturas_matriculadas()

```
def data.queries.asignaturas_matriculadas (
            alumno_id,
            curso_academico,
            titulacion )
```

Definición en la línea 70 del archivo queries.py.

```
00070 def asignaturas_matriculadas(alumno_id, curso_academico, titulacion):
00071     query = queries["alumnado"]["filters"]["asinaturas_matriculadas"]
00072     params = {
00073         "alumno_id": alumno_id,
00074         "curso_academico": curso_academico,
00075         "titulacion": titulacion,
00076     }
00077
00078     return check_data(query, params)
00079
00080
00081 @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 128 del archivo queries.py.

```
00128 def asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion):
00129     query = queries["alumnado"]["graphs"]["personal"][
00130         "asignaturas_matriculadas_y_superadas"
00131     ]
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:

4.121.1.14. asignaturas_superadas()

```
def data.queries.asignaturas_superadas (
              alumno_id,
              curso_academico,
              titulacion )
```

Definición en la línea 90 del archivo queries.py.

```
00090 def asignaturas_superadas(alumno_id, curso_academico, titulacion):
00091     query = queries["alumnado"]["graphs"]["personal"][
00092         "curso_academico_asignaturas_superadas"
```

```
    ]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@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 142 del archivo queries.py.

```
):
    query = queries["alumnado"]["graphs"]["general"][
        "asignaturas_superadas_media_abandono"
    ]
    params = {
        "curso_academico": curso_academico,
        "asignaturas_matriculadas": asignaturas_matriculadas,
        "titulacion": titulacion,
        "cod_universidad": cod_universidad,
    }

    return check_data(query, params)


@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 (
            func )
```

Definición en la línea 18 del archivo queries.py.

```
00018 def cache_query(func):
00019     @lru_cache(maxsize=32)
00020     def wrapper(*args, **kwargs):
00021         return func(*args, **kwargs)
00022
00023     return wrapper
00024
00025
00026 @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 301 del archivo queries.py.

```
00301 def calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignaturas):
00302     query = queries["docente"]["graphs"]["general"]["calif_all_cualitativa_asignaturas"]
00303     params = {
00304         "titulacion": titulacion,
00305         "curso_academico": curso_academico,
00306         "asignaturas": asignaturas,
```

```
    }

    return check_data(query, params)


@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 174 del archivo queries.py.

```
):
    query = queries["alumnado"]["graphs"]["general"][
        "calif_cualitativa_alumno_asignaturas"
    ]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "asignaturas_matriculadas": asignaturas_matriculadas,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@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 104 del archivo queries.py.

```
00104 def calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion):
00105     query = queries["alumnado"]["graphs"]["personal"]["calif_cualitativa_asginatura"]
00106     params = {
00107         "alumno_id": alumno_id,
00108         "curso_academico": curso_academico,
00109         "titulacion": titulacion,
00110     }
00111
00112     return check_data(query, params)
00113
00114
00115 @cache_query
```

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:

4.121.1.20. calif_cualitativa_comparativa()

```
def data.queries.calif_cualitativa_comparativa (
              curso_academico,
              asignaturas_matriculadas,
              titulacion,
              cod_universidad )
```

Definición en la línea 159 del archivo queries.py.

```
00161 ):
00162     query = queries["alumnado"]["graphs"]["general"]["calif_cualitativa_comparativa"]
00163     params = {
00164         "curso_academico": curso_academico,
00165         "asignaturas_matriculadas": asignaturas_matriculadas,
00166         "titulacion": titulacion,
00167         "cod_universidad": cod_universidad,
00168     }
00169
00170     return check_data(query, params)
00171
00172
00173 @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 (
              titulacion,
              curso_academico,
              asignaturas )
```

Definición en la línea 313 del archivo queries.py.

```
00313 def calif_media_asignaturas(titulacion, curso_academico, asignaturas):
00314     query = queries["docente"]["graphs"]["general"]["calif_media_asignaturas"]
00315     params = {
00316         "titulacion": titulacion,
00317         "curso_academico": curso_academico,
00318         "asignaturas": asignaturas,
00319     }
00320
00321     return check_data(query, params)
00322
00323
00324 @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 116 del archivo queries.py.

```
00116 def calif_numerica_asignatura(alumno_id, curso_academico, titulacion):
00117     query = queries["alumnado"]["graphs"]["personal"]["calif_numerica_asignatura"]
00118     params = {
```

```
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@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 )
```

Definición en la línea 8 del archivo queries.py.

```
8 def check_data(query, params):
9     try:
10        data = db.execute_query(query, params)
11    except Exception as e:
12        print("Query execution failed: ", e)
13        return []
14    return data
15
16
17 # Decorador para cachear las queries y evitar hacer consultas innecesarias
```

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 285 del archivo queries.py.

```
00285 def curso_academico_actas_titulacion(titulacion):
00286     query = queries["docente"]["filters"]["curso_academico_actas_titulacion"]
00287     params = {"titulacion": titulacion}
00288
00289     return check_data(query, params)
00290
00291
00292 @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 62 del archivo queries.py.

```
00062 def curso_academico_alumnado(alumno_id, titulacion):
00063     query = queries["alumnado"]["filters"]["curso_academico_alumnado"]
00064     params = {"alumno_id": alumno_id, "titulacion": titulacion}
00065
00066     return check_data(query, params)
00067
00068
00069 @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 227 del archivo queries.py.

```
00227 def curso_academico_docente(id_docente, asignatura):
00228     query = queries["docente"]["filters"]["curso_academico_docente"]
00229     params = {"id_docente": id_docente, "asignatura": asignatura}
00230
00231     return check_data(query, params)
00232
00233
00234 @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 347 del archivo queries.py.

```
00347 def curso_academico_universidad(cod_universidad):
00348     query = queries["gestor"]["filters"]["curso_academico_universidad"]
00349     params = {"cod_universidad": cod_universidad}
00350
00351     return check_data(query, params)
00352
00353
00354 @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 443 del archivo queries.py.

```
00443 def cursos_academicos_egresados(cod_universidad):
00444     query = queries["gestor"]["common"]["cursos_academicos_de_egresados"]
00445     params = {"cod_universidad": cod_universidad}
00446
00447     return check_data(query, params)
00448
```

```
00449
00450 @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 486 del archivo queries.py.

```
00486 def data_for_model():
00487     query = queries["alumnado"]["common"]["data_for_model"]
00488     data = check_data(query, {})
00489
00490     if not data:
00491         return []
00492
00493     df = pd.DataFrame(data)
00494
00495     # Ajustar los tipos de datos de las columnas
00496     df = df.astype({
00497         'id': str,
00498         'anio_nac': int,
00499         'nacionalidad': str,
00500         'sexo': str,
00501         'titulacion': str,
00502         'nota_def_ebau': float,
00503         'nota_media': float,
00504         'abandona': str
00505     })
00506
00507     return df
00508
00509 @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 251 del archivo queries.py.

```
00251 def docentes_all():
00252     query = queries["docente"]["common"]["docentes_all"]
00253     return check_data(query, {})
00254
00255
00256 @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 433 del archivo queries.py.

```
00433 def duracion_media_estudios_nota_gestor(cod_universidad):
00434     query = queries["gestor"]["graphs"]["resultados"][
00435         "duracion_media_estudios_nota_gestor"
00436     ]
00437     params = {"cod_universidad": cod_universidad}
00438
00439     return check_data(query, params)
00440
00441
00442 @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 325 del archivo queries.py.

```
00325 def gestores_all():
00326     query = queries["gestor"]["common"]["gestores_all"]
00327     return check_data(query, {})
00328
00329
00330 @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 425 del archivo queries.py.

```
00425 def nota_media_acceso_titulacion(cod_universidad):
00426     query = queries["gestor"]["graphs"]["resultados"]["nota_media_acceso_titulacion"]
00427     params = {"cod_universidad": cod_universidad}
00428
00429     return check_data(query, params)
00430
00431
00432 @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 49 del archivo queries.py.

```
00049 def nota_media_alumno_titulacion(alumno_id, titulacion):
00050     query = queries["alumnado"]["common"]["nota_media_alumno_titulacion"]
00051     params = {"alumno_id": alumno_id, "titulacion": titulacion}
00052
00053     data = check_data(query, params)
00054
00055     if not data:
00056         return "No disponible"
00057
00058     return round(data[0][0], 2)
00059
00060
00061 @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 191 del archivo queries.py.

```
00193 ):
00194     query = queries["alumnado"]["graphs"]["general"]["nota_media_general_mi_nota"]
00195     params = {
00196         "curso_academico": curso_academico,
00197         "asignaturas_matriculadas": asignaturas_matriculadas,
00198         "alumno_id": alumno_id,
00199         "titulacion": titulacion,
00200         "cod_universidad": cod_universidad,
00201     }
00202
00203     return check_data(query, params)
00204
00205
00206 @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 331 del archivo queries.py.

```
00331 def numero_alumnos_matriculados_universidad(universidad):
00332     query = queries["gestor"]["common"]["numero_alumnos_matriculados_universidad"]
```

```
    params = {"universidad": universidad}

    return check_data(query, params)


@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 33 del archivo queries.py.

```
33 def resumen_alumno(alumno_id, titulacion):
    query = queries["alumnado"]["common"]["resumen_alumno"]
    params = {"alumno_id": alumno_id, "titulacion": titulacion}

    return check_data(query, params)


@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 243 del archivo queries.py.

```
00243 def resumen_docente(id_docente, titulacion):
00244     query = queries["docente"]["common"]["resumen_docente"]
00245     params = {"id_docente": id_docente, "titulacion": titulacion}
00246
00247     return check_data(query, params)
00248
00249
00250 @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 41 del archivo queries.py.

```
00041 def resumen_gestor(gestor_id):
00042     query = queries["gestor"]["common"]["resumen_gestor"]
00043     params = {"gestor_id": gestor_id}
00044
00045     return check_data(query, params)
00046
```

```
00047
00048 @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 451 del archivo queries.py.

```
00451 def tasa_abandono_titulacion_gestor(cod_universidad, curso_academico):
00452     query = queries["gestor"]["graphs"]["riesgo_abandono"][
00453         "tasa_abandono_titulacion_gestor"
00454     ]
00455     params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00456
00457     return check_data(query, params)
00458
00459
00460 @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 461 del archivo queries.py.

```
00461 def tasa_graduacion_titulacion_gestor(cod_universidad, curso_academico):
00462     query = queries["gestor"]["graphs"]["riesgo_abandono"][
00463         "tasa_graduacion_titulacion_gestor"
00464     ]
00465     params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00466
00467     return check_data(query, params)
00468
00469
00470 @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 82 del archivo queries.py.

```
00082 def titulacion_alumnado(alumno_id):
00083     query = queries["alumnado"]["filters"]["titulacion_alumnado"]
```

```
    params = {"alumno_id": alumno_id}

    return check_data(query, params)


@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 235 del archivo queries.py.

```
def titulacion_docente(id_docente):
    query = queries["docente"]["filters"]["titulacion_docente"]
    params = {"id_docente": id_docente}

    return check_data(query, params)


@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 (
            cod_universidad,
            curso_academico )
```

Definición en la línea 355 del archivo queries.py.

```
00355 def titulaciones_universidad_gestor(cod_universidad, curso_academico):
00356     query = queries["gestor"]["filters"]["titulaciones_universidad_gestor"]
00357     params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}
00358
00359     return check_data(query, params)
00360
00361
00362 @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 471 del archivo queries.py.

```
00471 def universidad_alumno(alumno_id):
00472     query = queries["alumnado"]["common"]["universidad_alumno"]
00473     params = {"alumno_id": alumno_id}
00474
00475     return check_data(query, params)
00476
00477
00478 @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 479 del archivo queries.py.

```
00479 def universidades_docente(id_docente):
00480     query = queries["docente"]["common"]["universidades_docente"]
```

```
    params = {"id_docente": id_docente}

    return check_data(query, params)

@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 339 del archivo queries.py.

```
def universidades_gestor(gestor_id):
    query = queries["gestor"]["common"]["universidades_gestor"]
    params = {"gestor_id": gestor_id}

    return check_data(query, params)


@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 2 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 ( )

Retorna el layout del perfil "Alumno"

Returns:
html.Div: Layout del alumno
```

Definición en la línea 5 del archivo alumno_layout.py.

```
00005 def alumno_layout():
00006     """
00007 Retorna el layout del perfil "Alumno"
00008
00009     Returns:
00010     html.Div: Layout del alumno
00011     """
00012 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 ( )
```

```
Retorna el layout del perfil "Docente"

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

Definición en la línea 5 del archivo docente_layout.py.

```
00005 def docente_layout():
00006     """
00007 Retorna el layout del perfil "Docente"
00008
00009     Returns:
00010     html.Div: Layout del docente
00011     """
00012 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 ( )
```

```
Retorna el layout del perfil "Gestor"

Returns:
html.Div: Layout del gestor
```

Definición en la línea 5 del archivo gestor_layout.py.

```
00005 def gestor_layout():
00006     """
00007 Retorna el layout del perfil "Gestor"
00008
00009     Returns:
00010     html.Div: Layout del gestor
00011     """
00012 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_ebau', '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 = {}
- total
- desc
- grid_search = GridSearchCV(model, param_grids[key], cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)
- voting_clf
- string joblib_file = "trained_model.pkl"
- y_pred_voting = voting_clf.predict(X_test)
- predicciones
- 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 97 del archivo model.py.

4.127.1.2. columnas_categoricas

```
list model.columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
```

Definición en la línea 27 del archivo model.py.

4.127.1.3. columnas_numericas

```
list model.columnas_numericas = ['nota_def_ebau', 'nota_media', 'edad_actual']
```

Definición en la línea 28 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 141 del archivo model.py.

4.127.1.5. current_year

```
model.current_year = datetime.now().year
```

Definición en la línea 20 del archivo model.py.

4.127.1.6. data

```
model.data = load_data_for_model()
```

Definición en la línea 18 del archivo model.py.

4.127.1.7. desc

```
model.desc
```

Definición en la línea 98 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 100 del archivo model.py.

4.127.1.9. id_test

```
model.id_test
```

Definición en la línea 44 del archivo model.py.

4.127.1.10. id_train

```
model.id_train
```

Definición en la línea 44 del archivo model.py.

4.127.1.11. ids

```
model.ids = data['id']
```

Definición en la línea 41 del archivo model.py.

4.127.1.12. joblib_file

```
string model.joblib_file = "trained_model.pkl"
```

Definición en la línea 116 del archivo model.py.

4.127.1.13. modelos

```
dictionary model.modelos
```

Valor inicial:

```
{
    'rf': RandomForestClassifier(random_state=42, n_jobs=-1),
    'lr': LogisticRegression(max_iter=1000, random_state=42),
    'gb': GradientBoostingClassifier(random_state=42),
    'ab': AdaBoostClassifier(random_state=42),
    'xgb': xgb.XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss')
}
```

Definición en la línea 59 del archivo model.py.

4.127.1.14. param_grids

```
dictionary model.param_grids
```

Valor inicial:

```
{
    'rf': {
        'n_estimators': [100, 200, 300],
        'max_depth': [10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'bootstrap': [True, False]
    },
    'lr': {
        'C': [0.01, 0.1, 1.0, 10.0],
        'solver': ['liblinear', 'lbfgs']
    },
    'gb': {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.01, 0.05, 0.1],
        'max_depth': [3, 5, 7]
    },
    'ab': {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.05, 0.1]
    },
    'xgb': {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.01, 0.05, 0.1],
        'max_depth': [3, 5, 7]
    }
}
```

Definición en la línea 68 del archivo model.py.

4.127.1.15. poly

```
model.poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
```

Definición en la línea 54 del archivo model.py.

4.127.1.16. predicciones

```
model.predicciones
```

Valor inicial:

```
00001 =  pd.DataFrame({
00002     'id': id_test,
00003     'abandona_real': y_test,
00004     'abandona_predicho': y_pred_voting
00005 })
```

Definición en la línea 131 del archivo model.py.

4.127.1.17. preprocessor

```
model.preprocessor
```

Valor inicial:

```
00001 =  ColumnTransformer(
00002     transformers=[
00003         ('num', Pipeline([('imputer', SimpleImputer(strategy='mean')), ('scaler', StandardScaler())]),
      columnas_numericas),
00004         ('cat', Pipeline([('imputer', SimpleImputer(strategy='constant', fill_value='Desconocido')),
00005                           ('encoder', OneHotEncoder(handle_unknown='ignore'))]), columnas_categoricas)
00006     ])
```

Definición en la línea 31 del archivo model.py.

4.127.1.18. random_state

```
model.random_state
```

Definición en la línea 44 del archivo model.py.

4.127.1.19. smote

```
model.smote = SMOTE(random_state=42)
```

Definición en la línea 50 del archivo model.py.

4.127.1.20. stratify

```
model.stratify
```

Definición en la línea 44 del archivo model.py.

4.127.1.21. test_size

```
model.test_size
```

Definición en la línea 44 del archivo model.py.

4.127.1.22. total

```
model.total
```

Definición en la línea 98 del archivo model.py.

4.127.1.23. voting_clf

```
model.voting_clf
```

Valor inicial:

```
VotingClassifier(
    estimators=[(key, best_estimators[key]) for key in best_estimators],
    voting='soft',
    n_jobs=-1
)
```

Definición en la línea 106 del archivo model.py.

4.127.1.24. X

```
model.X = data[columnas_categoricas + columnas_numericas]
```

Definición en la línea 39 del archivo model.py.

4.127.1.25. X_test

```
model.X_test = preprocessor.transform(X_test)
```

Definición en la línea 44 del archivo model.py.

4.127.1.26. X_train

```
model.X_train = preprocessor.fit_transform(X_train)
```

Definición en la línea 44 del archivo model.py.

4.127.1.27. y

```
model.y = data['abandona']
```

Definición en la línea 40 del archivo model.py.

4.127.1.28. y_pred_voting

```
model.y_pred_voting = voting_clf.predict(X_test)
```

Definición en la línea 121 del archivo model.py.

4.127.1.29. y_test

```
model.y_test
```

Definición en la línea 44 del archivo model.py.

4.127.1.30. y_train

```
model.y_train
```

Definición en la línea 44 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 )
```

```
Función para convertir una lista en una tupla

Args:
    value: Lista o str a convertir en tupla

Returns:
    Tupla con el valor de entrada
```

Definición en la línea 15 del archivo util.py.

```
00015 def list_to_tuple(value):
00016     """
00017 Función para convertir una lista en una tupla
00018
00019 Args:
00020 value: Lista o str a convertir en tupla
00021
00022 Returns:
00023 Tupla con el valor de entrada
00024 """
00025 if isinstance(value, str):
00026         return (value,)
00027     elif isinstance(value, list):
00028         return tuple(value)
00029     else:
00030         return ValueError(
00031             "Tipo de dato no soportado, se esperaba un list o str. {}".format(
00032                 type(value)
00033             )
00034         )
00035
00036
```

Gráfico de llamadas a esta función:

4.128.1.2. load_data_for_model()

```
def util.load_data_for_model ( )
```

```
Función para cargar los datos del modelo

Returns:
    pd.DataFrame: Datos del modelo
```

Definición en la línea 66 del archivo util.py.

```
00066 def load_data_for_model():
00067     """
00068 Función para cargar los datos del modelo
00069
00070 Returns:
00071 pd.DataFrame: Datos del modelo
00072 """
00073 data = data_for_model()
00074 df = pd.DataFrame(data)
00075
00076 # Ajustar los tipos de datos de las columnas
00077     df = df.astype({
00078         'id': str,
00079         'anio_nac': int,
00080         'nacionalidad': str,
00081         'sexo': str,
00082         'titulacion': str,
00083         'nota_def_ebau': float,
00084         'nota_media': float,
00085         'abandona': str
00086     })
00087
00088     return df
00089
```

Gráfico de llamadas para esta función:

Gráfico de llamadas a esta función:

4.128.1.3. load_model()

```
def util.load_model ( )
```

Definición en la línea 90 del archivo util.py.

```
00090 def load_model():
00091
00092     model_file = "src/trained_model.pkl"
00093     voting_clf = joblib.load(model_file)
```

```
    data = load_data_for_model()

    current_year = datetime.now().year
    data['edad_actual'] = current_year - data['anio_nac']

    # Codificar variable objetivo
    data['abandona'] = data['abandona'].map({'si': 1, 'no': 0})

    # Definir columnas
    columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
    columnas_numericas = ['nota_def_ebau', 'nota_media', 'edad_actual']

    # Imputación de datos
    num_imputer = SimpleImputer(strategy='mean')
    cat_imputer = SimpleImputer(strategy='constant', fill_value='Desconocido')

    # Preprocesamiento
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', Pipeline([('imputer', num_imputer), ('scaler', StandardScaler())]), columnas_numericas),
            ('cat', Pipeline([('imputer', cat_imputer), ('encoder', OneHotEncoder(handle_unknown='ignore'))]), columnas_categoricas)
        ]
    )

    # Dividir datos en características y variable objetivo
    X = data[columnas_categoricas + columnas_numericas]
    y = data['abandona']
    ids = data['id']

    # Filtrar los estudiantes que no han abandonado aún
    X_non_abandon = X[y == 0]
    data_non_abandon = data[y == 0].copy()  # Mantener una copia de los datos originales

    # Aplicar preprocesamiento
    X_non_abandon_preprocessed = preprocessor.fit_transform(X_non_abandon)

    # Feature engineering: polynomial features
    poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
    X_non_abandon_poly = poly.fit_transform(X_non_abandon_preprocessed)

    # Predecir probabilidades de abandono
    probabilidades_abandono = voting_clf.predict_proba(X_non_abandon_poly)[:, 1]

    # Añadir las probabilidades al DataFrame original de los estudiantes que no han abandonado
    data_non_abandon['probabilidad_abandono'] = probabilidades_abandono

    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 )
```

```
Función para generar colores aleatorios en formato hexadecimal

Args:
    size: Cantidad de colores a generar

Returns:
    Lista de colores en formato hexadecimal
```

Definición en la línea 37 del archivo util.py.

```
00037 def random_color(size):
00038     """
00039 Función para generar colores aleatorios en formato hexadecimal
00040
00041 Args:
00042 size: Cantidad de colores a generar
00043
00044 Returns:
00045 Lista de colores en formato hexadecimal
00046 """
00047 return ["#%06X"% random.randint(0, 0xFFFFFF) for _ in range(size)]
00048
00049
00050 # 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 51 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

:param dbname: Nombre de la base de datos
:param user: Usuario de la base de datos
:param password: Contraseña del usuario
:param host: Host de la base de datos
:param port: Puerto de la base de datos
:param pool_size: Tamaño del pool de conexiones
:param max_overflow: Número máximo de conexiones que pueden ser creadas
```

Definición en la línea 6 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 18 del archivo db_connector.py.

```
00020     ):
00021         self.db_url = f"postgresql+psycopg2://{user}:{password}@{host}:{port}/{dbname}"
00022         self.engine = create_engine(
00023             self.db_url,
00024             poolclass=QueuePool,
00025             pool_size=pool_size,
00026             max_overflow=max_overflow,
00027         )
00028
```

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 41 del archivo db_connector.py.

```
00041     def close(self):
00042         """
00043 Cierra la conexión a la base de datos
00044 """
00045 self.engine.dispose()
00046
00047
```

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

:param query: Consulta a ejecutar
:param params: Parámetros de la consulta
:return: Resultado de la consulta
```

Definición en la línea 29 del archivo db_connector.py.

```
00029     def execute_query(self, query, params=None):
00030         """
00031 Ejecuta una consulta en la base de datos
00032
00033 :param query: Consulta a ejecutar
00034 :param params: Parámetros de la consulta
00035 :return: Resultado de la consulta
00036         """
00037 with self.engine.connect() as connection:
00038             result = connection.execute(text(query), params or {})
00039             return result.fetchall()
00040
```

5.1.4. Documentación de los datos miembro

5.1.4.1. db_url

`data.db_connector.DatabaseConnector.db_url`

Definición en la línea 21 del archivo db_connector.py.

5.1.4.2. engine

`data.db_connector.DatabaseConnector.engine`

Definición en la línea 22 del archivo db_connector.py.

La documentación para esta clase fue generada a partir del siguiente fichero:

- C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import dash, html
import components.common.header as header
import components.common.footer as footer
from layouts.alumno_layout import alumno_layout
from layouts.docente_layout import docente_layout
from layouts.gestor_layout import gestor_layout
from callbacks.common.callback_update_layout import update_layout
from data.db_connector import db
import atexit
import dash_bootstrap_components as dbc


app = dash.Dash(
    __name__,
    external_stylesheets=[
        'src/assets/css/styles.css',
        dbc.themes.BOOTSTRAP
    ],
    suppress_callback_exceptions=True
)

app.title = 'Dashboard académico'
app._favicon = 'src/assets/images/favicon.ico'
```

```
server = app.server

app.layout = html.Div([
    header.header('store-role'),
    html.Div(id='page-content'),
    footer.footer(),
])

#Cierre de la conexión a la base de datos
atexit.register(db.close)

# Iniciar el servidor
if __name__ == '__main__':
    app.run_server(debug=True)
```

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.

```
from dash import Input, Output, State, callback, callback_context
from data.queries import asignaturas_matriculadas
from util import list_to_tuple

@callback(
    Output('asignaturas-matriculadas', 'options'),
    Output('asignaturas-matriculadas', 'value'),
    Input('selected-alumnado-store', 'data'),
    Input('curso-academico', 'value'),
    Input('titulacion-alumnado', 'value'),
    Input('select-all-button', 'n_clicks'),
    State('asignaturas-matriculadas', 'options')
)
def 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
    """

ctx = callback_context
trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None

    # Evento de selección de todas las asignaturas matriculadas
    if trigger_id == 'select-all-button' and n_clicks > 0:
        return existing_options, [option['value'] for option in existing_options]

    if not (alumno_id and curso_academico and titulacion):
        return [], None

    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        return [], None

    data = asignaturas_matriculadas(alumno_id, curso_academico, titulacion)

    if not data:
        return [], None

    opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]
    value = [option['value'] for option in opciones_dropdown]

    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.

```
from dash import Input, Output,State, callback, callback_context
from callbacks.alumnado.utils.callback_select_alumnado import store_selected_alumnado
from data.queries import curso_academico_alumnado

@callback(
    Output('curso-academico', 'options'),
    Output('curso-academico', 'value'),
    Input('selected-alumnado-store', 'data'),
    Input('titulacion-alumnado', 'value'),
    Input('select-all-cursos-academicos', 'n_clicks'),
    State('curso-academico', 'options')
)
def 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
    """

ctx = callback_context
trigger_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else None

    # Evento de selección de todos los cursos académicos
    if trigger_id == 'select-all-cursos-academicos' and n_clicks > 0:
        return existing_options, [option['value'] for option in existing_options]

    if not (alumno_id and titulacion):
        return [], None

    # Obtener los cursos académicos desde la base de datos
    data = curso_academico_alumnado(alumno_id, titulacion)

    if not data:
        return [], None

    opciones_dropdown = [{'label': curso[0], 'value': curso[0]} for curso in data]
    value = [option['value'] for option in opciones_dropdown]

    return opciones_dropdown, value

```

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.

```
from dash import Input, Output, State, callback
from data.queries import titulacion_alumnado

@callback(
    Output('titulacion-alumnado', 'options'),
    Output('titulacion-alumnado', 'value'),
    Output('selected-titulacion-alumnado-store', 'data'),
    Input('selected-alumnado-store', 'data'),
    Input('titulacion-alumnado', 'value'),
    State('selected-titulacion-alumnado-store', 'data')
)
def 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

    """
if not alumno_id:
        return [], None, None

    data = titulacion_alumnado(alumno_id)

    if not data:
        return [], None, None

    opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]

    # Si no hay valor seleccionado pero hay un valor almacenado, lo usamos si está en las opciones
    if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
        return opciones_dropdown, stored_titulacion, stored_titulacion

    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 from dash import callback, Input, Output
00002 import plotly.graph_objs as go
00003 import pandas as pd
00004 from data.queries import asignaturas_superadas_media_abandono, universidad_alumno
00005 from util import list_to_tuple
00006
00007
00008 @callback(
00009     Output("asignaturas-superadas-general-mi-nota", "figure"),
00010     Input("curso-academico", "value"),
00011     Input("selected-alumnado-store", "data"),
00012     Input("asignaturas-matriculadas", "value"),
00013     Input("titulacion-alumnado", "value"),
00014 )
00015 def update_graph_alumnado(curso_academico, alumno_id, asignaturas_matriculadas, titulacion):
00016     """
00017 Actualiza el gráfico de relación entre la nota media y el número de asignaturas superadas
00018 por alumno del perfil "Alumno" de la pestaña "Rendimiento académico general".
00019
00020     Args:
00021     curso_academico (list): Lista con los cursos académicos
00022     alumno_id (str): Identificador del alumno
00023     asignaturas_matriculadas (list): Lista con las asignaturas matriculadas
00024     titulacion (str): Titulación seleccionada
00025
00026     Returns:
00027     go.Figure: Figura con el gráfico
00028
00029     """
00030 fig = go.Figure()
00031
00032 fig.update_layout(
00033 title={
00034 "text": "Relación nota media y número de asignaturas superadas por alumno",
00035             "x": 0.5,
00036         },
00037         xaxis_title="Nota Media",
00038         yaxis_title="N° Asignaturas superadas",
00039         legend_title="Estado del alumno",
00040         showlegend=True,
00041         xaxis=dict(range=[0, 10]),
00042         yaxis=dict(range=[0, 40]),
00043     )
00044
00045     if not (curso_academico and alumno_id and asignaturas_matriculadas and titulacion):
00046         return fig
00047
00048     try:
00049         curso_academico = list_to_tuple(curso_academico)
00050         asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
00051     except Exception as e:
00052         print("Error:", e)
00053         return fig
00054
00055     data_universidad = universidad_alumno(alumno_id)
00056
00057     if not data_universidad:
00058         return fig
00059
00060     data = asignaturas_superadas_media_abandono(
00061         curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
00062     )
00063
00064     if not data:
00065         return fig
00066
00067     df = pd.DataFrame(
00068         data, columns=["Alumno_id", "Abandono", "Nota_Media", "Asignaturas_Superadas"]
00069     )
00070
00071     df["Abandono"] = (
00072         df["Abandono"]
```

```
        .str.strip()
        .str.lower()
        .replace({"si": "Abandona", "no": "No abandona"})
    )
    df["Personal"] = df["Alumno_id"].apply(lambda x: " (Yo)" if x == alumno_id else "")
    df["Key"] = df["Abandono"] + df["Personal"]

    colors = {
        "Abandona": "red",
        "No abandona": "blue",
        "Abandona (Yo)": "yellow",
        "No abandona (Yo)": "yellow",
    }

    for key, group in df.groupby("Key"):
        fig.add_trace(
            go.Scatter(
                x=group["Nota_Media"],
                y=group["Asignaturas_Superadas"],
                mode="markers",
                name=key,
                marker=dict(
                    size=12,
                    line=dict(width=2 if " (Yo)" in key else 1),
                    color=colors[key],
                ),
                opacity=1.0 if " (Yo)" in key else 0.8,
            )
        )

    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.

```
from dash import callback, Input, Output
import plotly.graph_objs as go
import pandas as pd
from data.queries import (
    calif_cualitativa_comparativa,
    calif_cualitativa_alumno_asignaturas,
    universidad_alumno,
)
from util import list_to_tuple

```

```python

@callback(
    Output("nota-cualitativa-general-mi-nota", "figure"),
    Input("curso-academico", "value"),
    Input("asignaturas-matriculadas", "value"),
    Input("selected-alumnado-store", "data"),
    Input("titulacion-alumnado", "value"),
)
def 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
    """
fig = go.Figure()

fig.update_layout(
title={
"text": "Calificaciones cualitativas general por curso académico",
            "x": 0.5,
        },
        barmode="stack",
        xaxis={"title": "Asignatura", "tickangle": 45},
        yaxis={"title": "N° Alumnos matriculados"},
        showlegend=True,
        legend={"title": "Calificación"},
        height=600,
    )

    if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
        asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
    except Exception as e:
        print("Error:", e)
        return fig

    data_universidad = universidad_alumno(alumno_id)

    if not data_universidad:
        return fig

    general_data = calif_cualitativa_comparativa(
        curso_academico, asignaturas_matriculadas, titulacion, data_universidad[0][0]
    )
    student_data = calif_cualitativa_alumno_asignaturas(
        alumno_id, curso_academico, asignaturas_matriculadas, titulacion
    )

    if not general_data:
        return fig

    general_df = pd.DataFrame(
        general_data, columns=["Asignatura", "Calificacion", "Numero"]
    )
    student_df = pd.DataFrame(
        student_data, columns=["Asignatura", "Calificacion", "Numero"]
    )

    categories = ["Suspenso", "No presentado", "Aprobado", "Notable", "Sobresaliente"]

    color_mapping = {
        "Sobresaliente": "blue",
        "Notable": "green",
        "Aprobado": "orange",
        "Suspenso": "red",
        "No presentado": "gray",
    }

    general_pivot = general_df.pivot_table(
        index="Asignatura", columns="Calificacion", values="Numero", fill_value=0
    )
    general_pivot = general_pivot.reindex(columns=categories, fill_value=0)

    for category in categories:
        fig.add_trace(
            go.Bar(
```

```
                x=general_pivot.index,
                y=general_pivot[category],
                name=category,
                marker=dict(color=color_mapping[category]),
                opacity=0.7,
            )
        )

    for category in categories:
        y = [
            1 if (
                (subject in student_df['Asignatura'].values) and
                (student_df[student_df['Asignatura'] == subject]['Calificacion'].values[0] == category)
            ) else 0 for subject in general_pivot.index
        ]
        fig.add_trace(
            go.Bar(
                x=general_pivot.index,
                y=y,
                name=f"{category} (Yo)",
                marker=dict(color=color_mapping[category], line=dict(color='black', width=2)),
                opacity=1
            )
        )

    return fig
```

6.87. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import callback, Input, Output
import pandas as pd
import plotly.graph_objs as go
from data.queries import nota_media_general_mi_nota, universidad_alumno
from util import list_to_tuple


@callback(
    Output("nota-media-general-mi-nota", "figure"),
    Input("curso-academico", "value"),
    Input("asignaturas-matriculadas", "value"),
    Input("selected-alumnado-store", "data"),
    Input("titulacion-alumnado", "value"),
)
```

```
def 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
    """
fig = go.Figure()

fig.update_layout(
title={
"text": "Relación calificaciones del alumno con nota media general",
            "x": 0.5,
        },
        xaxis={"title": "Asignatura", "tickangle": 45},
        yaxis={"title": "Nota"},
        barmode="group",
        height=600,
    )

    if not (curso_academico and asignaturas_matriculadas and alumno_id and titulacion):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
        asignaturas_matriculadas = list_to_tuple(asignaturas_matriculadas)
    except Exception as e:
        print("Error:", e)
        return fig

    data_universidad = universidad_alumno(alumno_id)

    if not data_universidad:
        return fig

    data = nota_media_general_mi_nota(
        curso_academico,
        asignaturas_matriculadas,
        alumno_id,
        titulacion,
        data_universidad[0][0],
    )

    if not data:
        return fig

    df = pd.DataFrame(data, columns=["Asignatura", "NotaMediaGeneral", "MiNota"])

    fig.add_trace(
        go.Bar(
            x=df["Asignatura"],
            y=df["MiNota"],
            name="Mi nota",
            marker_color="blue",
            opacity=0.7,
        )
    )

    fig.add_trace(
        go.Bar(
            x=df["Asignatura"],
            y=df["NotaMediaGeneral"],
            name="Nota media general",
            marker_color="grey",
            opacity=0.7,
        )
    )

    return fig
```

6.89. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, callback
00002 import plotly.graph_objs as go
00003 import pandas as pd
00004 from data.queries import calif_cualitativa_asignatura
00005 from util import list_to_tuple
00006
00007
00008 @callback(
00009     Output("graph-bar-evolucion-asignaturas-matriculadas", "figure"),
00010     Input("selected-alumnado-store", "data"),
00011     Input("curso-academico", "value"),
00012     Input("titulacion-alumnado", "value"),
00013 )
00014 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00015     """
00016 Actualiza el gráfico de calificaciones cualitativas de las asignaturas matriculadas
00017 del perfil "Alumno" de la pestaña "Expediente académico personal".
00018
00019     Args:
00020     alumno_id (str): Identificador del alumno.
00021     curso_academico (list): Lista con los cursos académicos
00022     titulacion (str): Titulación seleccionada
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Calificaciones cualitativas", "x": 0.5},
00032         barmode="stack",
00033         xaxis={"title": "Curso académico"},
00034         yaxis={"title": "N° Asignaturas matriculadas"},
00035         showlegend=True,
00036         legend={"title": "Calificación"},
00037     )
00038
00039     if not (alumno_id and curso_academico and titulacion):
00040         return fig
00041
00042     try:
00043         curso_academico = list_to_tuple(curso_academico)
00044     except Exception as e:
00045         print("Error:", e)
00046         return fig
```

```
    data = calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion)

    if not data:
        return fig

    df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "cantidad"])
    df_pivot = df.pivot_table(
        index="curso_academico", columns="calificacion", values="cantidad", fill_value=0
    )

    categories = ["No presentado", "Suspenso", "Aprobado", "Notable", "Sobresaliente"]
    for category in categories:
        if category not in df_pivot.columns:
            df_pivot[category] = 0

    df_pivot = df_pivot[categories]
    df_pivot = df_pivot.sort_index()

    color_mapping = {
        "Sobresaliente": "blue",
        "Notable": "green",
        "Aprobado": "orange",
        "Suspenso": "red",
        "No presentado": "gray",
    }

    for category in categories:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot[category],
                name=category,
                marker_color=color_mapping[category],
                opacity=0.7,
            )
        )

    return fig
```

6.91. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import Input, Output, callback
import plotly.graph_objs as go
```

```python
import pandas as pd
from data.queries import calif_numerica_asignatura
from util import list_to_tuple, random_color


@callback(
    Output("graph-bar-calificaciones-por-asignatura", "figure"),
    Input("selected-alumnado-store", "data"),
    Input("curso-academico", "value"),
    Input("titulacion-alumnado", "value"),
)
def 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
    """

fig = go.Figure()

fig.update_layout(
title={"text": "Calificaciones cuantitativas", "x": 0.5},
        xaxis={"title": "Asignatura", "tickangle": 45},
        yaxis={"title": "Calificación"},
        height=600,
        showlegend=True,
        legend=dict(
            x=1,
            y=1,
            traceorder="normal",
            font=dict(
                size=10,
            ),
            title="Asignaturas",
        ),
    )

    if not (alumno_id and curso_academico and titulacion):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        print("Error:", e)
        return fig

    data = calif_numerica_asignatura(alumno_id, curso_academico, titulacion)

    if not data:
        return fig

    df = pd.DataFrame(data, columns=["Asignatura", "Calificacion"])
    colors = random_color(len(df))

    for index, row in df.iterrows():
        fig.add_trace(
            go.Bar(
                x=[row["Asignatura"]],
                y=[row["Calificacion"]],
                name=row["Asignatura"],
                marker_color=colors[index],
                opacity=0.7,
            )
        )

    return fig
```

6.93. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, callback
00002 import plotly.graph_objs as go
00003 import pandas as pd
00004 from data.queries import asignaturas_superadas
00005 from util import list_to_tuple
00006
00007
00008 @callback(
00009     Output("graph-evolucion-progreso-academico", "figure"),
00010     Input("selected-alumnado-store", "data"),
00011     Input("curso-academico", "value"),
00012     Input("titulacion-alumnado", "value"),
00013 )
00014 def update_graph_alumnado(alumno_id, curso_academico, titulacion):
00015     """
00016 Actualiza el gráfico de evolución del progreso académico del perfil "Alumno"
00017     de la pestaña "Expediente académico personal".
00018
00019     Args:
00020     alumno_id (str): Identificador del alumno.
00021     curso_academico (list): Lista con los cursos académicos
00022     titulacion (str): Titulación seleccionada
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Evolución del progreso académico", "x": 0.5},
00032         xaxis={"title": "Curso académico"},
00033         yaxis={"title": "N° Asignaturas de superadas"},
00034         showlegend=False,
00035     )
00036
00037     if not (alumno_id and curso_academico and titulacion):
00038         return fig
00039
00040     try:
00041         curso_academico = list_to_tuple(curso_academico)
00042     except Exception as e:
00043         print("Error:", e)
00044         return fig
00045
00046     data = asignaturas_superadas(alumno_id, curso_academico, titulacion)
```

```

    if not data:
        return fig

    data = pd.DataFrame(data)
    academic_years = data["curso_academico"]
    subjects_passed = data["n_asig_superadas"]

    cumulative_passed = []
    cumulative_total = 0
    for count in subjects_passed:
        cumulative_total += count
        cumulative_passed.append(cumulative_total)

    fig.add_trace(
        go.Bar(x=academic_years, y=cumulative_passed, marker_color="blue", opacity=0.7)
    )

    return fig
```

6.95. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import Input, Output, callback
import plotly.graph_objs as go
import pandas as pd
from data.queries import asignaturas_matriculadas_y_superadas
from util import list_to_tuple


@callback(
    Output("graph-bar-tasa-rendimiento", "figure"),
    Input("selected-alumnado-store", "data"),
    Input("curso-academico", "value"),
    Input("titulacion-alumnado", "value"),
)
def 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

    """

fig = go.Figure()

fig.update_layout(
title={"text": "Tasa de rendimiento por curso académico", "x": 0.5},
        xaxis={"title": "Tasa de rendimiento (%)"},
        yaxis={"title": "Curso académico"},
        showlegend=False,
    )

    if not (alumno_id and curso_academico and titulacion):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        print("Error:", e)
        return fig

    data = asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion)

    if not data:
        return fig

    df = pd.DataFrame(data, columns=["curso_academico", "matriculadas", "superadas"])
    df["tasa_rendimiento"] = (df["superadas"] / df["matriculadas"]) * 100

    fig.add_trace(
        go.Bar(
            x=df["tasa_rendimiento"],
            y=df["curso_academico"],
            orientation="h",
            marker_color="blue",
            opacity=0.7,
            width=0.7,
        )
    )

    return fig
```

6.97. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import html, callback, Output, Input
import pandas as pd
from data.queries import nota_media_alumno_titulacion, resumen_alumno
from util import load_model

@callback(
    Output("resumen-alumnado", "children"),
    Input("selected-alumnado-store", "data"),
    Input("titulacion-alumnado", "value"),
)
def 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"
    """
if not (alumno_id and titulacion):
        return not_data()

    data = resumen_alumno(alumno_id, titulacion)

    data = pd.DataFrame(data, columns=["id_alumno", "universidad", "titulacion", "abandona"])

    if data.empty:
        return not_data()

    componentes = [
        html.H2("Resumen"),
        html.P("Universidad:", className="resumen-label"),
        html.P(data["universidad"].iloc[0]),
        html.P("Titulación:", className="resumen-label"),
        html.P(data["titulacion"].iloc[0]),
        html.P("Alumno:", className="resumen-label"),
        html.P(alumno_id),
        html.P("Nota Media:", className="resumen-label"),
        html.P(nota_media_alumno_titulacion(alumno_id, titulacion))
    ]

    if data["abandona"].iloc[0] == 'no':
        componentes.append(html.P("Estado:", className="resumen-label"))
        componentes.append(html.P("Activo"))
        componentes.append(html.P("Probabilidad de abandono:", className="resumen-label"))
        data_model = load_model()

        probabilidad_abandono = data_model[data_model['id'] == alumno_id]['probabilidad_abandono'].values[0]
        componentes.append(html.P(f"{probabilidad_abandono:.2%}"))
    else:
        componentes.append(html.P("Estado:", className="resumen-label"))
        componentes.append(html.P("Abandona"))


    componentes.append(html.Hr())

    return html.Div(componentes)



def not_data():
    return html.Div(
        [
            html.H2("Resumen"),
            html.P("Universidad:", className="resumen-label"),
            html.P("No disponible"),
            html.P("Titulación:", className="resumen-label"),
            html.P("No disponible"),
            html.P("Alumno:", className="resumen-label"),
            html.P("No disponible"),
            html.P("Nota Media:", className="resumen-label"),
            html.P("No disponible"),
            html.Hr(),
        ]
    )
```

6.99. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, State, callback
00002 from data.queries import alumnos_all
00003
00004
00005 @callback(
00006     Output("alumnado-dropdown", "value"),
00007     Output("alumnado-dropdown", "options"),
00008     Output("selected-alumnado-store", "data"),
00009     Input("alumnado-dropdown", "value"),
00010     State("selected-alumnado-store", "data"),
00011 )
00012 def store_selected_alumnado(selected_value, stored_value):
00013     """
00014 Almacena el valor seleccionado en el dropdown de alumnos.
00015
00016 Args:
00017 selected_value (str): Valor seleccionado en el dropdown
00018 stored_value (str): Valor almacenado
00019
00020 Returns:
00021 str: Valor seleccionado
00022 list: Opciones del dropdown
00023 str: Valor almacenado
00024 """
00025 data = alumnos_all()
00026
00027 if not data:
00028         return None, [], None
00029
00030     opciones_dropdown = [{"label": alumno[0], "value": alumno[0]} for alumno in data]
00031
00032     if selected_value is None and stored_value in [
00033         op["value"] for op in opciones_dropdown
00034     ]:
00035         return stored_value, opciones_dropdown, stored_value
00036
00037     return selected_value, opciones_dropdown, selected_value
```

6.101. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import html, callback, Output, Input, State
00002 from components.common.sidebar import sidebar
00003 from components.common.filters import filters
00004 from components.alumnado.utils.select_alumnado import select_alumnado
00005 from components.alumnado.utils.recomendador_alumnado import recomendador_alumnado
00006 from components.alumnado.utils.resumen_alumnado import resumen_alumnado
00007 from components.alumnado.filters.filter_curso_academico_alumnado import
      filter_curso_academico_alumnado
00008 from components.alumnado.filters.filter_asignaturas_matri_alumnado import
      filter_asignaturas_matri_alumnado
00009 from components.alumnado.filters.filter_titulacion_alumnado import filter_titulacion_alumnado
00010 from components.alumnado.graphs.graphs_general_alumnado import graphs_general_alumnado
00011 from components.alumnado.graphs.graphs_personal_alumnado import graphs_personal_alumnado
00012
00013
00014 @callback(
00015     Output('tabs-alumnado-content', 'children'),
00016     Input('tabs-alumnado', 'value'),
00017     State('selected-alumnado-store', 'data')
00018 )
00019 def render_content(tab, selected_alumnado):
00020     """
00021 Renderiza el contenido de las pestañas del dashboard del alumnado.
00022
00023 Args:
00024 tab (str): Pestaña seleccionada
00025 selected_alumnado (str): Alumno seleccionado
00026
00027 Returns:
00028 list: Componentes de la pestaña seleccionada
00029 """
00030 if tab == 'expediente-personal-tab':
00031         return html.Div([
00032             select_alumnado(),
00033             html.H2("Dashboard Alumnado", style={'textAlign': 'center'}),
00034             html.Div([
00035                 sidebar([
00036                     resumen_alumnado(),
00037                     filters([
00038                         filter_titulacion_alumnado(),
00039                         filter_curso_academico_alumnado()
00040                         ])
00041                     ]),
00042                 graphs_personal_alumnado()
00043             ], className='content-layout-dashboard')
00044         ])
00045     elif tab == 'rendimiento-academico-tab':
00046         return html.Div([
00047             html.H2("Dashboard Alumnado", style={'textAlign': 'center'}),
00048             html.Div([
00049                 sidebar([
00050                     resumen_alumnado(),
00051                     filters([
00052                         filter_titulacion_alumnado(),
00053                         filter_curso_academico_alumnado(),
00054                         filter_asignaturas_matri_alumnado()
00055                         ])
00056                     ]),
00057                 graphs_general_alumnado()
00058             ], className='content-layout-dashboard')
00059         ])
00060     elif tab == 'recomendador-tab':
00061         return html.Div([
00062             recomendador_alumnado()
00063         ])
00064
```

6.103. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ Universidad/6_curso/BI_Universities_↩ TFG/src/callbacks/common/callback_sidebar_collapse.py

Namespaces

- namespace callbacks
- namespace callbacks.common
- namespace callbacks.common.callback_sidebar_collapse

Funciones

- def callbacks.common.callback_sidebar_collapse.sidebar_collapse (n, is_open)

6.104. callback_sidebar_collapse.py

Ir a la documentación de este archivo.

```
00001 from dash import Output, Input, State, callback
00002
00003
00004 @callback(
00005     Output("collapse", "is_open"),
00006     Input("sidebar-toggle", "n_clicks"),
00007     State("collapse", "is_open"),
00008 )
00009 def sidebar_collapse(n, is_open):
00010     """
00011 Callback que permite colapsar y descolapsar el sidebar.
00012
00013 Args:
00014 n (int): Número de clicks
00015 is_open (bool): Estado del sidebar
00016
00017 Returns:
00018 bool: Estado del sidebar
00019 """
00020 if n:
00021         return not is_open
00022     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.

```
from dash import Output, Input, callback
from layouts.alumno_layout import alumno_layout
from layouts.docente_layout import docente_layout
from layouts.gestor_layout import gestor_layout

@callback(
    Output('page-content', 'children'),
    Input('store-role', 'data')
)
def 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
"""
if role == 'Alumno':
        return alumno_layout()
    elif role == 'Docente':
        return docente_layout()
    elif role == 'Gestor':
        return gestor_layout()
    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.

```
from dash import Output, Input, State, callback


@callback(
    Output("dropdown_role", "value"),
    Input("store-role", "modified_timestamp"),
    State("store-role", "data"),
    prevent_initial_call=True,
)
def 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
"""
return stored_role if ts is not None else "Alumno"


@callback(
    Output("store-role", "data"),
    Input("dropdown_role", "value"),
    State("store-role", "data"),
    prevent_initial_call=True,
)
def 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
"""
if new_role != current_role:
        return new_role
    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 from dash import Input, Output, State, callback, callback_context
00002 from data.queries import asignaturas_actas_titulacion
00003
00004
00005 @callback(
00006     Output("all-asignaturas-titulacion-docente", "options"),
00007     Output("all-asignaturas-titulacion-docente", "value"),
00008     Input("titulacion-docente", "value"),
00009     Input("all-cursos-academicos-docente", "value"),
00010     Input("select-all-asignaturas-titulacion-docente", "n_clicks"),
00011     State("all-asignaturas-titulacion-docente", "options"),
00012 )
00013 def update_filter_asignaturas_docente(titulacion, curso_academico, n_clicks, existing_options):
00014     """
00015 Actualiza las opciones del dropdown con todas las asignaturas de la titulación seleccionada y
00016 gestiona el evento del botón que selecciona todas las asignaturas.
00017
00018 Args:
00019 titulacion (str): Titulación seleccionada
00020 curso_academico (list): Cursos académicos seleccionados
00021 n_clicks (int): Número de clicks en el botón "Seleccionar todo"
00022     existing_options (list): Opciones actuales del dropdown
00023
00024     Returns:
00025     list: Opciones del dropdown
00026     list: Valor seleccionado
00027     """
00028 ctx = callback_context
00029 trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None
00030
00031     if trigger_id == "select-all-asignaturas-titulacion-docente" and n_clicks > 0:
00032         return existing_options, [option["value"] for option in existing_options]
00033
00034     if not (titulacion and curso_academico):
00035         return [], None
00036
00037     data = asignaturas_actas_titulacion(titulacion, curso_academico)
00038
00039     if not data:
00040         return [], None
00041
00042     opciones_dropdown = [
00043         {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
00044     ]
00045     value = (
00046         [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
00047     )
00048
00049     return opciones_dropdown, value
```

6.111. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Input, Output, callback
from data.queries import curso_academico_actas_titulacion


@callback(
    Output("all-cursos-academicos-docente", "options"),
    Output("all-cursos-academicos-docente", "value"),
    Input("titulacion-docente", "value"),
)
def 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
"""

if not titulacion:
        return [], None

    data = curso_academico_actas_titulacion(titulacion)

    if not data:
        return [], None

    opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
    value = opciones_dropdown[0]["value"] if opciones_dropdown else None

    return opciones_dropdown, value
```

6.113. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Input, Output, callback
from data.queries import asignaturas_docente


@callback(
```

```
    Output("asignaturas-docente", "options"),
    Output("asignaturas-docente", "value"),
    Input("selected-docente-store", "data"),
    Input("titulacion-docente", "value"),
)
def 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
"""

if not (docente_id and titulacion):
        return [], None

    data = asignaturas_docente(docente_id, titulacion)

    if not data:
        return [], None

    opciones_dropdown = [
        {"label": asignatura[0], "value": asignatura[0]} for asignatura in data
    ]
    value = opciones_dropdown[0]["value"] if opciones_dropdown else None

    return opciones_dropdown, value
```

6.115. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Input, Output, callback, State, callback_context
from data.queries import curso_academico_docente
from callbacks.docente.utils.callback_select_docente import store_selected_docente


@callback(
    Output("curso-academico-docente", "options"),
    Output("curso-academico-docente", "value"),
    Input("selected-docente-store", "data"),
    Input("asignaturas-docente", "value"),
    Input("select-all-cursos-academicos-docente", "n_clicks"),
```

```
    State("curso-academico-docente", "options"),
)
def 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

    """
ctx = callback_context
trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None

    if trigger_id == "select-all-cursos-academicos" and n_clicks > 0:
        return existing_options, [option["value"] for option in existing_options]

    if not (docente_id and asignatura):
        return [], None

    data = curso_academico_docente(docente_id, asignatura)

    if not data:
        return [], None

    opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
    value = [option["value"] for option in opciones_dropdown]

    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.

```
from dash import Input, Output, State, callback
from data.queries import titulacion_docente

@callback(
  Output('titulacion-docente', 'options'),
  Output('titulacion-docente', 'value'),
  Output('selected-titulacion-docente-store', 'data'),
```

```
  Input('selected-docente-store', 'data'),
  Input('titulacion-docente', 'value'),
  State('selected-titulacion-docente-store', 'data')
)
def 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

"""
if not docente_id:
        return [], None, None

    data = titulacion_docente(docente_id)

    if not data:
        return [], None, None

    opciones_dropdown = [{'label': asignatura[0], 'value': asignatura[0]} for asignatura in data]

    if selected_value is None and stored_titulacion in [op['value'] for op in opciones_dropdown]:
        return opciones_dropdown, stored_titulacion, stored_titulacion

    return opciones_dropdown, selected_value, selected_value


```

6.119. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Output, Input, callback
import plotly.graph_objs as go
import pandas as pd
from data.queries import calif_all_cualitativa_asignaturas
from util import list_to_tuple


```

```
@callback(
    Output("calificaiones-cuali-all-asig-docente", "figure"),
    Input("titulacion-docente", "value"),
    Input("all-cursos-academicos-docente", "value"),
    Input("all-asignaturas-titulacion-docente", "value"),
)
def 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
    """

fig = go.Figure()

fig.update_layout(
barmode="stack",
        title={"text": "Calificaciones cualitativas de la titulación", "x": 0.5},
        xaxis={"title": "Asignaturas", "tickangle": 45},
        yaxis={"title": "N° Alumnos matriculados"},
        legend=dict(
            x=1,
            y=1,
            traceorder="normal",
            font=dict(
                size=10
            ),
            title="Calificaciones"
        ),
    )

    if not (curso_academico and asignatura):
        return fig

    try:
        asignatura = list_to_tuple(asignatura)
    except Exception as e:
        return fig

    data = calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignatura)

    if not data:
        return fig

    df = pd.DataFrame(
        data, columns=["titulacion", "asignatura", "calificación", "n_alumnos"]
    )

    colors_mapping = {
        "Sobresaliente": "blue",
        "Notable": "green",
        "Aprobado": "orange",
        "Suspenso": "red",
        "No presentado": "gray",
    }

    for calif, color in colors_mapping.items():
        df_calif = df[df["calificación"] == calif]
        fig.add_trace(
            go.Bar(
                x=df_calif["asignatura"],
                y=df_calif["n_alumnos"],
                name=calif,
                marker_color=color,
                opacity=0.7,
            )
        )

    fig.update_xaxes(categoryorder="array", categoryarray=asignatura)

    return fig
```

6.121. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 from dash import callback, Output, Input
00002 import plotly.graph_objs as go
00003 import pandas as pd
00004 from data.queries import calif_media_asignaturas
00005 from util import list_to_tuple
00006
00007
00008 @callback(
00009     Output("calificaiones-media-all-asig-docente", "figure"),
00010     Input("titulacion-docente", "value"),
00011     Input("all-cursos-academicos-docente", "value"),
00012     Input("all-asignaturas-titulacion-docente", "value"),
00013 )
00014 def update_graph_docente(titulacion, curso_academico, asignatura):
00015     """
00016 Actualiza el gráfico de nota media por asignatura de la titulación
00017 seleccionada del perfil "Docente" de la pestaña "Rendimiento académico general".
00018
00019     Args:
00020     titulacion (str): Titulación seleccionada
00021     curso_academico (list): Lista con los cursos académicos
00022     asignatura (list): Lista con las asignaturas seleccionadas
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 title={"text": "Nota media por asignatura de la titulación", "x": 0.5},
00032         xaxis_title="Asignaturas",
00033         yaxis_title="Nota media",
00034     )
00035
00036     if not (curso_academico and asignatura):
00037         return fig
00038
00039     try:
00040         asignatura = list_to_tuple(asignatura)
00041     except Exception as e:
00042         return fig
00043
00044     data = calif_media_asignaturas(titulacion, curso_academico, asignatura)
00045
00046     if not data:
```

```
        return fig

    df = pd.DataFrame(data, columns=["asignatura", "media_calif"])

    fig.add_trace(
        go.Bar(
            x=df["asignatura"], y=df["media_calif"], marker_color="blue", opacity=0.7
        )
    )

    return fig
```

6.123. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import Input, Output, callback
import plotly.graph_objs as go
import pandas as pd
from data.queries import alumnos_genero_docente
from util import list_to_tuple


@callback(
    Output("graph-alumnos-matri-genero", "figure"),
    Input("asignaturas-docente", "value"),
    Input("curso-academico-docente", "value"),
    Input("selected-docente-store", "data"),
)
def 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
    """

fig = go.Figure()

fig.update_layout(
```

```
barmode="stack",
        title={"text": "Evolución alumnos matriculados por género", "x": 0.5},
        xaxis={"title": "Curso académico"},
        yaxis={"title": "Nº Alumnos matriculados"},
        legend={"title": "Género"},
    )

    if not (asignaturas and curso_academico and docente_id):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        return fig

    data = alumnos_genero_docente(docente_id, asignaturas, curso_academico)

    if not data:
        return fig

    df = pd.DataFrame(data, columns=["curso_academico", "genero", "cantidad"])

    df_pivot = df.pivot(
        index="curso_academico", columns="genero", values="cantidad"
    ).fillna(0)

    colores = {"Femenino": "red", "Masculino": "blue"}

    for genero in df_pivot.columns:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot[genero],
                name="Mujeres" if genero == "Femenino" else "Hombres",
                marker_color=colores[genero],
                opacity=0.7,
            )
        )

    return fig
```

6.125. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import Input, Output, callback
import plotly.graph_objs as go
import pandas as pd
from data.queries import alumnos_nota_media_docente
from util import list_to_tuple


@callback(
    Output("graph-alumnos-nota-media", "figure"),
    Input("asignaturas-docente", "value"),
    Input("curso-academico-docente", "value"),
    Input("selected-docente-store", "data"),
)
def 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
    """

fig = go.Figure()

fig.update_layout(
title={"text": "Evolución nota media por asignatura", "x": 0.5},
        xaxis_title="Curso académico",
        yaxis_title="Nota media",
        xaxis=dict(type="category"),
    )

    if not (asignaturas and curso_academico and docente_id):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
        asignaturas = list_to_tuple(asignaturas)
    except Exception as e:
        return fig

    data = alumnos_nota_media_docente(asignaturas, curso_academico)

    if not data:
        return fig

    df = pd.DataFrame(data, columns=["curso_academico", "asignatura", "nota_media"])

    fig.add_trace(
        go.Bar(
            x=df["curso_academico"],
            y=df["nota_media"],
            name="Nota media",
            marker=dict(color="blue"),
            opacity=0.7,
        )
    )

    return fig
```

6.127. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, callback
00002 import plotly.graph_objs as go
00003 import pandas as pd
00004 from data.queries import alumnos_nota_cualitativa_docente
00005 from util import list_to_tuple
00006
00007
00008 @callback(
00009     Output("graph-alumnos-nota-cualitativa", "figure"),
00010     Input("asignaturas-docente", "value"),
00011     Input("curso-academico-docente", "value"),
00012     Input("selected-docente-store", "data"),
00013 )
00014 def update_graph_docente(asignaturas, curso_academico, docente_id):
00015     """
00016 Actualiza el gráfico de evolución de calificaciones cualitativas
00017 del perfil "Docente" de la pestaña "Rendimiento académico personal".
00018
00019     Args:
00020     asignaturas (list): Lista con las asignaturas seleccionadas
00021     curso_academico (list): Lista con los cursos académicos
00022     docente_id (str): Identificador del docente
00023
00024     Returns:
00025     go.Figure: Figura con el gráfico
00026     """
00027
00028 fig = go.Figure()
00029
00030 fig.update_layout(
00031 barmode="stack",
00032         title={"text": "Evolución calificaciones cualitativas", "x": 0.5},
00033         xaxis={"title": "Curso académico"},
00034         yaxis={"title": "Nº Alumnos matriculados"},
00035         legend_title_text="Calificación",
00036     )
00037
00038     if not (asignaturas and curso_academico and docente_id):
00039         return fig
00040
00041     try:
00042         curso_academico = list_to_tuple(curso_academico)
00043         asignaturas = list_to_tuple(asignaturas)
00044     except Exception as e:
00045         return fig
00046
00047     data = alumnos_nota_cualitativa_docente(asignaturas, curso_academico)
00048     if not data:
00049         return fig
00050
00051     df = pd.DataFrame(data, columns=["curso_academico", "calificacion", "n_alumnos"])
00052
00053     # Pivotear el DataFrame para obtener las cantidades por calificación en columnas separadas
00054     df_pivot = df.pivot(
00055         index="curso_academico", columns="calificacion", values="n_alumnos"
00056     ).fillna(0)
00057     catagories = ["No presentado", "Suspenso", "Aprobado", "Notable", "Sobresaliente"]
00058
00059     for calif in catagories:
00060         if calif not in df_pivot:
00061             df_pivot[calif] = 0
00062
```

```
    df_pivot = df_pivot[catagories]

    colors = {
        "Sobresaliente": "blue",
        "Notable": "green",
        "Aprobado": "orange",
        "Suspenso": "red",
        "No presentado": "gray",
    }

    for calif in catagories:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot[calif],
                name=calif,
                marker=dict(color=colors[calif]),
                opacity=0.7,
            )
        )

    return fig
```

6.129. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import Input, Output, callback
import plotly.graph_objs as go
import pandas as pd
from data.queries import alumnos_repetidores_nuevos
from util import list_to_tuple


@callback(
    Output("graph-alumnos-repetidores-nuevos", "figure"),
    Input("asignaturas-docente", "value"),
    Input("curso-academico-docente", "value"),
    Input("selected-docente-store", "data"),
)
def 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
    """

fig = go.Figure()

fig.update_layout(
barmode="stack",
        title={"text": "Evolución alumnos de nuevo ingreso y repetidores", "x": 0.5},
        xaxis={"title": "Curso académico"},
        yaxis={"title": "N° Alumnos matriculados"},
        showlegend=True,
        legend={"orientation": "h", "x": 0, "y": 1.1},
    )

    if not (asignaturas and curso_academico and docente_id):
        return fig

    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        return fig

    data = alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas)

    if not data:
        return fig

    df = pd.DataFrame(
        data,
        columns=["curso_academico", "alumnos_repetidores", "alumnos_nuevo_ingreso"],
    )

    fig.add_trace(
        go.Bar(
            x=df["curso_academico"],
            y=df["alumnos_repetidores"],
            name="Alumnos repetidores",
            marker_color="red",
            opacity=0.7,
        )
    )

    fig.add_trace(
        go.Bar(
            x=df["curso_academico"],
            y=df["alumnos_nuevo_ingreso"],
            name="Alumnos de primera matrícula",
            marker_color="green",
            opacity=0.7,
        )
    )

    return fig
```

6.131. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import html, callback, Output, Input
from data.queries import universidades_docente


@callback(
    Output("resumen-docente", "children"),
    Input("selected-docente-store", "data"),
    Input("titulacion-docente", "value"),
)
def 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
"""
if not (docente_id and titulacion):
        return not_data()

    data = universidades_docente(docente_id)

    if not data:
        return not_data()

    return html.Div(
        [
            html.H2("Resumen"),
            html.P("Universidad:", className="resumen-label"),
            html.P(data[0][1]),
            html.P("Titulación:", className="resumen-label"),
            html.P(titulacion),
            html.P("Docente:", className="resumen-label"),
            html.P(docente_id),
            html.Hr(),
        ]
    )


def not_data():
    return html.Div(
        [
            html.H2("Resumen"),
            html.P("Universidad:", className="resumen-label"),
            html.P("No disponible"),
            html.P("Titulación:", className="resumen-label"),
            html.P("No disponible"),
            html.P("Docente:", className="resumen-label"),
            html.P("No disponible"),
            html.Hr(),
        ]
    )
```

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 from dash import Input, Output, callback, State
00002 from data.queries import docentes_all
00003
00004
00005 @callback(
00006     Output("docente-dropdown", "value"),
00007     Output("docente-dropdown", "options"),
00008     Output("selected-docente-store", "data"),
00009     Input("docente-dropdown", "value"),
00010     State("selected-docente-store", "data"),
00011 )
00012 def store_selected_docente(selected_value, stored_value):
00013     """
00014 Almacena el docente seleccionado en el store.
00015
00016 Args:
00017 selected_value (str): Valor seleccionado
00018 stored_value (str): Valor almacenado
00019
00020 Returns:
00021 str: Valor seleccionado
00022 list: Opciones del dropdown
00023 str: Valor almacenado
00024 """
00025
00026 data = docentes_all()
00027
00028 if not data:
00029         return None, [], None
00030
00031     opciones_dropdown = [{"label": docente[0], "value": docente[0]} for docente in data]
00032
00033     if selected_value is None and stored_value in [
00034         op["value"] for op in opciones_dropdown
00035     ]:
00036         return stored_value, opciones_dropdown, stored_value
00037
00038     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 from dash import html, callback, Input, Output,State
00002 from components.docente.utils.select_docente import select_docente
00003 from components.common.sidebar import sidebar
```

```
from components.common.filters import filters
from components.docente.utils.resumen_docente import resumen_docente
from components.docente.utils.recomendador_docente import recomendador_docente
from components.docente.filters.filter_curso_academico_docente import filter_curso_academico_docente
from components.docente.filters.filter_asignaturas_docente import filter_asignaturas_docente
from components.docente.filters.filter_titulacion_docente import filter_titulacion_docente
from components.docente.filters.filter_all_curso_academico import filter_all_curso_academico
from components.docente.filters.filter_all_asignaturas_titulacion_docente import
filter_all_asignaturas_titulacion_docente
from components.docente.graphs.graphs_personal_docente import graphs_personal_docente
from components.docente.graphs.graphs_general_docente import graphs_general_docente


@callback(
    Output('tabs-docente-content', 'children'),
    Input('tabs-docente', 'value'),
    State('selected-docente-store', 'data')
    )
def 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:
list: Componentes de la pestaña seleccionada
"""
if tab == 'rendimiento-academico-asignatura-tab':
        return html.Div([
            select_docente(),
            html.H2("Dashboard Docente", style={'textAlign': 'center'}),
            html.Div([
                sidebar([
                  resumen_docente(),
                  filters([
                    filter_titulacion_docente(),
                    filter_asignaturas_docente(),
                    filter_curso_academico_docente()
                  ]),
                ]),
                graphs_personal_docente()
            ], className='content-layout-dashboard')
        ])
    elif tab == 'rendimiento-academico-tab':
        return html.Div([
            html.H2("Dashboard Docente", style={'textAlign': 'center'}),
            html.Div([
                sidebar([
                  resumen_docente(),
                  filters([
                    filter_titulacion_docente(),
                    filter_all_curso_academico(),
                    filter_all_asignaturas_titulacion_docente()
                  ]),
                ]),
                graphs_general_docente()
            ], className='content-layout-dashboard')
        ])
    elif tab == 'recomendaciones-tab':
        return html.Div([
            recomendador_docente()
        ])
```

6.137. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Input, Output, State, callback, callback_context
from data.queries import universidades_gestor, curso_academico_universidad


@callback(
    Output("curso-all-academico-gestor", "options"),
    Output("curso-all-academico-gestor", "value"),
    Input("selected-gestor-store", "data"),
    Input("select-all-curso-academico-button", "n_clicks"),
    State("curso-all-academico-gestor", "options"),
)
def 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
    """

ctx = callback_context
trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None

    if trigger_id == "select-all-curso-academico-button" and n_clicks > 0:
        return existing_options, [option["value"] for option in existing_options]

    if not gestor_id:
        return [], None

    cod_universidad = universidades_gestor(gestor_id)

    if not cod_universidad:
        return [], None

    data = curso_academico_universidad(cod_universidad[0][0])

    if not data:
        return [], None

    opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
    value = (
        [option["value"] for option in opciones_dropdown] if opciones_dropdown else []
    )

    return opciones_dropdown, value
```

6.139. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Input, Output, callback
from data.queries import universidades_gestor, curso_academico_universidad


@callback(
    Output("curso-academico-gestor", "options"),
    Output("curso-academico-gestor", "value"),
    Input("selected-gestor-store", "data"),
)
def 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
    """

if not gestor_id:
        return [], None

    cod_universidad = universidades_gestor(gestor_id)

    if not cod_universidad:
        return [], None

    data = curso_academico_universidad(cod_universidad[0][0])

    if not data:
        return [], None

    opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
    value = opciones_dropdown[0]["value"] if opciones_dropdown else None

    return opciones_dropdown, value
```

6.141. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Input, Output, State, callback, callback_context
from data.queries import universidades_gestor, titulaciones_universidad_gestor


@callback(
    Output("titulaciones-gestor", "options"),
    Output("titulaciones-gestor", "value"),
    Input("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("select-all-titulaciones-gestor", "n_clicks"),
    State("titulaciones-gestor", "options"),
)
def 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" em
    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
    """
ctx = callback_context
trigger_id = ctx.triggered[0]["prop_id"].split(".")[0] if ctx.triggered else None

    if trigger_id == "select-all-titulaciones-gestor" and n_clicks > 0:
        return existing_options, [option["value"] for option in existing_options]

    if not (docente_id and curso_academico):
        return [], None

    cod_universidad = universidades_gestor(docente_id)

    if not cod_universidad:
        return [], None

    data = titulaciones_universidad_gestor(cod_universidad[0][0], curso_academico)

    if not data:
        return [], None

    opciones_dropdown = [{"label": curso[0], "value": curso[0]} for curso in data]
    value = (
        [option["value"] for option in opciones_dropdown] if opciones_dropdown else None
    )

    return opciones_dropdown, value
```

6.143. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, State, callback
00002 import plotly.graph_objects as go
00003 import dash_bootstrap_components as dbc
00004 import pandas as pd
00005 from util import list_to_tuple
00006 from data.queries import alumnos_egresados_genero_titulacion, universidades_gestor
00007
00008
00009 @callback(
00010     Output("egresados-genero-gestor", "figure"),
00011     Input("selected-gestor-store", "data"),
00012     Input("curso-academico-gestor", "value"),
00013     Input("titulaciones-gestor", "value"),
00014 )
00015 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00016     """
00017 Actualiza el gráfico de alumnos egresados por género y titulación
00018 del perfil "Gestor" de la pestaña "Indicadores académicos".
00019
00020     Args:
00021     gestor_id (str): ID del gestor seleccionado
00022     curso_academico (list): Lista con los cursos académicos
00023     titulaciones (list): Lista con las titulaciones seleccionadas
00024
00025     Returns:
00026     go.Figure: Figura con el gráfico
00027
00028     """
00029 fig = go.Figure()
00030
00031 fig.update_layout(
00032 barmode="stack",
00033         title={"text": "Alumnos egresados por género y titulación", "x": 0.5},
00034         xaxis=dict(title="Titulaciones"),
00035         yaxis=dict(title="N° Alumnos"),
00036         showlegend=True,
00037         legend={"title": "Género"},
00038     )
00039
00040     df = get_data(gestor_id, curso_academico, titulaciones)
00041
00042     if df.empty:
00043         return fig
00044
00045     def map_genero(genero):
00046         if "masculin" in genero.lower():
00047             return "Hombres"
00048         elif "fem" in genero.lower():
00049             return "Mujeres"
00050         return genero
00051
00052     df["genero"] = df["genero"].map(map_genero)
00053
00054     # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
00055     df_pivot = df.pivot_table(
00056         index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
00057     ).fillna(0)
00058
00059     if "Hombres" in df_pivot.columns:
```

```
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot["Hombres"],
                name="Hombres",
                marker_color="blue",
                opacity=0.7,
            )
        )

    if "Mujeres" in df_pivot.columns:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot["Mujeres"],
                name="Mujeres",
                marker_color="red",
                opacity=0.7,
            )
        )

    return fig


@callback(
    Output("modal-egresados-genero", "is_open"),
    Input("btn-ver-datos-egresados-genero", "n_clicks"),
    State("modal-egresados-genero", "is_open"),
)
def toggle_modal(btn, is_open):
    """
Alterna 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

    """

if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-egresados-genero", "children"),
    Input("btn-ver-datos-egresados-genero", "n_clicks"),
    Input("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
def 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

    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-egresados-genero", "href"),
    Input("btn-ver-datos-egresados-genero", "n_clicks"),
```

```
    State("selected-gestor-store", "data"),
    State("curso-academico-gestor", "value"),
    State("titulaciones-gestor", "value"),
)
def 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

    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return ""

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


def 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
"""
empty = pd.DataFrame()

if not gestor_id or not curso_academico or not titulaciones:
        return empty

    try:
        titulaciones = list_to_tuple(titulaciones)
    except Exception as e:
        return empty

    data_universidad = universidades_gestor(gestor_id)
    if not data_universidad:
        return empty

    data = alumnos_egresados_genero_titulacion(
        data_universidad[0][0], curso_academico, titulaciones
    )

    if not data:
        return empty

    return pd.DataFrame(data, columns=["titulacion", "genero", "cantidad"])
```

6.145. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, State, callback
00002 import plotly.graph_objs as go
00003 import dash_bootstrap_components as dbc
00004 import pandas as pd
00005 from util import list_to_tuple
00006 from data.queries import alumnos_egresados_nacionalidad_titulacion, universidades_gestor
00007
00008
00009 @callback(
00010     Output("egresados-nacionalidad-gestor", "figure"),
00011     Input("selected-gestor-store", "data"),
00012     Input("curso-academico-gestor", "value"),
00013     Input("titulaciones-gestor", "value"),
00014 )
00015 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00016     """
00017 Actualiza el gráfico de alumnos egresados por nacionalidad y titulación
00018 del perfil "Gestor" de la pestaña "Indicadores académicos".
00019
00020     Args:
00021     gestor_id (str): ID del gestor seleccionado
00022     curso_academico (list): Lista con los cursos académicos
00023     titulaciones (list): Lista con las titulaciones seleccionadas
00024
00025     Returns:
00026     go.Figure: Figura con el gráfico
00027
00028     """
00029
00030 fig = go.Figure()
00031
00032 fig.update_layout(
00033 barmode="stack",
00034         title={"text": "Alumnos egresados por nacionalidad y titulación", "x": 0.5},
00035         xaxis=dict(title="Titulaciones"),
00036         yaxis=dict(title="N° Alumnos"),
00037         showlegend=True,
00038         legend={"title": "Nacionalidad"},
00039     )
00040
00041     df = get_data(gestor_id, curso_academico, titulaciones)
00042
00043     if df.empty:
00044         return fig
00045
00046     df_pivot = df.pivot_table(
00047         index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
00048     ).fillna(0)
00049
```

```
    for nacionalidad in df_pivot.columns:
        fig.add_trace(
            go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
        )

    return fig


@callback(
    Output("modal-egresados-nacionalidad", "is_open"),
    Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
    State("modal-egresados-nacionalidad", "is_open"),
)
def toggle_modal(btn, is_open):
    """
Alterna 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
"""

if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-egresados-nacionalidad", "children"),
    Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
def 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
    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-egresados-nacionalidad", "href"),
    Input("btn-ver-datos-egresados-nacionalidad", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
def 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
    """

if not btn:
```

```
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return ""

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


def 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
    """

empty = pd.DataFrame()

if not gestor_id or not curso_academico or not titulaciones:
        return empty

    data_universidad = universidades_gestor(gestor_id)
    if not data_universidad:
        return empty

    try:
        titulaciones = list_to_tuple(titulaciones)
    except Exception as e:
        return empty

    data = alumnos_egresados_nacionalidad_titulacion(
        data_universidad[0][0], curso_academico, titulaciones
    )

    if not data:
        return empty

    return pd.DataFrame(data, columns=["titulacion", "nacionalidad", "cantidad"])
```

6.147. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, State, callback
00002 import plotly.graph_objs as go
00003 import dash_bootstrap_components as dbc
00004 import pandas as pd
00005 from util import list_to_tuple
00006 from data.queries import alumnos_nuevo_ingreso_genero_titulacion, universidades_gestor
00007
00008
00009 @callback(
00010     Output("nuevo-ingreso-genero-gestor", "figure"),
00011     Input("selected-gestor-store", "data"),
00012     Input("curso-academico-gestor", "value"),
00013     Input("titulaciones-gestor", "value"),
00014 )
00015 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00016     """
00017 Actualiza el gráfico de alumnos de nuevo ingreso por género y titulación
00018 del perfil "Gestor" de la pestaña "Indicadores académicos".
00019
00020     Args:
00021     gestor_id (str): ID del gestor seleccionado
00022     curso_academico (list): Lista con los cursos académicos
00023     titulaciones (list): Lista con las titulaciones seleccionadas
00024
00025     Returns:
00026     go.Figure: Figura con el gráfico
00027
00028     """
00029 fig = go.Figure()
00030
00031 fig.update_layout(
00032 barmode="stack",
00033         title={"text": "Alumnos de nuevo ingreso por género y titulación", "x": 0.5},
00034         xaxis=dict(title="Titulaciones"),
00035         yaxis=dict(title="N° Alumnos"),
00036         showlegend=True,
00037         legend={"title": "Género"},
00038     )
00039
00040     # Asegurarse de que las columnas sean consistentes
00041     def map_genero(genero):
00042         if "masculin" in genero.lower():
00043             return "Hombres"
00044         elif "fem" in genero.lower():
00045             return "Mujeres"
00046         return genero
00047
00048     df = get_data(gestor_id, curso_academico, titulaciones)
00049
00050     if df.empty:
00051         return fig
00052
00053     df["genero"] = df["genero"].map(map_genero)
00054
00055     # Pivotear el DataFrame para obtener las cantidades por género en columnas separadas
00056     df_pivot = df.pivot_table(
00057         index="titulacion", columns="genero", values="cantidad", aggfunc="sum"
00058     ).fillna(0)
00059
00060     # Asegurarse de que las columnas existen antes de graficar
00061     if "Hombres" in df_pivot.columns:
00062         fig.add_trace(
00063             go.Bar(
00064                 x=df_pivot.index,
00065                 y=df_pivot["Hombres"],
00066                 name="Hombres",
00067                 marker_color="blue",
00068                 opacity=0.7,
```

```
            )
        )

    if "Mujeres" in df_pivot.columns:
        fig.add_trace(
            go.Bar(
                x=df_pivot.index,
                y=df_pivot["Mujeres"],
                name="Mujeres",
                marker_color="red",
                opacity=0.7,
            )
        )

    return fig


@callback(
    Output("modal-nuevo-ingreso-genero", "is_open"),
    Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
    State("modal-nuevo-ingreso-genero", "is_open"),
)
def toggle_modal(btn, is_open):
    """
Alterna 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
"""
if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-nuevo-ingreso-genero", "children"),
    Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
    Input("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
def 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
    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-nuevo-ingreso-genero", "href"),
    Input("btn-ver-datos-nuevo-ingreso-genero", "n_clicks"),
    State("selected-gestor-store", "data"),
    State("curso-academico-gestor", "value"),
    State("titulaciones-gestor", "value"),
)
def generate_csv(btn, gestor_id, curso_academico, titulaciones):
    """
Genera un archivo CSV descargable con los datos de los alumnos de nuevo ingres
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
    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)
    if df.empty:
        return ""

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


def 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
"""

empty = pd.DataFrame()

if not gestor_id or not curso_academico or not titulaciones:
        return empty

    try:
        titulaciones = list_to_tuple(titulaciones)
    except Exception as e:
        return empty, None

    data_universidad = universidades_gestor(gestor_id)
    if not data_universidad:
        return empty

    data = alumnos_nuevo_ingreso_genero_titulacion(
        curso_academico, titulaciones, data_universidad[0][0]
    )

    if not data:
        return empty

    return pd.DataFrame(
        data, columns=["curso_academico", "titulacion", "genero", "cantidad"]
    )
```

6.149. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, State, callback
00002 import plotly.graph_objs as go
00003 import dash_bootstrap_components as dbc
00004 import pandas as pd
00005 from util import list_to_tuple
00006 from data.queries import (
00007     alumnos_nuevo_ingreso_nacionalidad_titulacion,
00008     universidades_gestor,
00009 )
00010
00011
00012 @callback(
00013     Output("nuevo_ingreso_nacionalidad-gestor", "figure"),
00014     Input("selected-gestor-store", "data"),
00015     Input("curso-academico-gestor", "value"),
00016     Input("titulaciones-gestor", "value"),
00017 )
00018 def update_graph_gestor(gestor_id, curso_academico, titulaciones):
00019     """
00020 Actualiza el gráfico de alumnos de nuevo ingreso por nacionalidad y titulación
00021 del perfil "Gestor" de la pestaña "Indicadores académicos".
00022
00023     Args:
00024     gestor_id (str): ID del gestor seleccionado
00025     curso_academico (list): Lista con los cursos académicos
00026     titulaciones (list): Lista con las titulaciones seleccionadas
00027
00028     Returns:
00029     go.Figure: Figura con el gráfico
00030
00031     """
00032
00033 fig = go.Figure()
00034
00035 fig.update_layout(
00036 barmode="stack",
00037         title={
00038             "text": "Alumnos de nuevo ingreso por nacionalidad y titulación",
00039             "x": 0.5,
00040         },
00041         xaxis_title="Titulaciones",
00042         yaxis_title="N° Alumnos",
00043         showlegend=True,
00044         legend={"title": "Nacionalidad"},
00045     )
00046
00047     df = get_data(gestor_id, curso_academico, titulaciones)
00048
00049     if df.empty:
00050         return fig
00051
00052     # Pivotear el DataFrame para obtener las cantidades por nacionalidad en columnas separadas
00053     df_pivot = df.pivot_table(
00054         index="titulacion", columns="nacionalidad", values="cantidad", aggfunc="sum"
00055     ).fillna(0)
00056
00057     for nacionalidad in df_pivot.columns:
00058         fig.add_trace(
00059             go.Bar(x=df_pivot.index, y=df_pivot[nacionalidad], name=nacionalidad)
```

```
        )

    return fig


@callback(
    Output("modal-nuevo-ingreso-nacionalidad", "is_open"),
    Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
    State("modal-nuevo-ingreso-nacionalidad", "is_open"),
)
def toggle_modal(btn, is_open):
    """
Alterna 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

    """

if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-nuevo-ingreso-nacionalidad", "children"),
    Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
def 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

    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-nuevo-ingreso-nacionalidad", "href"),
    Input("btn-ver-datos-nuevo-ingreso-nacionalidad", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-academico-gestor", "value"),
    Input("titulaciones-gestor", "value"),
)
def 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

    """
```

```

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico, titulaciones)

    if df.empty:
        return ""

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


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

Args:
gesto_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

"""

empty = pd.DataFrame()

if not gesto_id or not curso_academico or not titulaciones:
        return empty

    data_universidad = universidades_gestor(gesto_id)
    if not data_universidad:
        return empty

    try:
        titulaciones = list_to_tuple(titulaciones)
    except Exception as e:
        return empty

    data = alumnos_nuevo_ingreso_nacionalidad_titulacion(
        data_universidad[0][0], curso_academico, titulaciones
    )

    if not data:
        return empty

    return pd.DataFrame(data, columns=["titulacion", "nacionalidad", "cantidad"])
```

6.151. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from dash import Input, Output, State, callback
00002 import plotly.express as px
00003 import plotly.graph_objects as go
00004 import dash_bootstrap_components as dbc
00005 import pandas as pd
00006 from data.queries import duracion_media_estudios_nota_gestor, universidades_gestor
00007
00008
00009 @callback(
00010     Output("duración-estudios-nota-media-gestor", "figure"),
00011     Input("selected-gestor-store", "data"),
00012 )
00013 def update_graph_gestor(gestor_id):
00014     """
00015 Actualiza el gráfico de duración media de los estudios con respecto a la nota media
00016 del perfil "Gestor" de la pestaña "Resultados académicos".
00017
00018     Args:
00019     gestor_id (str): ID del gestor seleccionado
00020
00021     Returns:
00022     go.Figure: Figura con el gráfico
00023     """
00024
00025 fig = go.Figure()
00026
00027 if not gestor_id:
00028         return fig
00029
00030     data = get_data(gestor_id)
00031
00032     if data.empty:
00033         return fig
00034
00035     fig = px.scatter(
00036         data,
00037         x="nota_media",
00038         y="duracion_media",
00039         color="rama",
00040         size="numero_alumnos",
00041         hover_name="titulacion",
00042         animation_frame="curso_academico",
00043         animation_group="rama",
00044         category_orders={
00045             "rama": sorted(data["rama"].unique())
00046         },
00047     )
00048
00049     fig.update_layout(
00050         title={
00051             "text": "Duración media de los estudios con respecto a la nota media",
00052             "x": 0.5,
00053         },
00054         xaxis_title="Nota media",
00055         yaxis_title="Duración media de los estudios",
00056         legend={"title": "Rama de conocimiento"},
00057     )
00058
00059     fig.update_traces(
00060         marker=dict(line=dict(width=1, color="DarkSlateGrey")),
00061         textposition="top center"
00062     )
00063
00064     # Ajustar el rango de los ejes para que las burbujas no se corten
00065     fig.update_xaxes(range=[data["nota_media"].min() - 1, data["nota_media"].max() + 1])
```

```
    fig.update_yaxes(
        range=[data["duracion_media"].min() - 1, data["duracion_media"].max() + 1]
    )

    if len(data["curso_academico"].unique()) > 1:
        fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 1000

    return fig


@callback(
    Output("modal-duracion-estudios", "is_open"),
    Input("btn-ver-datos-duracion-estudios", "n_clicks"),
    State("modal-duracion-estudios", "is_open"),
)
def toggle_modal(btn, is_open):
    """
Alterna 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
"""
if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-duracion-estudios", "children"),
    Input("btn-ver-datos-duracion-estudios", "n_clicks"),
    State("selected-gestor-store", "data"),
)
def 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
    """
if not btn:
        return ""

    df = get_data(gestor_id)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-csv-duracion-estudios", "href"),
    Input("btn-descargar-csv-duracion-estudios", "n_clicks"),
    State("selected-gestor-store", "data"),
)
def 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
    """

if not btn:
        return ""

    df = get_data(gestor_id)

    if df.empty:
        return ""
```

```python

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


def 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
"""
empty = pd.DataFrame()

if not gestor_id:
        return empty

    data_universidad = universidades_gestor(gestor_id)
    if not data_universidad:
        return empty

    data = duracion_media_estudios_nota_gestor(data_universidad[0][0])
    if not data:
        return empty

    df = pd.DataFrame(
        data,
        columns=[
            "nota_media",
            "titulacion",
            "rama",
            "curso_academico",
            "duracion_media",
            "numero_alumnos",
        ],
    )
    return df
```

6.153. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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_grpht_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 from dash import Input, Output, State, callback
00002 import plotly.graph_objs as go
00003 import dash_bootstrap_components as dbc
00004 import pandas as pd
00005 from data.queries import nota_media_acceso_titulacion, universidades_gestor
00006
00007
00008 @callback(
00009     Output("nota-acceso-titulacion", "figure"),
00010     Input("selected-gestor-store", "data")
00011 )
00012 def update_grpht_gestor(gestor_id):
00013     """
00014 Actualiza el gráfico de evolución de la nota de acceso por titulación
00015 del perfil "Gestor" de la pestaña "Resultados académicos".
00016
00017     Args:
00018     gestor_id (str): ID del gestor seleccionado
00019
00020     Returns:
00021     go.Figure: Figura con el gráfico
00022
00023     """
00024
00025 fig = go.Figure()
00026
00027 fig.update_layout(
00028 title={"text": "Evolución nota de acceso por titulación", "x": 0.5},
00029         xaxis_title="Curso académico",
00030         yaxis_title="Nota media de acceso",
00031         legend={"title": "Titulaciones"},
00032     )
00033
00034     df = get_data(gestor_id)
00035
00036     if df.empty:
00037         return fig
00038
00039     for titulation, group in df.groupby("titulacion"):
00040         fig.add_trace(
00041             go.Scatter(
00042                 x=group["curso_academico"],
00043                 y=group["nota"],
00044                 mode="lines+markers",
00045                 name=titulation,
00046             )
00047         )
00048
00049     return fig
00050
00051
00052 @callback(
00053     Output("modal-nota-acceso", "is_open"),
00054     Input("btn-ver-datos-nota-acceso", "n_clicks"),
00055     State("modal-nota-acceso", "is_open"),
00056 )
00057 def toggle_modal(btn, is_open):
00058     """
00059 Alternar la visibilidad del modal de datos de la nota de acceso por titulación
00060 del perfil "Gestor" de la pestaña "Resultados académicos".
00061
00062     Args:
00063     btn (int): Número de clicks en el botón
00064     is_open (bool): Estado actual del modal
00065
00066     Returns:
00067     bool: Nuevo estado del modal
00068
00069     """
00070 if btn:
00071         return not is_open
00072     return is_open
00073
00074
00075 @callback(
00076     Output("table-container-nota-acceso", "children"),
00077     Input("btn-ver-datos-nota-acceso", "n_clicks"),
00078     State("selected-gestor-store", "data"),
00079 )
00080 def update_table(btn, gestor_id):
00081     """
00082 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

    """

if not btn:
        return ""

    df = get_data(gestor_id)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True
    )


@callback(
    Output("btn-descargar-csv-nota-acceso", "href"),
    Input("btn-ver-datos-nota-acceso", "n_clicks"),
    State("selected-gestor-store", "data"),
)
def 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

    """

if not btn:
        return ""

    df = get_data(gestor_id)

    if df.empty:
        return ""

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


def 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
"""
empty = pd.DataFrame()

if not gestor_id:
        return empty

    data_universidad = universidades_gestor(gestor_id)

    if not data_universidad:
        return empty

    data = nota_media_acceso_titulacion(data_universidad[0][0])

    if not data:
        return empty

    return pd.DataFrame(data, columns=["curso_academico", "titulacion", "nota"])
```

6.155. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import Output, Input, State, callback
import plotly.graph_objs as go
import dash_bootstrap_components as dbc
import pandas as pd
from data.queries import tasa_abandono_titulacion_gestor, universidades_gestor
from util import list_to_tuple


@callback(
    Output("tasa-abandono-gestor", "figure"),
    Input("curso-all-academico-gestor", "value"),
    Input("selected-gestor-store", "data"),
)
def 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
    """
fig = go.Figure()

fig.update_layout(
title={"text": "Tasa de abandono por titulación ", "x": 0.5},
        xaxis_title="Curso académico",
        yaxis_title="Tasa de abandono (%)",
        showlegend=True,
        legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
    )
```

```python

    df = get_data(gestor_id, curso_academico)

    if df.empty:
        return fig

    df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100

    for titulacion in df["titulacion"].unique():
        df_titulacion = df[df["titulacion"] == titulacion]
        fig.add_trace(
            go.Scatter(
                x=df_titulacion["curso_academico"],
                y=df_titulacion["tasa_abandono"],
                mode="lines+markers",
                name=titulacion,
            )
        )

    return fig


@callback(
    Output("modal-tasa-abandono", "is_open"),
    Input("btn-ver-datos-tasa-abandono", "n_clicks"),
    State("modal-tasa-abandono", "is_open"),
)
def 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
    """

if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-tasa-abandono", "children"),
    Input("btn-ver-datos-tasa-abandono", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-all-academico-gestor", "value"),
)
def 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
    """

if not btn:
        return ""

    df = get_data(gestor_id, curso_academico)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True, responsive=True
    )


@callback(
    Output("btn-descargar-tasa-abandono", "href"),
    Input("btn-ver-datos-tasa-abandono", "n_clicks"),
    State("selected-gestor-store", "data"),
    Input("curso-all-academico-gestor", "value"),
)
def 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
    """

if not btn:
        return ""
    df = get_data(gestor_id, curso_academico)

    if df.empty:
        return ""

    df["tasa_abandono"] = (df["numero_abandonos"] / df["numero_matriculados"]) * 100
    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string
    return csv_string


def 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
"""
empty = pd.DataFrame()

if not gestor_id or not curso_academico:
        return empty

    data_universidad = universidades_gestor(gestor_id)
    if not data_universidad:
        return empty
    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        print("Error:", e)
        return empty

    data = tasa_abandono_titulacion_gestor(data_universidad[0][0], curso_academico)

    if not data:
        return empty

    return pd.DataFrame(
        data,
        columns=[
            "curso_academico",
            "titulacion",
            "numero_matriculados",
            "numero_abandonos",
        ],
    )
```

6.157. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 from dash import Input, Output, State, callback
00002 import plotly.graph_objs as go
00003 import dash_bootstrap_components as dbc
00004 import pandas as pd
00005 from util import list_to_tuple
00006 from data.queries import tasa_graduacion_titulacion_gestor, universidades_gestor
00007
00008
00009 @callback(
00010     Output("tasa-graduacion-gestor", "figure"),
00011     Input("curso-all-academico-gestor", "value"),
00012     Input("selected-gestor-store", "data"),
00013 )
00014 def update_graph_gestor(curso_academico, gestor_id):
00015     """
00016 Actualiza el gráfico de la tasa de graduación por titulación
00017 del perfil "Gestor" de la pestaña "Riesgo de abandono".
00018
00019     Args:
00020     curso_academico (list): Lista con los cursos académicos
00021     gestor_id (str): ID del gestor seleccionado
00022
00023     Returns:
00024     go.Figure: Figura con el gráfico
00025     """
00026
00027 fig = go.Figure()
00028
00029 fig.update_layout(
00030 title={"text": "Tasa de graduación por titulación ", "x": 0.5},
00031         xaxis_title="Curso académico",
00032         yaxis_title="Tasa de graduación (%)",
00033         showlegend=True,
00034         legend={"title": "Titulaciones", "orientation": "h", "y": -0.5},
00035     )
00036
00037     df = get_data(gestor_id, curso_academico)
00038
00039     if df.empty:
00040         return fig
00041
00042     for titulacion, group in df.groupby("titulacion"):
00043         fig.add_trace(
00044             go.Scatter(
00045                 x=group["curso_academico"],
00046                 y=group["tasa_graduacion"],
00047                 mode="lines+markers",
00048                 name=titulacion,
00049             )
00050         )
00051
00052     return fig
00053
00054
00055 @callback(
00056     Output("modal-tasa-graduacion", "is_open"),
00057     Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
00058     State("modal-tasa-graduacion", "is_open"),
00059 )
```

```
def 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
    """

if btn:
        return not is_open
    return is_open


@callback(
    Output("table-container-tasa-graduacion", "children"),
    Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
    State("curso-all-academico-gestor", "value"),
    State("selected-gestor-store", "data"),
)
def 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
    """
if not btn:
        return ""

    df = get_data(gestor_id, curso_academico)

    if df.empty:
        return dbc.Alert("No hay datos disponibles", color="info")

    return dbc.Table.from_dataframe(
        df.head(50), striped=True, bordered=True, hover=True, responsive=True
    )


@callback(
    Output("btn-descargar-tasa-graduacion", "href"),
    Input("btn-ver-datos-tasa-graduacion", "n_clicks"),
    State("curso-all-academico-gestor", "value"),
    State("selected-gestor-store", "data"),
)
def 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
    """

if not n:
        return None

    df = get_data(gestor_id, curso_academico)

    if df is None:
        return None

    csv_string = df.to_csv(index=False, encoding="utf-8")
    csv_string = "data:text/csv;charset=utf-8," + csv_string

    return csv_string


def 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
    """

empty = pd.DataFrame()

if not gestor_id or not curso_academico:
        return empty

    try:
        curso_academico = list_to_tuple(curso_academico)
    except Exception as e:
        print("Error:", e)
        return empty

    data_universidad = universidades_gestor(gestor_id)

    if not data_universidad:
        return empty

    data = tasa_graduacion_titulacion_gestor(data_universidad[0][0], curso_academico)

    if not data:
        return empty

    df = pd.DataFrame(
        data,
        columns=[
            "numero_matriculados",
            "numero_egresados",
            "curso_academico",
            "titulacion",
        ],
    )
    df["tasa_graduacion"] = (df["numero_egresados"] / df["numero_matriculados"]) * 100

    return df
```

6.159. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import html, Output, Input, callback
from data.queries import numero_alumnos_matriculados_universidad, universidades_gestor


@callback(
        Output("resumen-gestor", "children"),
        Input("selected-gestor-store", "data")
)
```

```
def 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
"""

if not gestor_id:
        return not_data()

    data = universidades_gestor(gestor_id)

    if not data:
        return not_data()

    data_alumnos = numero_alumnos_matriculados_universidad(data[0][1])

    if not data_alumnos:
        return not_data()

    return html.Div(
        [
            html.H2("Resumen"),
            html.P("Universidad:", className="resumen-label"),
            html.P(data[0][1]),
            html.P("Gestor:", className="resumen-label"),
            html.P(gestor_id),
            html.P("Número de alumnos matriculados:", className="resumen-label"),
            html.P(data_alumnos[0][0]),
            html.Hr(),
        ]
    )


def not_data():
    return html.Div(
        [
            html.H2("Resumen"),
            html.P("Universidad:", className="resumen-label"),
            html.P("No disponible"),
            html.P("Gestor:", className="resumen-label"),
            html.P("No disponible"),
            html.P("Número de alumnos matriculados:", className="resumen-label"),
            html.P("No disponible"),
            html.Hr(),
        ]
    )
```

6.161. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import Input, Output, callback, State
from data.queries import gestores_all


@callback(
    Output("gestor-dropdown", "value"),
    Output("gestor-dropdown", "options"),
    Output("selected-gestor-store", "data"),
    Input("gestor-dropdown", "value"),
    State("selected-gestor-store", "data"),
)
def 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
"""

data = gestores_all()

if not data:
        return None, [], None

    opciones_dropdown = [{"label": gestor[0], "value": gestor[0]} for gestor in data]

    if selected_value is None and stored_value in [
        op["value"] for op in opciones_dropdown
    ]:
        return stored_value, opciones_dropdown, stored_value

    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.

```
from dash import Input, Output, State, html, callback
from components.common.sidebar import sidebar
from components.common.filters import filters
```

```
from components.gestor.utils.select_gestor import select_gestor
from components.gestor.utils.resumen_gestor import resumen_gestor
from components.gestor.utils.recomendador_gestor import recomendador_gestor
from components.gestor.filters.filter_curso_academico_gestor import filter_curso_academico_gestor
from components.gestor.filters.filter_titulaciones_gestor import filter_titulaciones_gestor
from components.gestor.filters.filter_all_curso_academico_gestor import
filter_all_curso_academico_gestor
from components.gestor.graphs.graphs_indicadores_gestor import graphs_indicadores_gestor
from components.gestor.graphs.graphs_resultados_gestor import graphs_resultados_gestor
from components.gestor.graphs.graphs_riesgo_abandono_gestor import graphs_riesgo_abandono_gestor

@callback(
    Output('tabs-gestor-content', 'children'),
    Input('tabs-gestor', 'value'),
    State('selected-gestor-store', 'data')
)
def 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
    """

if tab == 'indicadores-academicos-tab':
        return html.Div([
            select_gestor(),
            html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
            html.Div([
                sidebar([
                    resumen_gestor(),
                    filters([
                        filter_curso_academico_gestor(),
                        filter_titulaciones_gestor()
                    ]),
                ]),
                graphs_indicadores_gestor()
            ], className='content-layout-dashboard')

        ])
    elif tab == 'resultados-academicos-tab':
        return html.Div([
            html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
             html.Div([
                sidebar([
                    resumen_gestor(),
                ]),
                graphs_resultados_gestor()
            ], className='content-layout-dashboard')
        ])
    elif tab == 'riesgo-abandono-tab':
        return html.Div([
           html.H2("Dashboard Gestor", style={'textAlign': 'center'}),
             html.Div([
                sidebar([
                    resumen_gestor(),
                    filters([
                        filter_all_curso_academico_gestor()
                    ]),
                ]),
                graphs_riesgo_abandono_gestor()
            ], className='content-layout-dashboard')
        ])
    elif tab == 'recomendaciones-tab':
        return html.Div([
            recomendador_gestor()
        ])
```

6.165. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 from dash import html, dcc
00002 from callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado import
      update_filter_asignaturas_matri_alumnado
00003
00004
00005 def filter_asignaturas_matri_alumnado():
00006     """
00007 Crea un componente con un dropdown que contiene las asignaturas matriculadas por el alumno.
00008
00009 Returns:
00010 html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
00011 """
00012 return html.Div(
00013         [
00014             html.Br(),
00015             html.Label("Asignaturas matriculadas"),
00016             dcc.Dropdown(
00017                 id="asignaturas-matriculadas",
00018                 searchable=True,
00019                 multi=True,
00020                 value=None,
00021                 clearable=True,
00022                 options=[],
00023                 maxHeight=300,
00024                 optionHeight=50,
00025                 placeholder="Seleccione una opción",
00026             ),
00027             html.Button(
00028                 "Seleccionar todo",
00029                 id="select-all-button",
00030                 className="button-select-all-filter",
00031                 n_clicks=0,
00032             ),
00033         ]
00034     )
```

6.167. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html, dcc
from callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado import
update_filter_curso_academico_alumnado


def 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
"""
return html.Div(
        [
            html.Br(),
            html.Label("Curso académico"),
            dcc.Dropdown(
                id="curso-academico",
                searchable=False,
                multi=True,
                clearable=True,
                options=[],
                value=None,
                maxHeight=300,
                placeholder="Seleccione una opción",
            ),
            html.Button(
                "Seleccionar todo",
                id="select-all-cursos-academicos",
                className="button-select-all-filter",
                n_clicks=0,
            ),
        ]
    )
```

6.169. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.filter_titulacion_alumnado ()

6.170. filter_titulacion_alumnado.py

Ir a la documentación de este archivo.

```
from dash import html, dcc
from callbacks.alumnado.filters.callback_filter_titulacion_alumnado import update_filter_titulacion_alumnado


def 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
    """
return html.Div(
        [
            html.Label("Titulación"),
            dcc.Dropdown(
                id="titulacion-alumnado",
                options=[],
                value=None,
                searchable=False,
                clearable=False,
                optionHeight=50,
                maxHeight=300,
                placeholder="Seleccione una opción",
            ),
            dcc.Store(id="selected-titulacion-alumnado-store", storage_type="local"),
        ]
    )
```

6.171. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html
from callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa import update_graph_alumnado
from callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_nota import update_graph_alumnado
from callbacks.alumnado.graphs.general.callback_graph_asig_superadas_media import update_graph_alumnado
from components.common.create_graph import create_graph


def graphs_general_alumnado():
    """
```

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

    Returns:
    html.Div: Gráficos
    """
graph_ids = [
"asignaturas-superadas-general-mi-nota",
        "nota-cualitativa-general-mi-nota",
        "nota-media-general-mi-nota",
    ]

    config = {"displayModeBar": False}
    item_class = "graph-item-general-alumnado"

    return html.Div(
        [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
        className="graphs-container-general-alumnado",
    )
```

6.173. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html
from callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado import
update_graph_alumnado
from callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado import
update_graph_alumnado
from callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_alumnado import
update_graph_alumnado
from callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado import
update_graph_alumnado
from components.common.create_graph import create_graph


def graphs_personal_alumnado():
    """
Retorna los gráficos de la pestaña "Expediente académico Personal" del perfil "Alumnado"

    Returns:
    html.Div: Gráficos
    """
graph_ids = [
"graph-evolucion-progreso-academico",
        "graph-bar-evolucion-asignaturas-matriculadas",
        "graph-bar-calificaciones-por-asignatura",
        "graph-bar-tasa-rendimiento",
    ]

    config = {"displayModeBar": False}
    item_class = "graph-item-personal-alumnado"

    return html.Div(
        [create_graph(graph_id, item_class, config) for graph_id in graph_ids],
        className="graphs-container-personal-alumnado",
    )
```

6.175. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.recomendador_alumnado ()

6.176. recomendador_alumnado.py

Ir a la documentación de este archivo.

```
00001 from dash import html
00002
00003 def recomendador_alumnado():
00004     """
00005 Esta función retorna el contenido de la pestaña "Recomendador" del perfil "Alumnado".
00006
00007     Returns:
00008     html.Div: Layout de la pestaña "Recomendador" del perfil "Alumnado"
00009     """
00010
00011 return html.Div([
00012         html.H1("Deserción universitaria: ¿Cuáles son las razones y cómo prevenirla?",
      className='titulo-recomendador-alumnado'),
00013         html.Img(src='assets/images/abandono_academico.jpg',
      className='imagen-recomendador-alumnado'),
00014         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'),
00015         html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-alumnado'),
00016         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'),
00017         html.Br(),
00018         html.Br(),
00019         html.H2("¿Cuáles son las razones de la deserción universitaria?",
      className='sb-recomendador-alumnado'),
00020         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'),
00021         html.Ul([
00022             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'),
00023             html.Li(html.Span([html.Strong("Problemas académicos:"), " Las dificultades para aprobar
      asignaturas pueden hacer que te sientas frustrado."]), className='li-recomendador-alumnado'),
00024             html.Li(html.Span([html.Strong("Problemas personales:"), " Situaciones como problemas de
      salud, familiares o relaciones personales pueden afectarte."]), className='li-recomendador-alumnado'),
00025             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'),
00026             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'),
00027         ], className='ul-recomendador-alumnado'),
00028         html.H2("¿Cómo prevenir la deserción universitaria?", className='sb-recomendador-alumnado'),
00029         html.P("Para prevenir la deserción, puedes buscar ayuda en los siguientes recursos que la
      universidad ofrece:", className='p-recomendador-alumnado'),
00030         html.Ul([
00031             html.Li("Solicitar becas y ayudas económicas si tienes dificultades financieras.",
      className='li-recomendador-alumnado'),
00032             html.Li("Participar en programas de tutoría y apoyo académico si tienes problemas con tus
      asignaturas.", className='li-recomendador-alumnado'),
00033             html.Li("Aprovechar los programas de orientación y asesoramiento para adaptarte mejor a la
      vida universitaria.", className='li-recomendador-alumnado'),
00034             html.Li("Asistir a actividades y talleres que mejoren tu motivación y habilidades de
      estudio.", className='li-recomendador-alumnado'),
00035         ], className='ul-recomendador-alumnado'),
00036     ])
```

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 from dash import html
00002 from callbacks.alumnado.utils.callback_resumen_alumnado import update_resumen_alumnado
00003
00004
00005 def resumen_alumnado():
00006     """
00007 Crea el layout del resumen del alumnado.
00008
00009 Returns:
00010 html.Div: Layout del resumen del alumnado
00011 """
00012 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.

```
from dash import html, dcc
from callbacks.alumnado.utils.callback_select_alumnado import store_selected_alumnado


def select_alumnado():
    """
Crea el desplegable para seleccionar un alumno.

Returns:
html.Div: Desplegable para seleccionar un alumno
"""
return html.Div(
        [
            html.Div(
                [
                    dcc.Dropdown(
                        options=[],
                        value=None,
                        id="alumnado-dropdown",
                        clearable=False,
                        placeholder="Seleccione un alumno",
                    )
                ],
                className="select-alumnado",
            ),
        ]
    )
```

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.

```
from dash import html, dcc
from callbacks.alumnado.utils.callback_tabs_alumnado import render_content


def tabs_alumnado():
    """
Contiene las pestañas de la vista del alumnado.

Returns:
html.Div: Pestañas del alumnado
"""
return html.Div(
        [
            dcc.Tabs(
```

```
                id="tabs-alumnado",
                value="expediente-personal-tab",
                children=[
                    dcc.Tab(
                        label="Expediente académico personal",
                        value="expediente-personal-tab",
                    ),
                    dcc.Tab(
                        label="Rendimiento académico general",
                        value="rendimiento-academico-tab",
                    ),
                    dcc.Tab(label="Recomendaciones", value="recomendador-tab"),
                ],
                className="tabs",
            ),
            html.Div(id="tabs-alumnado-content"),
            dcc.Store(id="selected-alumnado-store", storage_type="local"),
            dcc.Location(id="url", refresh=False),
        ]
    )
```

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.

```
from dash import html, dcc


def 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
"""
return html.Div(
        [dcc.Loading(children=[dcc.Graph(id=graph_id, figure={}, config=config)])],
        className=item_class,
    )
```

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 from components.common.create_graph import create_graph
00002 from components.common.modal_data import create_modal
00003 from dash import html
00004
00005
00006 def create_graph_with_table(graph_id, item_class, config, modal_config):
00007     """
00008 Crea un gráfico con un modal para mostrar los datos y descargarlos.
00009
00010 Args:
00011 graph_id (str): ID del gráfico
00012 item_class (str): Clase del gráfico
00013 config (dict): Configuración del gráfico
00014 modal_config (dict): Configuración del modal
00015
00016 Returns:
00017 html.Div: Gráfico con modal para mostrar los datos
00018 """
00019 return html.Div(
00020         [create_graph(graph_id, item_class, config), create_modal(**modal_config)],
00021         className=item_class,
00022     )
```

6.187. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html


def 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
"""
return html.Div([html.H2("Filtros"), html.Div(filters)], className="filters")
```

6.189. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html

def footer():
    """
Retorna el footer de la aplicación

Returns:
html.Div: Footer de la aplicación
"""
return html.Div([
        html.P('2023-2024 Universidad de la Laguna - Visualización de datos académicos'),
        html.P('Pabellón de Gobierno, C/ Padre Herrera s/n Apartado Postal 456 38200, San Cristóbal de La Laguna'),
        html.P('Santa Cruz de Tenerife - España')], className='footer')
```

6.191. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html, dcc
from callbacks.common.callback_user_role import initialize_dropdown, update_role


def 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
"""
return html.Div(
        [
            html.Img(src="assets/images/logoULL.png", className="logo"),
            html.H1("Visualización de datos académicos", className="title"),
            dcc.Dropdown(
                options=[
                    {"label": "Alumno", "value": "Alumno"},
                    {"label": "Docente", "value": "Docente"},
                    {"label": "Gestor", "value": "Gestor"},
                ],
                id="dropdown_role",
                className="dropdown_role",
                clearable=False,
                searchable=False,
                placeholder="Selecciona un rol",
                value="Alumno",
            ),
            dcc.Store(id=store_role, storage_type="session"),
        ],
        className="header",
    )
```

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.

```
from dash import html
import dash_bootstrap_components as dbc


def 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
"""
return html.Div([
        dbc.Button('Ver datos', id=view_data_button_id, n_clicks=0, color='primary'),
        dbc.Modal([
            dbc.ModalHeader("Datos"),
            dbc.ModalBody(html.Div(id=table_container_id), style={'maxHeight': 'calc(100vh - 200px)', 'overflowY': 'auto'}),
            dbc.ModalFooter(
                dbc.Button("Descargar CSV", id=download_button_id, color='primary')
            )
        ], id=modal_id, is_open=False, size="lg")
    ])
```

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.

```
from dash import html
import dash_bootstrap_components as dbc
from callbacks.common.callback_sidebar_collapse import sidebar_collapse


def 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

"""
sidebar_content = html.Div(components, className="sidebar")
    sidebar_collapse = dbc.Collapse(sidebar_content, id="collapse", is_open=True)

    return html.Div(sidebar_collapse)
```

6.197. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 from dash import html, dcc
00002 from callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente import
      update_filter_asignaturas_docente
00003
00004
00005 def filter_all_asignaturas_titulacion_docente():
00006     """
00007 Crea un componente con un dropdown que contiene todas las asignaturas de la titulación del perfil
      "Docente"
00008     en la pestaña "Rendimiento académico general".
00009
00010     Returns:
00011     html.Div: Componente con un dropdown y un botón para seleccionar todas las asignaturas
00012     """
00013 return html.Div(
00014         [
00015             html.Br(),
00016             html.Label("Asignaturas"),
00017             dcc.Dropdown(
00018                 id="all-asignaturas-titulacion-docente",
00019                 searchable=True,
00020                 value=None,
00021                 clearable=True,
00022                 options=[],
00023                 maxHeight=300,
00024                 optionHeight=50,
00025                 persistence=True,
00026                 persistence_type="local",
00027                 multi=True,
00028             ),
00029             html.Button(
00030                 "Seleccionar todo",
00031                 id="select-all-asignaturas-titulacion-docente",
00032                 className="button-select-all-filter",
00033                 n_clicks=0,
00034             ),
00035         ]
00036     )
```

6.199. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 from dash import html, dcc
00002 from callbacks.docente.filters.callback_filter_all_curso_academico import
      update_filter_all_cursos_academicos_docente
00003
00004
00005 def filter_all_curso_academico():
00006     """
00007 Crea un componente con un dropdown que contiene todos los cursos académicos del perfil "Docente"
00008     en la pestaña "Rendimiento académico general".
00009
00010     Returns:
00011     html.Div: Componente con un dropdown
00012     """
00013 return html.Div(
00014         [
00015             html.Br(),
00016             html.Label("Curso académico"),
00017             dcc.Dropdown(
00018                 id="all-cursos-academicos-docente",
00019                 searchable=False,
00020                 multi=False,
00021                 clearable=False,
00022                 options=[],
00023                 value=None,
00024                 maxHeight=300,
00025             ),
00026         ]
00027     )
```

6.201. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html, dcc
from callbacks.docente.filters.callback_filter_asignaturas_docente import
update_filter_asignaturas_docente


def 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
    """

return html.Div([
        html.Br(),
        html.Label("Asignaturas"),
        dcc.Dropdown(
            id="asignaturas-docente",
            searchable=True,
            value=None,
            clearable=True,
            options=[],
            maxHeight=300,
            optionHeight=50,
        )
    ])
```

6.203. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html, dcc
from callbacks.docente.filters.callback_filter_curso_academico_docente import
update_filter_curso_academico_docente


def 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
    """
return html.Div(
        [
            html.Br(),
            html.Label("Curso académico"),
            dcc.Dropdown(
                id="curso-academico-docente",
                searchable=False,
                multi=True,
                clearable=True,
                options=[],
                value=None,
                maxHeight=300,
            ),
            html.Button(
                "Seleccionar todo",
                id="select-all-cursos-academicos-docente",
                className="button-select-all-filter",
                n_clicks=0,
            ),
        ]
    )
```

6.205. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import html, dcc
from callbacks.docente.filters.callback_filter_titulacion_docente import
update_filter_titulacion_docente


def 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
    """

return html.Div(
        [
            html.Label("Titulación"),
            dcc.Dropdown(
                id="titulacion-docente",
                options=[],
                value=None,
                searchable=False,
                clearable=False,
                optionHeight=50,
                maxHeight=300,
            ),
            dcc.Store(id="selected-titulacion-docente-store", storage_type="local"),
        ]
    )
```

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.

```
from dash import html
from callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente import
update_graph_docente
from callbacks.docente.graphs.general.callback_graph_all_calif_media_docente import
update_graph_docente
from components.common.create_graph import create_graph
from util import config_mode_bar_buttons_gestor


def graphs_general_docente():
    """
Retorna los gráficos de la pestaña "Rendimiento académico general"
    del perfil "Docente"

    Returns:
    html.Div: Gráficos
    """

graphs_ids = [
"calificaiones-cuali-all-asig-docente",
        "calificaiones-media-all-asig-docente",
    ]

    item_class = "graph-item-general-docente"
    config = config_mode_bar_buttons_gestor

    return html.Div(
        [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
        className="graphs-container-general-docente",
    )
```

6.209. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 from dash import html
00002 from callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente import
      update_graph_docente
00003 from callbacks.docente.graphs.personal.callback_graph_alu_genero_docente import update_graph_docente
00004 from callbacks.docente.graphs.personal.callback_graph_alu_media_docente import update_graph_docente
00005 from callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente import
      update_graph_docente
00006 from components.common.create_graph import create_graph
00007 from util import config_mode_bar_buttons_gestor
00008
00009
00010 def graphs_personal_docente():
00011     """
00012 Retorna los gráficos de la pestaña "Rendimiento académico personal"
00013     del perfil "Docente"
00014     """
00015 graphs_ids = [
00016 "graph-alumnos-repetidores-nuevos",
00017         "graph-alumnos-matri-genero",
00018         "graph-alumnos-nota-media",
00019         "graph-alumnos-nota-cualitativa",
00020     ]
00021
00022     item_class = "graph-item-personal-docente"
00023     config = config_mode_bar_buttons_gestor
00024
00025     return html.Div(
00026         [create_graph(graph_id, item_class, config) for graph_id in graphs_ids],
00027         className="graphs-container-personal-docente",
00028     )
00029
```

6.211. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html

def recomendador_docente():
    """
Retorna el componente del recomendador para docentes.

Returns:
html.Div: Componente del recomendador para docentes
"""

return html.Div([
        html.H1("Deserción universitaria: ¿Cómo pueden ayudar los docentes a prevenirla?", className='titulo-recomendador-docente'),
        html.Img(src='assets/images/docente_clase.jpg', className='imagen-recomendador-docente'),
        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'),
        html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-docente'),
        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'),
        html.Br(),
        html.H2("¿Cómo pueden los docentes ayudar a prevenir la deserción universitaria?", className='sb-recomendador-docente'),
        html.Ul([
            html.Li("Identificar y apoyar a los estudiantes con dificultades académicas proporcionando tutorías y asesorías.", className='li-recomendador-docente'),
            html.Li("Fomentar un ambiente de clase inclusivo y motivador que promueva la participación y el interés por el aprendizaje.", className='li-recomendador-docente'),
            html.Li("Mantener una comunicación abierta y constante con los estudiantes para entender sus problemas y necesidades.", className='li-recomendador-docente'),
            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'),
            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'),
        ], className='ul-recomendador-docente'),
    ])
```

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.

```
from dash import html
from callbacks.docente.utils.callback_resumen_docente import update_resumen_docente


def resumen_docente():
    """
Crea el layout del resumen del docente

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

```
from dash import html, dcc
from callbacks.docente.utils.callback_select_docente import store_selected_docente


def select_docente():
    """
Crea el desplegable para seleccionar un docente.

Returns:
html.Div: Desplegable para seleccionar un docente

"""
return html.Div(
        [
            html.Div(
                [
                    dcc.Dropdown(
                        options=[],
                        value=None,
                        id="docente-dropdown",
                        clearable=False,
                        placeholder="Selecciona un docente",
                    )
                ],
                className="select-docente",
            ),
        ]
    )
```

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.

```
from dash import html, dcc
from callbacks.docente.utils.callback_tabs_docente import render_content


def tabs_docente():
    """
Crea las pestañas del perfil "Docente"

    Returns:
    html.Div: Pestañas del docente
    """

return html.Div(
        [
            dcc.Tabs(
                id="tabs-docente",
                value="rendimiento-academico-asignatura-tab",
                children=[
                    dcc.Tab(
                        label="Rendimiento académico personal",
                        value="rendimiento-academico-asignatura-tab",
                    ),
                    dcc.Tab(
                        label="Rendimiento académico general",
                        value="rendimiento-academico-tab",
                    ),
                    dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
                ],
                className="tabs",
            ),
            html.Div(id="tabs-docente-content"),
            dcc.Store(id="selected-docente-store", storage_type="local"),
        ]
    )
```

6.219. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html, dcc
from callbacks.gestor.filters.callback_filter_all_curso_academico_gestor import
update_filter_all_curso_academico_gestor


def 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
    """

return html.Div(
        [
            html.Label("Curso académico"),
            dcc.Dropdown(
                id="curso-all-academico-gestor",
                searchable=True,
                multi=True,
                clearable=True,
                options=[],
                value=None,
                maxHeight=200,
                placeholder="Seleccione una opción",
            ),
            html.Button(
                "Seleccionar todo",
                id="select-all-curso-academico-button",
                className="button-select-all-filter",
                n_clicks=0,
            ),
            dcc.Store(id="curso-all-academico-gestor-store", storage_type="local"),
        ]
    )
```

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.

```
from dash import html, dcc
from callbacks.gestor.filters.callback_filter_curso_academico_gestor import
update_filter_curso_academico_gestor


def 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
    """
return html.Div(
        [
            html.Label("Curso académico"),
            dcc.Dropdown(
                id="curso-academico-gestor",
                searchable=True,
                multi=False,
                clearable=False,
                options=[],
                value=None,
                maxHeight=300,
                placeholder="Selecciona una opción",
            ),
        ]
    )
```

6.223. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.

```
from dash import html, dcc
from callbacks.gestor.filters.callback_filter_titulaciones_gestor import
update_filter_titulaciones_gestor


def 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
    """
```

```
return html.Div(
        [
            html.Br(),
            html.Label("Titulaciones"),
            dcc.Dropdown(
                id="titulaciones-gestor",
                searchable=True,
                multi=True,
                clearable=True,
                options=[],
                value=None,
                maxHeight=200,
                placeholder="Seleccione una opción",
            ),
            html.Button(
                "Seleccionar todo",
                id="select-all-titulaciones-gestor",
                className="button-select-all-filter",
                n_clicks=0,
            ),
        ]
    )
```

6.225. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.graphs_indicadores_gestor ()

6.226. graphs_indicadores_gestor.py

Ir a la documentación de este archivo.

```
from dash import html
from callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor import update_graph_gestor
from callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor import update_graph_gestor
from callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor import update_graph_gestor
from callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor import update_graph_gestor
from components.common.create_graph_with_table import create_graph_with_table
from util import config_mode_bar_buttons_gestor


def 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
    """

```

```
graphs_info = [
{
"graph_id": "nuevo-ingreso-genero-gestor",
            "modal_id": "modal-nuevo-ingreso-genero",
            "table_container_id": "table-container-nuevo-ingreso-genero",
            "download_button_id": "btn-descargar-nuevo-ingreso-genero",
            "view_data_button_id": "btn-ver-datos-nuevo-ingreso-genero",
        },
        {
            "graph_id": "nuevo_ingreso_nacionalidad-gestor",
            "modal_id": "modal-nuevo-ingreso-nacionalidad",
            "table_container_id": "table-container-nuevo-ingreso-nacionalidad",
            "download_button_id": "btn-descargar-nuevo-ingreso-nacionalidad",
            "view_data_button_id": "btn-ver-datos-nuevo-ingreso-nacionalidad",
        },
        {
            "graph_id": "egresados-genero-gestor",
            "modal_id": "modal-egresados-genero",
            "table_container_id": "table-container-egresados-genero",
            "download_button_id": "btn-descargar-egresados-genero",
            "view_data_button_id": "btn-ver-datos-egresados-genero",
        },
        {
            "graph_id": "egresados-nacionalidad-gestor",
            "modal_id": "modal-egresados-nacionalidad",
            "table_container_id": "table-container-egresados-nacionalidad",
            "download_button_id": "btn-descargar-egresados-nacionalidad",
            "view_data_button_id": "btn-ver-datos-egresados-nacionalidad",
        },
    ]

    graph_elements = [
        create_graph_with_table(
            graph_info["graph_id"],
            "graph-item-indicadores-gestor",
            config_mode_bar_buttons_gestor,
            {
                "modal_id": graph_info["modal_id"],
                "table_container_id": graph_info["table_container_id"],
                "download_button_id": graph_info["download_button_id"],
                "view_data_button_id": graph_info["view_data_button_id"],
            },
        )
        for graph_info in graphs_info
    ]

    return html.Div(graph_elements, className="graphs-container-indicadores-gestor")
```

6.227. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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.graphs_resultados_gestor ()

6.228. graphs_resultados_gestor.py

Ir a la documentación de este archivo.

```
from dash import html
from callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor import update_grpht_gestor
from callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor import update_graph_gestor
from components.common.create_graph_with_table import create_graph_with_table
from util import config_mode_bar_buttons_gestor


def 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
    """
graphs_info = [
{
"graph_id": "duración-estudios-nota-media-gestor",
            "modal_id": "modal-duracion-estudios",
            "table_container_id": "table-container-duracion-estudios",
            "download_button_id": "btn-descargar-csv-duracion-estudios",
            "view_data_button_id": "btn-ver-datos-duracion-estudios",
        },
        {
            "graph_id": "nota-acceso-titulacion",
            "modal_id": "modal-nota-acceso",
            "table_container_id": "table-container-nota-acceso",
            "download_button_id": "btn-descargar-csv-nota-acceso",
            "view_data_button_id": "btn-ver-datos-nota-acceso",
        },
    ]

    graph_elements = [
        create_graph_with_table(
            graph_info["graph_id"],
            "graph-item-resultados-gestor",
            config_mode_bar_buttons_gestor,
            {
                "modal_id": graph_info["modal_id"],
                "table_container_id": graph_info["table_container_id"],
                "download_button_id": graph_info["download_button_id"],
                "view_data_button_id": graph_info["view_data_button_id"],
            },
        )
        for graph_info in graphs_info
    ]

    return html.Div(graph_elements, className="graphs-container-resultados-gestor")
```

6.229. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html
from callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor import update_graph_gestor
from callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor import update_graph_gestor
from components.common.create_graph_with_table import create_graph_with_table
from util import config_mode_bar_buttons_gestor


def 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
    """
graphs_info = [
{
"graph_id": "tasa-abandono-gestor",
            "modal_id": "modal-tasa-abandono",
            "table_container_id": "table-container-tasa-abandono",
            "download_button_id": "btn-descargar-tasa-abandono",
            "view_data_button_id": "btn-ver-datos-tasa-abandono",
        },
        {
            "graph_id": "tasa-graduacion-gestor",
            "modal_id": "modal-tasa-graduacion",
            "table_container_id": "table-container-tasa-graduacion",
            "download_button_id": "btn-descargar-tasa-graduacion",
            "view_data_button_id": "btn-ver-datos-tasa-graduacion",
        },
    ]

    graph_elements = [
        create_graph_with_table(
            graph_info["graph_id"],
            "graph-item-riesgo-abandono-gestor",
            config_mode_bar_buttons_gestor,
            {
                "modal_id": graph_info["modal_id"],
                "table_container_id": graph_info["table_container_id"],
                "download_button_id": graph_info["download_button_id"],
                "view_data_button_id": graph_info["view_data_button_id"],
            },
        )
        for graph_info in graphs_info
    ]

    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.

```
from dash import html

def 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
    """
return html.Div([
        html.H1("Deserción universitaria", className='titulo-recomendador-gestor'),
        html.Img(src='assets/images/abandono-gestores.jpg', className='imagen-recomendador-gestor'),
        html.P("La deserción universitaria es un desafío significativo para las instituciones de educación superior. Los gestores tienen la responsabilidad de desarrollar e implementar políticas efectivas para reducirla.", className='p-recomendador-gestor'),
        html.H2("¿Qué es la deserción universitaria?", className='sb-recomendador-gestor'),
        html.P("La deserción universitaria se refiere al abandono de los estudios por parte de los estudiantes antes de completar su formación. Este fenómeno tiene implicaciones negativas para la universidad y para la sociedad.", className='p-recomendador-gestor'),
        html.Br(),
        html.H2("Estrategias para prevenir la deserción universitaria", className='sb-recomendador-gestor'),
        html.Ul([
            html.Li("Desarrollar programas de becas y ayudas financieras para apoyar a los estudiantes con dificultades económicas.", className='li-recomendador-gestor'),
            html.Li("Implementar sistemas de tutoría y mentoría para ofrecer apoyo académico y emocional a los estudiantes.", className='li-recomendador-gestor'),
            html.Li("Establecer programas de orientación y adaptación para nuevos estudiantes, facilitando su integración en la vida universitaria.", className='li-recomendador-gestor'),
            html.Li("Promover la formación continua del personal docente en metodologías pedagógicas innovadoras.", className='li-recomendador-gestor'),
            html.Li("Utilizar sistemas de seguimiento y análisis de datos para identificar a estudiantes en riesgo y actuar de manera preventiva.", className='li-recomendador-gestor'),
            html.Li("Fomentar un clima institucional inclusivo y de apoyo que motive a los estudiantes a continuar con sus estudios.", className='li-recomendador-gestor'),
        ], className='ul-recomendador-gestor'),
    ])
```

6.233. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html
from callbacks.gestor.utils.callback_resumen_gestor import update_resumen_gestor

```

```

def resumen_gestor():
    """
Crea un contenedor para mostrar un resumen de los datos del gestor seleccionado.

Returns:
html.Div: Contenedor para mostrar el resumen
"""

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.

```
from dash import html, dcc
from callbacks.gestor.utils.callback_select_gestor import store_selected_gestor


def select_gestor():
    """
Crea un dropdown para seleccionar un gestor.

Returns:
html.Div: Dropdown para seleccionar un gestor
"""

return html.Div(
        [
            html.Div(
                [
                    dcc.Dropdown(
                        options=[],
                        value=None,
                        id="gestor-dropdown",
                        clearable=False,
                        placeholder="Seleccione un gestor",
                    )
                ],
                className="select-gestor",
            ),
        ]
    )
```

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 from dash import html, dcc
00002 from callbacks.gestor.utils.callback_tabs_gestor import render_content
00003
00004
00005 def tabs_gestor():
00006     """
00007 Crea las pestañas de la sección "Gestor". Cada pestaña contiene un conjunto de gráficos
00008     y tablas con los datos de los alumnos del perfil "Gestor".
00009
00010     Returns:
00011     html.Div: Pestañas de la sección "Gestor"
00012     """
00013
00014 return html.Div(
00015         [
00016             dcc.Tabs(
00017                 id="tabs-gestor",
00018                 value="indicadores-academicos-tab",
00019                 children=[
00020                     dcc.Tab(
00021                         label="Indicadores académicos",
00022                         value="indicadores-academicos-tab",
00023                     ),
00024                     dcc.Tab(
00025                         label="Resultados académicos", value="resultados-academicos-tab"
00026                     ),
00027                     dcc.Tab(label="Riesgo de abandono", value="riesgo-abandono-tab"),
00028                     dcc.Tab(label="Recomendaciones", value="recomendaciones-tab"),
00029                 ],
00030                 className="tabs",
00031             ),
00032             html.Div(id="tabs-gestor-content"),
00033             dcc.Store(id="selected-gestor-store", storage_type="local"),
00034             dcc.Location(id="url", refresh=False),
00035         ]
00036     )
```

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 from sqlalchemy import create_engine, text
00002 from sqlalchemy.pool import QueuePool
00003 import json
00004
00005
00006 class DatabaseConnector:
00007     """
00008 Clase para conectarse a la base de datos
00009
00010 :param dbname: Nombre de la base de datos
00011 :param user: Usuario de la base de datos
00012 :param password: Contraseña del usuario
00013 :param host: Host de la base de datos
00014 :param port: Puerto de la base de datos
00015 :param pool_size: Tamaño del pool de conexiones
00016 :param max_overflow: Número máximo de conexiones que pueden ser creadas
00017 """
00018 def __init__(
00019         self, dbname, user, password, host, port, pool_size=5, max_overflow=10
00020     ):
00021         self.db_url = f"postgresql+psycopg2://{user}:{password}@{host}:{port}/{dbname}"
00022         self.engine = create_engine(
00023             self.db_url,
00024             poolclass=QueuePool,
00025             pool_size=pool_size,
00026             max_overflow=max_overflow,
00027         )
00028
00029     def execute_query(self, query, params=None):
00030         """
00031 Ejecuta una consulta en la base de datos
00032
00033 :param query: Consulta a ejecutar
00034 :param params: Parámetros de la consulta
00035 :return: Resultado de la consulta
00036         """
00037 with self.engine.connect() as connection:
00038             result = connection.execute(text(query), params or {})
00039             return result.fetchall()
00040
00041     def close(self):
00042         """
00043 Cierra la conexión a la base de datos
00044 """
00045 self.engine.dispose()
00046
00047
00048 with open("src/data/db_properties.json") as f:
00049     config = json.load(f)
00050
00051 # Se crea la instancia del conector
00052 db = DatabaseConnector(
00053     dbname=config["dbname"],
00054     user=config["user"],
00055     password=config["password"],
00056     host=config["host"],
00057     port=config["port"],
00058 )
```

6.241. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 = 5
- int data.generate_synthetic_data.num_docentes = 1
- int data.generate_synthetic_data.num_gestores = 5
- 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.

```
import pandas as pd
import numpy as np
from faker import Faker
import random
from tqdm import tqdm
import os

# Inicialización de Faker
fake = Faker('es_ES')

# Diccionario con ramas, titulaciones y palabras clave
educacion = {
    'Ingeniería y Tecnología': {
        'Ingeniería Informática': [
            'Programación', 'Algoritmos', 'Estructuras de Datos', 'Bases de Datos', 'Sistemas Operativos',
            'Redes', 'Inteligencia Artificial', 'Seguridad Informática', 'Desarrollo Web', 'Software',
            'Ciberseguridad', 'Ingeniería del Software', 'Bases de Datos Avanzadas', 'Arquitectura de Computadoras',
            'Gráficos por Computadora', 'Desarrollo de Aplicaciones', 'Computación en la Nube', 'Blockchain',
            'Big Data', 'Analítica de Datos', 'Aprendizaje Automático', 'Computación Paralela', 'Ciencias de la Computación',
            'Tecnologías de la Información', 'Interacción Humano-Computadora', 'Teoría de la Computación', 'Robótica',
            'Computación Cuántica', 'Sistemas Embebidos', 'Desarrollo de Videojuegos', 'Visión por Computador',
            'Minería de Datos', 'Internet de las Cosas', 'Bioinformática', 'Sistemas Distribuidos', 'Cómputo de Alto Rendimiento',
            'Cálculo Numérico', 'Realidad Virtual', 'Realidad Aumentada'
        ]
    },
    'Ciencias de la Salud': {
        'Medicina': [
            'Anatomía', 'Fisiología', 'Bioquímica', 'Farmacología', 'Patología',
            'Genética', 'Neurología', 'Cardiología', 'Cirugía', 'Dermatología',
            'Endocrinología', 'Epidemiología', 'Gastroenterología', 'Ginecología',
            'Hematología', 'Infectología', 'Medicina Interna', 'Nefrología',
            'Neumología', 'Oncología', 'Pediatría', 'Psiquiatría', 'Radiología',
            'Reumatología', 'Urología', 'Medicina Familiar', 'Medicina Preventiva',
            'Emergencias Médicas', 'Cuidados Intensivos', 'Medicina Deportiva', 'Medicina Nuclear',
            'Oftalmología', 'Otorrinolaringología', 'Rehabilitación', 'Medicina del Trabajo',
            'Medicina Forense', 'Geriatría', 'Inmunología', 'Toxicología', 'Anestesiología',
            'Nutriología', 'Microbiología', 'Parasitología', 'Patología Clínica', 'Genómica',
            'Neurocirugía'
        ]
    },
    'Ciencias Sociales y Jurídicas': {
        'Derecho': [
            'Derecho Civil', 'Derecho Penal', 'Derecho Constitucional', 'Derecho Administrativo', 'Derecho Laboral',
            'Derecho Mercantil', 'Derecho Internacional', 'Derecho Tributario', 'Derecho Ambiental', 'Derecho de Familia',
            'Derecho Procesal', 'Derecho Romano', 'Derecho Comparado', 'Derecho Financiero', 'Derecho Agrario',
            'Derecho Informático', 'Derecho de la Competencia', 'Derecho de la Propiedad Intelectual', 'Derecho Bancario',
            'Derecho Marítimo', 'Derecho Aéreo', 'Derecho del Consumo', 'Derecho de la Seguridad Social', 'Derecho Minero',
            'Derecho de la Energía', 'Derecho Urbanístico', 'Derecho de la Salud', 'Derecho del Deporte', 'Derecho de las Telecomunicaciones',
            'Derecho Internacional Privado', 'Derecho Internacional Público', 'Derecho de la Unión Europea', 'Derecho Canónico',
            'Derecho Militar', 'Derecho Electoral', 'Derecho Humanitario', 'Derecho Notarial', 'Derecho Penitenciario',
            'Derecho Registral'
        ]
    },
    'Humanidades': {
        'Historia': [
            'Historia Antigua', 'Historia Medieval', 'Historia Moderna', 'Historia Contemporánea', 'Historia de América',
            'Historia de Europa', 'Historia de Asia', 'Historia de África', 'Historia de Oceanía', 'Historia del Arte',
            'Historia Económica', 'Historia Política', 'Historia Social', 'Historiografía', 'Historia de la Ciencia',
            'Arqueología', 'Historia Militar', 'Historia de las Religiones', 'Historia Cultural', 'Historia de las Ideas',
            'Teoría de la Historia', 'Métodos de Investigación Histórica', 'Historia de la Educación', 'Historia de la Mujer',
            'Historia Ambiental', 'Historia Oral', 'Historia de la Tecnología', 'Historia Urbana', 'Historia de la Salud'
```

```
        ]
    }
}

# Diccionario de universidades con su código correspondiente
universidades_dict = {
    'Universidad de Madrid': 'UDM01',
    'Universidad de Barcelona': 'UDB02',
    'Universidad de La Laguna': 'ULL03',
}

# Generar asignaturas para cada titulación
def generate_asignaturas(educacion, min_asignaturas=38):
    asignaturas_dict = {}
    for rama, titulaciones in educacion.items():
        for titulacion, palabras in titulaciones.items():
            num_asignaturas = min(min_asignaturas, len(palabras))
            asignaturas = random.sample(palabras, num_asignaturas)
            asignaturas_dict[titulacion] = asignaturas
    return asignaturas_dict

# Generar diccionario de asignaturas
asignaturas_dict = generate_asignaturas(educacion)

# Función para generar códigos únicos de asignaturas
def generate_unique_cod_asignaturas(asignaturas_dict):
    cod_asignaturas = {}
    for titulacion, asignaturas in asignaturas_dict.items():
        for asignatura in asignaturas:
            cod_asignatura = asignatura[:4].upper() + fake.bothify(text='#####')
            cod_asignaturas[asignatura] = cod_asignatura
    return cod_asignaturas

# Generar un diccionario con códigos únicos para cada asignatura
cod_asignaturas_dict = generate_unique_cod_asignaturas(asignaturas_dict)
cod_plan_dict = {titulacion: fake.bothify(text='PLAN###??') for titulacion in asignaturas_dict.keys()}

# Función para generar datos sintéticos para la tabla 'alumnos'
def generate_alumnos(n):
    ids = [fake.unique.bothify(text='???#####') for _ in range(n)]
    anios_nac = np.random.randint(1980, 2000, n)
    universidades = random.choices(list(universidades_dict.keys()), k=n)
    cod_universidades = [universidades_dict[uni] for uni in universidades]
    abandona = random.choices(['si', 'no'], k=n)
    data = {
        'id': ids,
        'anio_nac': anios_nac,
        'universidad': universidades,
        'cod_universidad': cod_universidades,
        'abandona': abandona
    }
    return pd.DataFrame(data)

# Función para generar cursos académicos en formato "YYYY/YYYY"
def generate_curso_aca(start_year, end_year):
    return [f"{year}/{year+1}" for year in range(start_year, end_year)]

# Generar una lista de cursos académicos
cursos_academicos = generate_curso_aca(2017, 2024)

# Función para generar datos sintéticos para la tabla 'matricula'
def generate_matricula(alumnos_df):
    n = len(alumnos_df)
    num_matriculas = np.random.randint(1, 6, n)
    matricula_data = []
    for i, row in tqdm(alumnos_df.iterrows(), total=n, desc="matrícula"):
        titulacion = random.choice(list(asignaturas_dict.keys()))
        # Encontrar la rama correspondiente a la titulación
        rama = next(rama for rama, titulaciones in educacion.items() if titulacion in titulaciones)

        common_data = {
            'municipio': fake.city(),
            'nacionalidad': fake.country(),
            'provincia': fake.state(),
            'rama': rama,
            'sexo': random.choice(['Masculino', 'Femenino']),
            'titulacion': titulacion,
            'nombre_plan_propio': fake.word(),
            'universidad': row['universidad'],
            'cod_universidad': row['cod_universidad'],
            'cod_tipo_matricula': fake.bothify(text='T##'),
            'tipo_matricula': fake.word(),
            'cod_plan': cod_plan_dict[titulacion],
        }
        cursos_aca_usados = set()
        for j in range(num_matriculas[i]):
            curso_aca = random.choice([ca for ca in cursos_academicos if ca not in cursos_aca_usados])
```

```
            cursos_aca_usados.add(curso_aca)
            nuevo_ingreso = 'si' if j == 0 else 'no'
            matricula_data.append({
                'indice': len(matricula_data) + 1,
                'ambito_isced': fake.bothify(text='IS##'),
                'cod_plan': common_data['cod_plan'],
                'cod_mec': fake.bothify(text='MEC##'),
                'curso_aca': curso_aca,
                'municipio': common_data['municipio'],
                'nacionalidad': common_data['nacionalidad'],
                'nuevo_ingreso': nuevo_ingreso,
                'provincia': common_data['provincia'],
                'rama': common_data['rama'],
                'sexo': common_data['sexo'],
                'titulacion': common_data['titulacion'],
                'nombre_plan_propio': common_data['nombre_plan_propio'],
                'universidad': common_data['universidad'],
                'cod_universidad': common_data['cod_universidad'],
                'cod_tipo_matricula': common_data['cod_tipo_matricula'],
                'tipo_matricula': common_data['tipo_matricula'],
                'id': row['id']
            })
    return pd.DataFrame(matricula_data)

# Función para generar datos sintéticos para la tabla 'asignaturas_matriculadas'
def generate_asignaturas_matriculadas(matricula_df, cod_asignaturas):
    asignaturas_data = []
    for i, row in tqdm(matricula_df.iterrows(), total=len(matricula_df), desc="asignaturas_matriculadas"):
        num_asignaturas = min(random.randint(1, 20), len(asignaturas_dict[row['titulacion']]))
        asignaturas = random.sample(asignaturas_dict[row['titulacion']], num_asignaturas)
        for asignatura in asignaturas:
            cod_asignatura = cod_asignaturas[asignatura]
            asignaturas_data.append({
                'indice': len(asignaturas_data) + 1,
                'asignatura': asignatura,
                'cod_asignatura': cod_asignatura,
                'cod_plan': row['cod_plan'],
                'curso_aca': row['curso_aca'],
                'id': row['id'],
                'cod_tipologia': fake.bothify(text='T##'),
                'plan': row['titulacion'],
                'tipologia': fake.word(),
                'universidad': row['universidad'],
                'cod_universidad': row['cod_universidad']
            })
    return pd.DataFrame(asignaturas_data)

# Función para generar datos sintéticos para la tabla 'lineas_actas'
def generate_lineas_actas(asignaturas_matriculadas_df):
    actas_data = []
    for i, row in tqdm(asignaturas_matriculadas_df.iterrows(), total=len(asignaturas_matriculadas_df), desc="líneas_actas"):
        calif_numerica = round(random.uniform(0, 10), 2)
        if calif_numerica < 5:
            calif = 'Suspenso'
        elif calif_numerica < 7:
            calif = 'Aprobado'
        elif calif_numerica < 9:
            calif = 'Notable'
        else:
            calif = 'Sobresaliente'

        actas_data.append({
            'indice': len(actas_data) + 1,
            'asignatura': row['asignatura'],
            'calif_numerica': calif_numerica,
            'calif': calif,
            'cod_asig': row['cod_asignatura'],
            'cod_plan': row['cod_plan'],
            'conv': fake.bothify(text='C##'),
            'curso_aca': row['curso_aca'],
            'id': row['id'],
            'plan': row['plan'],
            'universidad': row['universidad'],
            'cod_universidad': row['cod_universidad']
        })
    return pd.DataFrame(actas_data)

# Función para generar datos sintéticos para la tabla 'docentes'
def generate_docentes(n, universidades, cod_asignaturas, asignaturas_dict):
    docentes_data = []

    for i in tqdm(range(n), desc="docentes"):
        id_docente = fake.bothify(text='D#####')
        universidad, cod_universidad = random.choice(list(universidades.items()))

```

```
        num_asignaturas = random.randint(1, 5)
        asignaturas_seleccionadas = random.sample(list(cod_asignaturas.items()), num_asignaturas)

        for asignatura, cod_asignatura in asignaturas_seleccionadas:
            # Encontrar la titulación correspondiente a la asignatura
            titulacion = next(titulacion for titulacion, asignaturas in asignaturas_dict.items() if asignatura in asignaturas)
            cod_plan = cod_plan_dict[titulacion]
            docentes_data.append({
                'indice': len(docentes_data) + 1,
                'id_docente': id_docente,
                'cod_universidad': cod_universidad,
                'universidad': universidad,
                'titulacion': titulacion,
                'cod_plan': cod_plan,
                'cod_asignatura': cod_asignatura,
                'asignatura': asignatura,
                'curso_aca': random.choice(cursos_academicos)
            })

    return pd.DataFrame(docentes_data)

# Función para generar datos sintéticos para la tabla 'ebau_prueba'
def generate_ebau_prueba(n, alumnos_ids, universidades):
    assert n == len(alumnos_ids), "El número de registros debe ser igual al número de alumnos"

    # Generar datos para cada campo
    data = {
        'indice': range(1, n + 1),
        'conv': [fake.bothify(text='C##') for _ in range(n)],
        'curso_aca': np.random.choice(cursos_academicos, n),
        'especialidad': [fake.word() for _ in range(n)],
        'id': alumnos_ids,
        'nota_bach': np.round(np.random.uniform(5, 10, n), 2),
        'nota_def': np.round(np.random.uniform(5, 10, n), 2),
        'nota_prue': np.round(np.random.uniform(5, 10, n), 2),
        'universidad': np.random.choice(list(universidades.keys()), n),
    }
    data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]

    return pd.DataFrame(data)

# Función para generar datos sintéticos para la tabla 'egresados'
def generate_egresados(alumnos_df, matricula_df, lineas_actas_df):
    aprobados = lineas_actas_df[lineas_actas_df['calif_numerica'] >= 5].groupby('id').size()
    egresados_ids = aprobados[aprobados >= 38].index.tolist()
    alumnos_no_abandona = alumnos_df[alumnos_df['abandona'] == 'no']['id'].tolist()
    egresados_ids = [id for id in egresados_ids if id in alumnos_no_abandona]

    egresados_data = []
    for i, id_alumno in tqdm(enumerate(egresados_ids), total=len(egresados_ids), desc="egresados"):
        universidad, cod_universidad = alumnos_df.loc[alumnos_df['id'] == id_alumno, ['universidad', 'cod_universidad']].values[0]
        cod_plan = matricula_df.loc[matricula_df['id'] == id_alumno, 'cod_plan'].values[0]
        ultimo_curso_aca = matricula_df.loc[matricula_df['id'] == id_alumno, 'curso_aca'].max()
        egresados_data.append({
            'indice': i + 1,
            'cod_plan': cod_plan,
            'curso_aca': ultimo_curso_aca,
            'id': id_alumno,
            'nota_media': round(random.uniform(5, 10), 2),
            'plan': fake.word(),
            'universidad': universidad,
            'cod_universidad': cod_universidad
        })
    return pd.DataFrame(egresados_data)

# Función para generar datos sintéticos para la tabla 'gestores'
def generate_gestores(n, universidades):
    data = {
        'gestor_id': range(1, n + 1),
        'universidad': np.random.choice(list(universidades.keys()), n)
    }
    data['cod_universidad'] = [universidades[uni] for uni in data['universidad']]
    return pd.DataFrame(data)


print("Generando datos....")
num_alumnos = 5
num_docentes = 1
num_gestores = 5

#crea un directorio llamado data/csv si no existe

if not os.path.exists('src/data/csv'):
    os.makedirs('src/data/csv')

```

```
alumnos_df = generate_alumnos(num_alumnos)
matricula_df = generate_matricula(alumnos_df)
asignaturas_matriculadas_df = generate_asignaturas_matriculadas(matricula_df, cod_asignaturas_dict)
lineas_actas_df = generate_lineas_actas(asignaturas_matriculadas_df)
docentes_df = generate_docentes(num_docentes, universidades_dict, cod_asignaturas_dict,
    asignaturas_dict)
ebau_prueba_df = generate_ebau_prueba(num_alumnos, alumnos_df['id'].tolist(), universidades_dict)
egresados_df = generate_egresados(alumnos_df, matricula_df, lineas_actas_df)
gestores_df = generate_gestores(num_gestores, universidades_dict)

# Guardar datos en archivos CSV
alumnos_df.to_csv('src/data/csv/alumnos.csv', index=False)
matricula_df.to_csv('src/data/csv/matricula.csv', index=False)
asignaturas_matriculadas_df.to_csv('src/data/csv/asignaturas_matriculadas.csv', index=False)
lineas_actas_df.to_csv('src/data/csv/lineas_actas.csv', index=False)
docentes_df.to_csv('src/data/csv/docentes.csv', index=False)
ebau_prueba_df.to_csv('src/data/csv/ebau_prueba.csv', index=False)
egresados_df.to_csv('src/data/csv/egresados.csv', index=False)
gestores_df.to_csv('src/data/csv/gestores.csv', index=False)

print("Datos generados correctamente.")
```

6.243. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 from data.db_connector import db
00002 import pandas as pd
00003 from data.queries_dictionary import queries
00004 from functools import lru_cache
00005
00006
00007 # Función para comprobar si la query se ha ejecutado correctamente
00008 def check_data(query, params):
00009     try:
00010         data = db.execute_query(query, params)
00011     except Exception as e:
00012         print("Query execution failed: ", e)
00013         return []
00014     return data
00015
00016
00017 # Decorador para cachear las queries y evitar hacer consultas innecesarias
00018 def cache_query(func):
00019     @lru_cache(maxsize=32)
00020     def wrapper(*args, **kwargs):
00021         return func(*args, **kwargs)
00022
00023     return wrapper
00024
00025
00026 @cache_query
00027 def alumnos_all():
00028     query = queries["alumnado"]["common"]["alumnos_all"]
00029     return check_data(query, {})
00030
00031
00032 @cache_query
00033 def resumen_alumno(alumno_id, titulacion):
00034     query = queries["alumnado"]["common"]["resumen_alumno"]
00035     params = {"alumno_id": alumno_id, "titulacion": titulacion}
```

```python

    return check_data(query, params)


@cache_query
def resumen_gestor(gestor_id):
    query = queries["gestor"]["common"]["resumen_gestor"]
    params = {"gestor_id": gestor_id}

    return check_data(query, params)


@cache_query
def nota_media_alumno_titulacion(alumno_id, titulacion):
    query = queries["alumnado"]["common"]["nota_media_alumno_titulacion"]
    params = {"alumno_id": alumno_id, "titulacion": titulacion}

    data = check_data(query, params)

    if not data:
        return "No disponible"

    return round(data[0][0], 2)


@cache_query
def curso_academico_alumnado(alumno_id, titulacion):
    query = queries["alumnado"]["filters"]["curso_academico_alumnado"]
    params = {"alumno_id": alumno_id, "titulacion": titulacion}

    return check_data(query, params)


@cache_query
def asignaturas_matriculadas(alumno_id, curso_academico, titulacion):
    query = queries["alumnado"]["filters"]["asinaturas_matriculadas"]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@cache_query
def titulacion_alumnado(alumno_id):
    query = queries["alumnado"]["filters"]["titulacion_alumnado"]
    params = {"alumno_id": alumno_id}

    return check_data(query, params)


@cache_query
def asignaturas_superadas(alumno_id, curso_academico, titulacion):
    query = queries["alumnado"]["graphs"]["personal"][
        "curso_academico_asignaturas_superadas"
    ]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@cache_query
def calif_cualitativa_asignatura(alumno_id, curso_academico, titulacion):
    query = queries["alumnado"]["graphs"]["personal"]["calif_cualitativa_asginatura"]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@cache_query
def calif_numerica_asignatura(alumno_id, curso_academico, titulacion):
    query = queries["alumnado"]["graphs"]["personal"]["calif_numerica_asignatura"]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "titulacion": titulacion,
    }
```

```
    return check_data(query, params)


@cache_query
def asignaturas_matriculadas_y_superadas(alumno_id, curso_academico, titulacion):
    query = queries["alumnado"]["graphs"]["personal"][
        "asignaturas_matriculadas_y_superadas"
    ]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@cache_query
def asignaturas_superadas_media_abandono(
    curso_academico, asignaturas_matriculadas, titulacion, cod_universidad
):
    query = queries["alumnado"]["graphs"]["general"][
        "asignaturas_superadas_media_abandono"
    ]
    params = {
        "curso_academico": curso_academico,
        "asignaturas_matriculadas": asignaturas_matriculadas,
        "titulacion": titulacion,
        "cod_universidad": cod_universidad,
    }

    return check_data(query, params)


@cache_query
def calif_cualitativa_comparativa(
    curso_academico, asignaturas_matriculadas, titulacion, cod_universidad
):
    query = queries["alumnado"]["graphs"]["general"]["calif_cualitativa_comparativa"]
    params = {
        "curso_academico": curso_academico,
        "asignaturas_matriculadas": asignaturas_matriculadas,
        "titulacion": titulacion,
        "cod_universidad": cod_universidad,
    }

    return check_data(query, params)


@cache_query
def calif_cualitativa_alumno_asignaturas(
    alumno_id, curso_academico, asignaturas_matriculadas, titulacion
):
    query = queries["alumnado"]["graphs"]["general"][
        "calif_cualitativa_alumno_asignaturas"
    ]
    params = {
        "alumno_id": alumno_id,
        "curso_academico": curso_academico,
        "asignaturas_matriculadas": asignaturas_matriculadas,
        "titulacion": titulacion,
    }

    return check_data(query, params)


@cache_query
def nota_media_general_mi_nota(
    curso_academico, asignaturas_matriculadas, alumno_id, titulacion, cod_universidad
):
    query = queries["alumnado"]["graphs"]["general"]["nota_media_general_mi_nota"]
    params = {
        "curso_academico": curso_academico,
        "asignaturas_matriculadas": asignaturas_matriculadas,
        "alumno_id": alumno_id,
        "titulacion": titulacion,
        "cod_universidad": cod_universidad,
    }

    return check_data(query, params)


@cache_query
def alumnos_repetidores_nuevos(docente_id, curso_academico, asignaturas):
    query = queries["docente"]["graphs"]["personal"]["alumnos_repetidores_nuevos"]
    params = {
```

```
        "docente_id": docente_id,
        "curso_academico": curso_academico,
        "asignaturas": asignaturas,
    }

    return check_data(query, params)


@cache_query
def asignaturas_docente(id_docente, titulacion):
    query = queries["docente"]["filters"]["asignaturas_docente"]
    params = {"id_docente": id_docente, "titulacion": titulacion}

    return check_data(query, params)


@cache_query
def curso_academico_docente(id_docente, asignatura):
    query = queries["docente"]["filters"]["curso_academico_docente"]
    params = {"id_docente": id_docente, "asignatura": asignatura}

    return check_data(query, params)


@cache_query
def titulacion_docente(id_docente):
    query = queries["docente"]["filters"]["titulacion_docente"]
    params = {"id_docente": id_docente}

    return check_data(query, params)


@cache_query
def resumen_docente(id_docente, titulacion):
    query = queries["docente"]["common"]["resumen_docente"]
    params = {"id_docente": id_docente, "titulacion": titulacion}

    return check_data(query, params)


@cache_query
def docentes_all():
    query = queries["docente"]["common"]["docentes_all"]
    return check_data(query, {})


@cache_query
def alumnos_genero_docente(id_docente, asignaturas, curso_academico):
    query = queries["docente"]["graphs"]["personal"]["alumnos_genero_docente"]
    params = {
        "id_docente": id_docente,
        "asignaturas": asignaturas,
        "curso_academico": curso_academico,
    }

    return check_data(query, params)


@cache_query
def alumnos_nota_media_docente(asignaturas, curso_academico):
    query = queries["docente"]["graphs"]["personal"]["alumnos_nota_media_docente"]
    params = {"asignaturas": asignaturas, "curso_academico": curso_academico}

    return check_data(query, params)


@cache_query
def alumnos_nota_cualitativa_docente(asignaturas, curso_academico):
    query = queries["docente"]["graphs"]["personal"]["alumnos_nota_cualitativa_docente"]
    params = {"asignaturas": asignaturas, "curso_academico": curso_academico}

    return check_data(query, params)


@cache_query
def curso_academico_actas_titulacion(titulacion):
    query = queries["docente"]["filters"]["curso_academico_actas_titulacion"]
    params = {"titulacion": titulacion}

    return check_data(query, params)


@cache_query
def asignaturas_actas_titulacion(titulacion, curso_academico):
    query = queries["docente"]["filters"]["asignaturas_actas_titulacion"]
    params = {"titulacion": titulacion, "curso_academico": curso_academico}

```

```
    return check_data(query, params)


@cache_query
def calif_all_cualitativa_asignaturas(titulacion, curso_academico, asignaturas):
    query = queries["docente"]["graphs"]["general"]["calif_all_cualitativa_asignaturas"]
    params = {
        "titulacion": titulacion,
        "curso_academico": curso_academico,
        "asignaturas": asignaturas,
    }

    return check_data(query, params)


@cache_query
def calif_media_asignaturas(titulacion, curso_academico, asignaturas):
    query = queries["docente"]["graphs"]["general"]["calif_media_asignaturas"]
    params = {
        "titulacion": titulacion,
        "curso_academico": curso_academico,
        "asignaturas": asignaturas,
    }

    return check_data(query, params)


@cache_query
def gestores_all():
    query = queries["gestor"]["common"]["gestores_all"]
    return check_data(query, {})


@cache_query
def numero_alumnos_matriculados_universidad(universidad):
    query = queries["gestor"]["common"]["numero_alumnos_matriculados_universidad"]
    params = {"universidad": universidad}

    return check_data(query, params)


@cache_query
def universidades_gestor(gestor_id):
    query = queries["gestor"]["common"]["universidades_gestor"]
    params = {"gestor_id": gestor_id}

    return check_data(query, params)


@cache_query
def curso_academico_universidad(cod_universidad):
    query = queries["gestor"]["filters"]["curso_academico_universidad"]
    params = {"cod_universidad": cod_universidad}

    return check_data(query, params)


@cache_query
def titulaciones_universidad_gestor(cod_universidad, curso_academico):
    query = queries["gestor"]["filters"]["titulaciones_universidad_gestor"]
    params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}

    return check_data(query, params)


@cache_query
def alumnos_nuevo_ingreso_genero_titulacion(
    curso_academico, titulaciones, cod_universidad
):
    query = queries["gestor"]["graphs"]["indicadores"][
        "alumnos_nuevo_ingreso_genero_titulacion"
    ]
    params = {
        "curso_academico": curso_academico,
        "titulaciones": titulaciones,
        "cod_universidad": cod_universidad,
    }

    return check_data(query, params)


@cache_query
def alumnos_egresados_genero_titulacion(cod_universidad, curso_academico, titulaciones):
    query = queries["gestor"]["graphs"]["indicadores"][
        "alumnos_egresados_genero_titulacion"
    ]
    params = {
```

```
        "cod_universidad": cod_universidad,
        "curso_academico": curso_academico,
        "titulaciones": titulaciones,
    }

    return check_data(query, params)


@cache_query
def alumnos_egresados_nacionalidad_titulacion(
    cod_universidad, curso_academico, titulaciones
):
    query = queries["gestor"]["graphs"]["indicadores"][
        "alumnos_egresados_nacionalidad_titulacion"
    ]
    params = {
        "cod_universidad": cod_universidad,
        "curso_academico": curso_academico,
        "titulaciones": titulaciones,
    }

    return check_data(query, params)


@cache_query
def alumnos_nuevo_ingreso_nacionalidad_titulacion(
    cod_universidad, curso_academico, titulaciones
):
    query = queries["gestor"]["graphs"]["indicadores"][
        "alumnos_nuevo_ingreso_nacionalidad_titulacion"
    ]
    params = {
        "cod_universidad": cod_universidad,
        "curso_academico": curso_academico,
        "titulaciones": titulaciones,
    }

    return check_data(query, params)


@cache_query
def nota_media_acceso_titulacion(cod_universidad):
    query = queries["gestor"]["graphs"]["resultados"]["nota_media_acceso_titulacion"]
    params = {"cod_universidad": cod_universidad}

    return check_data(query, params)


@cache_query
def duracion_media_estudios_nota_gestor(cod_universidad):
    query = queries["gestor"]["graphs"]["resultados"][
        "duracion_media_estudios_nota_gestor"
    ]
    params = {"cod_universidad": cod_universidad}

    return check_data(query, params)


@cache_query
def cursos_academicos_egresados(cod_universidad):
    query = queries["gestor"]["common"]["cursos_academicos_de_egresados"]
    params = {"cod_universidad": cod_universidad}

    return check_data(query, params)


@cache_query
def tasa_abandono_titulacion_gestor(cod_universidad, curso_academico):
    query = queries["gestor"]["graphs"]["riesgo_abandono"][
        "tasa_abandono_titulacion_gestor"
    ]
    params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}

    return check_data(query, params)


@cache_query
def tasa_graduacion_titulacion_gestor(cod_universidad, curso_academico):
    query = queries["gestor"]["graphs"]["riesgo_abandono"][
        "tasa_graduacion_titulacion_gestor"
    ]
    params = {"cod_universidad": cod_universidad, "curso_academico": curso_academico}

    return check_data(query, params)


@cache_query
```

```
def universidad_alumno(alumno_id):
    query = queries["alumnado"]["common"]["universidad_alumno"]
    params = {"alumno_id": alumno_id}

    return check_data(query, params)


@cache_query
def universidades_docente(id_docente):
    query = queries["docente"]["common"]["universidades_docente"]
    params = {"id_docente": id_docente}

    return check_data(query, params)

@cache_query
def data_for_model():
    query = queries["alumnado"]["common"]["data_for_model"]
    data = check_data(query, {})

    if not data:
        return []

    df = pd.DataFrame(data)

    # Ajustar los tipos de datos de las columnas
    df = df.astype({
        'id': str,
        'anio_nac': int,
        'nacionalidad': str,
        'sexo': str,
        'titulacion': str,
        'nota_def_ebau': float,
        'nota_media': float,
        'abandona': str
    })

    return df

@cache_query
def resumen_alumno(alumno_id, titulacion):
    query = queries["alumnado"]["common"]["resumen_alumno"]
    params = {"alumno_id": alumno_id, "titulacion": titulacion}

    return check_data(query, params)
```

6.245. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
# Este diccionario contiene las consultas SQL utilizadas en el proyecto
queries = {
    "alumnado": {
        "common": {
            # Consulta para obtener los alumnos.
            "alumnos_all": """
```

```
SELECT id FROM alumnos
ORDER BY id
LIMIT 50;
""",
"resumen_alumno": """
SELECT DISTINCT alu.id, ma.universidad, titulacion, abandona
FROM alumnos alu
JOIN matricula ma ON ma.id = alu.id
WHERE alu.id = :alumno_id AND ma.titulacion = :titulacion;
""",
# Consulta para calcular la media de las calificaciones de un alumno en una titulación específica.
            "nota_media_alumno_titulacion": """
SELECT AVG(li.calif_numerica) AS media_calif
FROM public.lineas_actas li
JOIN public.matricula m ON m.id = li.id AND m.cod_plan = li.cod_plan
WHERE li.id = :alumno_id
AND li.calif_numerica >= 5
AND m.titulacion = :titulacion;
""",
"universidad_alumno": """
SELECT cod_universidad, universidad
FROM matricula
WHERE id = :alumno_id;
""",
"data_for_model": """

WITH average_grades AS (
SELECT
ma.id,
ma.titulacion,
AVG(la.calif_numerica) AS nota_media
FROM
public.lineas_actas la
JOIN
matricula ma ON la.id = ma.id AND ma.cod_plan = la.cod_plan
WHERE
la.calif_numerica >= 5
GROUP BY
ma.id, ma.titulacion
)
SELECT DISTINCT
alu.id,
alu.anio_nac,
ma.nacionalidad,
ma.sexo,
ma.titulacion,
ebau.nota_def AS nota_def_ebau,
alu.abandona,
ag.nota_media
FROM
alumnos alu
JOIN
matricula ma ON alu.id = ma.id
JOIN
ebau_prueba ebau ON ebau.id = ma.id
LEFT JOIN
average_grades ag ON ma.id = ag.id AND ma.titulacion = ag.titulacion
ORDER BY
alu.id;
"""
},
"filters": {
            # Consulta para obtener los cursos académicos en los que un alumno está matriculado en una titulación específica.
            "curso_academico_alumnado": """
SELECT curso_aca
FROM matricula
WHERE id = :alumno_id AND titulacion = :titulacion;
""",
# Consulta para obtener las asignaturas matriculadas por un alumno en un curso académico y titulación específicos.
            "asinaturas_matriculadas": """
SELECT DISTINCT AM.asignatura
FROM asignaturas_matriculadas AM
JOIN matricula MA ON AM.id = MA.id AND AM.cod_plan = MA.cod_plan
WHERE AM.id = :alumno_id AND AM.curso_aca IN :curso_academico AND MA.titulacion = :titulacion;
""",
# Consulta para obtener las titulaciones de un alumno.
            "titulacion_alumnado": """
SELECT DISTINCT titulacion
FROM matricula
WHERE id = :alumno_id;
""",
},
"graphs": {
            "personal": {
                # Consulta para obtener las asignaturas superadas por un alumno en un curso académico
```

```
y titulación específicos.
                "curso_academico_asignaturas_superadas": """
SELECT li.curso_aca AS curso_academico, COUNT(DISTINCT li.asignatura) AS n_asig_superadas
FROM lineas_actas li
JOIN matricula ma ON li.id = ma.id AND li.cod_plan = ma.cod_plan
WHERE li.id = :alumno_id AND li.calif_numerica >= 5 AND li.curso_aca IN :curso_academico
AND ma.titulacion = :titulacion
GROUP BY li.curso_aca
ORDER BY li.curso_aca;
""",
# Consulta para obtener la calificación cualitativa más alta por curso académico y asignatura de un alumno.
                "calif_cualitativa_asginatura": """
WITH RankedGrades AS (
SELECT li.curso_aca, li.calif,
ROW_NUMBER() OVER (PARTITION BY li.curso_aca, li.asignatura ORDER BY
CASE
WHEN li.calif = 'Sobresaliente' THEN 5
                                    WHEN li.calif = 'Notable' THEN 4
                                    WHEN li.calif = 'Aprobado' THEN 3
                                    WHEN li.calif = 'Suspenso' THEN 2
                                    WHEN li.calif = 'No presentado' THEN 1
                                    ELSE 0
                                END DESC) AS rk
                        FROM lineas_actas li
                        JOIN matricula ma ON li.id = ma.id AND li.cod_plan = ma.cod_plan
                        WHERE li.id = :alumno_id
                        AND li.curso_aca IN :curso_academico
                        AND ma.titulacion = :titulacion
                      )
                      SELECT curso_aca AS curso_academico, calif AS calificaciones, COUNT(*) AS n_calificaciones
                      FROM RankedGrades
                      WHERE rk = 1
                      GROUP BY curso_aca, calif
                      ORDER BY curso_aca;
                      """,
# Consulta para obtener la calificación numérica de las asignaturas matriculadas por un alumno.
                "calif_numerica_asignatura": """
SELECT li.asignatura, MAX(li.calif_numerica) AS calif_numerica
FROM lineas_actas li
JOIN matricula ma ON li.id = ma.id AND li.cod_plan = ma.cod_plan
WHERE li.id = :alumno_id
AND li.curso_aca IN :curso_academico
AND ma.titulacion = :titulacion
GROUP BY li.asignatura
ORDER BY li.asignatura;
""",
# Consulta para obtener el número de asignaturas matriculadas y superadas por un alumno.
                "asignaturas_matriculadas_y_superadas": """
SELECT
AM.curso_aca AS "curso_academico",
                        COUNT(DISTINCT AM.asignatura) AS "Total asignaturas matriculadas",
                        COUNT(DISTINCT CASE WHEN LA.calif_numerica >= 5 THEN LA.asignatura ELSE NULL END) AS "Total asignaturas superadas"
                      FROM asignaturas_matriculadas AM
                      LEFT JOIN lineas_actas LA ON AM.cod_asignatura = LA.cod_asig AND AM.id = LA.id AND AM.curso_aca = LA.curso_aca
                      JOIN matricula MA ON AM.id = MA.id AND AM.cod_plan = MA.cod_plan
                      WHERE
                        AM.id = :alumno_id
                        AND AM.curso_aca IN :curso_academico
                        AND MA.titulacion = :titulacion
                      GROUP BY
                        AM.curso_aca;
                      """,
},
"general": {
                # Esta consulta devuelve el id de los alumnos, nota media, abandono y total de asignaturas superadas.
                "asignaturas_superadas_media_abandono": """
SELECT
a.id,
a.abandona,
AVG(NULLIF(la.max_calif, 0)) AS NotaMedia,
COUNT(la.asignatura) AS "Total asignaturas superadas"
                        FROM
                            (
                                SELECT
                                    la.id,
                                    la.asignatura,
                                    MAX(la.calif_numerica) AS max_calif
                                FROM
                                    public.lineas_actas la
                                JOIN
                                    public.matricula ma
                                ON
```

```
                                    la.id = ma.id AND la.cod_plan = ma.cod_plan
                                WHERE
                                    la.curso_aca IN :curso_academico
                                    AND la.asignatura IN :asignaturas_matriculadas
                                    AND ma.titulacion = :titulacion
                                    AND ma.cod_universidad = :cod_universidad
                                GROUP BY
                                    la.id, la.asignatura
                            ) la
                        JOIN
                            public.alumnos a
                        ON
                            a.id = la.id
                        WHERE
                            la.max_calif >= 5
                        GROUP BY
                            a.id, a.abandona
                        ORDER BY
                            a.id;
                        """,
# Consulta para obtener la calificación cualitativa de los alumnos para compararlo con los datos generales.
                "calif_cualitativa_comparativa": """
WITH CalificacionesMaximas AS (
SELECT
la.id,
la.asignatura,
la.curso_aca,
la.calif,
ROW_NUMBER() OVER (PARTITION BY la.id, la.curso_aca, la.asignatura ORDER BY CASE
WHEN la.calif = 'Sobresaliente' THEN 5
                                  WHEN la.calif = 'Notable' THEN 4
                                  WHEN la.calif = 'Aprobado' THEN 3
                                  WHEN la.calif = 'Suspenso' THEN 2
                                  WHEN la.calif = 'No presentado' THEN 1
                              ELSE 0 END DESC) AS rk
                              FROM lineas_actas la
                              JOIN
                                  matricula ma ON la.id = ma.id AND la.cod_plan = ma.cod_plan
                              WHERE
                                  la.curso_aca IN :curso_academico AND
                                  la.asignatura IN :asignaturas_matriculadas AND
                                  la.calif IN ('Sobresaliente', 'Notable', 'Aprobado', 'Suspenso', 'No presentado') AND
                                  ma.titulacion = :titulacion AND
                                  ma.cod_universidad = :cod_universidad
                              )
                          SELECT
                              c.asignatura,
                              c.calif,
                              COUNT(*) AS count_grades
                          FROM CalificacionesMaximas c
                          WHERE c.rk = 1
                          GROUP BY c.asignatura, c.calif;
                          """,
# Consulta para obtener la calificación cualitativa del alumno.
                "calif_cualitativa_alumno_asignaturas": """
WITH CalificacionesMaximas AS (
SELECT
la.id,
la.asignatura,
MAX(
CASE
WHEN la.calif = 'Sobresaliente' THEN 5
                                        WHEN la.calif = 'Notable' THEN 4
                                        WHEN la.calif = 'Aprobado' THEN 3
                                        WHEN la.calif = 'Suspenso' THEN 2
                                        WHEN la.calif = 'No presentado' THEN 1
                                        ELSE 0
                                    END
                                ) AS max_calif_value
                            FROM
                                public.lineas_actas la
                            JOIN
                                public.matricula ma ON la.id = ma.id AND la.cod_plan = ma.cod_plan
                            WHERE
                                la.curso_aca IN :curso_academico AND
                                la.asignatura IN :asignaturas_matriculadas AND
                                la.id = :alumno_id AND
                                la.calif IN ('Sobresaliente', 'Notable', 'Aprobado', 'Suspenso', 'No presentado') AND
                                ma.titulacion = :titulacion
                            GROUP BY
                                la.id, la.asignatura
                        )
                        SELECT
                            cm.asignatura,
```

```
                            CASE cm.max_calif_value
                                WHEN 5 THEN 'Sobresaliente'
                                WHEN 4 THEN 'Notable'
                                WHEN 3 THEN 'Aprobado'
                                WHEN 2 THEN 'Suspenso'
                                WHEN 1 THEN 'No presentado'
                                ELSE 'Desconocido'
                            END AS calif,
                            COUNT(*) AS count_grades
                        FROM
                            CalificacionesMaximas cm
                        GROUP BY
                            cm.asignatura, cm.max_calif_value
                        ORDER BY
                            cm.asignatura;
                    """,
# Consulta para obtener la calificación media de los alumnos y la calificación del alumno.
                "nota_media_general_mi_nota": """
SELECT
subquery.asignatura,
AVG(subquery.max_calif_numerica) AS media_calif,
MAX(CASE WHEN subquery.id = :alumno_id THEN subquery.max_calif_numerica ELSE NULL END) AS calif_alumno
FROM (
SELECT
l.asignatura,
l.id,
l.curso_aca,
MAX(l.calif_numerica) AS max_calif_numerica
FROM
lineas_actas l
JOIN
matricula m ON l.id = m.id AND l.cod_plan = m.cod_plan
WHERE
l.asignatura IN :asignaturas_matriculadas AND
l.curso_aca IN :curso_academico AND
m.titulacion = :titulacion AND
m.cod_universidad = :cod_universidad
GROUP BY
l.asignatura,
l.id,
l.curso_aca
) subquery
GROUP BY
subquery.asignatura
ORDER BY
subquery.asignatura;
""",
},
},
},
"docente": {
        "common": {
            # Consulta para obtener los docentes.
            "docentes_all": """
SELECT DISTINCT id_docente FROM docentes
LIMIT 20;
""",
# Consulta para obtener la universidad de un docente.
            "universidades_docente": """
SELECT DISTINCT cod_universidad, universidad
FROM docentes
WHERE id_docente = :id_docente;
""",
},
"filters": {
            # Consulta para obtener las asignaturas de un docente según la titulación.
            "asignaturas_docente": """
SELECT DISTINCT asignatura
FROM docentes
WHERE id_docente = :id_docente AND titulacion = :titulacion;
""",
# Consulta para obtener los cursos académicos de un docente según la asignatura.
            "curso_academico_docente": """
SELECT curso_aca
FROM docentes
WHERE id_docente = :id_docente AND asignatura = :asignatura;
""",
# Consulta para obtener la titulación de un docente.
            "titulacion_docente": """
SELECT DISTINCT titulacion
FROM docentes
WHERE id_docente = :id_docente;
""",
# Consulta para obtener todos los cursos acádemicos de las actas según la titulación
            "curso_academico_actas_titulacion": """
SELECT DISTINCT li.curso_aca
FROM lineas_actas li
```

```
JOIN docentes ON li.cod_plan = docentes.cod_plan AND
li.cod_asig = docentes.cod_asignatura AND
li.curso_aca = docentes.curso_aca
WHERE docentes.titulacion = :titulacion
ORDER BY curso_aca;
""",
# Consulta para obtener todas las asignaturas por curso académico y titulación
            "asignaturas_actas_titulacion": """
SELECT DISTINCT li.asignatura
FROM lineas_actas li
JOIN docentes ON li.cod_plan = docentes.cod_plan
WHERE docentes.titulacion = :titulacion AND
li.curso_aca = :curso_academico AND
li.cod_asig = docentes.cod_asignatura
ORDER BY li.asignatura;

""",
},
"graphs": {
            "personal": {
                # Consulta para obtener el número de alumnos repetidores y de nuevo ingreso en una asignatura.
                "alumnos_repetidores_nuevos": """
WITH repetidores AS (
SELECT
am.id AS student_id,
MIN(am.curso_aca) AS primer_curso_aca
FROM public.asignaturas_matriculadas am
WHERE am.asignatura = :asignaturas
GROUP BY am.id
HAVING COUNT(DISTINCT am.curso_aca) > 1
),
alumnos_categoria AS (
SELECT
am.id,
am.curso_aca,
CASE
WHEN r.student_id IS NOT NULL AND r.primer_curso_aca <> am.curso_aca THEN 'repetidor'
                                    ELSE 'nuevo_ingreso'
                                END AS categoria
                            FROM public.asignaturas_matriculadas am
                            LEFT JOIN repetidores r ON am.id = r.student_id
                            WHERE am.asignatura = :asignaturas
                            AND am.curso_aca IN :curso_academico
                            AND EXISTS (
                                SELECT 1
                                FROM public.docentes d
                                WHERE d.cod_asignatura = am.cod_asignatura
                                AND d.id_docente = :docente_id
                            )
                        )
                        SELECT
                            am.curso_aca AS curso_academico,
                            COUNT(DISTINCT am.id) FILTER (WHERE categoria = 'repetidor') AS alumnos_repetidores,
                            COUNT(DISTINCT am.id) FILTER (WHERE categoria = 'nuevo_ingreso') AS alumnos_nuevo_ingreso
                        FROM alumnos_categoria am
                        GROUP BY am.curso_aca
                        ORDER BY am.curso_aca;

                        """,
# Consulta para el número de alumnos por género en una asignatura.
                "alumnos_genero_docente": """
WITH docente_asignaturas AS (
SELECT cod_asignatura
FROM public.docentes
WHERE id_docente = :id_docente
AND asignatura = :asignaturas
),
asignaturas_matriculadas AS (
SELECT DISTINCT am.id, am.cod_asignatura, am.curso_aca
FROM public.asignaturas_matriculadas am
JOIN docente_asignaturas da ON am.cod_asignatura = da.cod_asignatura
WHERE am.curso_aca IN :curso_academico
AND am.asignatura = :asignaturas
),
estudiantes_sexo AS (
SELECT DISTINCT m.id AS alumno_id, m.sexo
FROM public.matricula m
JOIN asignaturas_matriculadas am ON m.id = am.id
)
SELECT am.curso_aca, m.sexo, COUNT(*) AS cantidad
FROM estudiantes_sexo m
JOIN asignaturas_matriculadas am ON m.alumno_id = am.id
GROUP BY am.curso_aca, m.sexo
ORDER BY am.curso_aca, m.sexo;
```

```
""",
# Consulta para obtener la nota media de los alumnos por asignatura y curso académico.
                "alumnos_nota_media_docente": """
SELECT
l.curso_aca,
l.asignatura,
AVG(l.calif_numerica) AS media_calif
FROM
lineas_actas l
WHERE
l.asignatura IN :asignaturas AND
l.curso_aca IN :curso_academico
GROUP BY
l.curso_aca,
l.asignatura
ORDER BY
l.curso_aca,
l.asignatura;
""",
# Consulta para obtener la nota cualitativa de los alumnos por asignatura y curso académico.
                "alumnos_nota_cualitativa_docente": """
SELECT DISTINCT
l.curso_aca,
l.calif,
COUNT(*) AS num_alumnos
FROM
lineas_actas l
WHERE
l.asignatura = :asignaturas AND
l.curso_aca IN :curso_academico
GROUP BY
l.curso_aca,
l.calif
ORDER BY
l.curso_aca,
l.calif;
"""
},
"general": {
                # Consulta para obtener el número de calificaciones cualitativas
                # de los alumnos por asignatura y curso académico.
                "calif_all_cualitativa_asignaturas": """
SELECT
li.curso_aca,
li.asignatura,
li.calif,
COUNT(DISTINCT li.id) AS n_alumnos
FROM lineas_actas li
JOIN docentes ON li.cod_plan = docentes.cod_plan AND li.cod_asig = docentes.cod_asignatura
WHERE
docentes.titulacion = :titulacion AND
li.curso_aca = :curso_academico AND
li.asignatura IN :asignaturas
GROUP BY li.asignatura, li.curso_aca, li.calif
ORDER BY li.asignatura, li.curso_aca, li.calif;

""",
"calif_media_asignaturas": """
SELECT
subquery.asignatura,
AVG(subquery.calif_numerica) AS media_calif
FROM (
SELECT
l.asignatura,
l.id,
l.curso_aca,
l.calif_numerica
FROM
lineas_actas l
JOIN
matricula m ON l.id = m.id AND l.cod_plan = m.cod_plan
WHERE
l.asignatura IN :asignaturas AND
l.curso_aca = :curso_academico AND
m.titulacion = :titulacion
GROUP BY
l.asignatura,
l.id,
l.curso_aca,
l.calif_numerica
) subquery
GROUP BY
subquery.asignatura
ORDER BY
subquery.asignatura;
"""
},
```

```
},
},
"gestor": {
        "common": {
            # Consulta para obtener los gestores.
            "gestores_all": """
SELECT DISTINCT gestor_id FROM gestores
LIMIT 20;
""",
# Consulta para obtener el número de alumnos matriculados en una universidad.
            "numero_alumnos_matriculados_universidad": """
SELECT COUNT(alumnos.id) AS n_alumnos_universidad
FROM alumnos
WHERE universidad = :universidad AND abandona = 'no';
                """,
# Consulta que muestra la universidad de un gestor.
            "universidades_gestor": """
SELECT DISTINCT cod_universidad, universidad
FROM gestores
WHERE gestor_id = :gestor_id;
""",
"cursos_academicos_de_egresados": """
SELECT DISTINCT curso_aca
FROM egresados
WHERE cod_universidad = :cod_universidad
""",
},
"filters": {
            # Consulta para obtener los cursos académicos de una universidad.
            "curso_academico_universidad": """
SELECT DISTINCT curso_aca
FROM matricula
WHERE cod_universidad = :cod_universidad
ORDER BY curso_aca;
""",
# Consulta para obtener las titulaciones de una universidad.
            "titulaciones_universidad_gestor": """
SELECT DISTINCT titulacion
FROM matricula
WHERE cod_universidad = :cod_universidad AND curso_aca = :curso_academico;
""",
},
"graphs": {
            "indicadores": {
                # Consulta para obtener el número de alumnos de nuevo ingreso por género y titulación.
                "alumnos_nuevo_ingreso_genero_titulacion": """
SELECT curso_aca, titulacion, sexo, COUNT(*) as num_alumnos
                    FROM matricula
                    WHERE curso_aca = :curso_academico
                    AND titulacion IN :titulaciones
                    AND nuevo_ingreso = 'si'
                    AND cod_universidad = :cod_universidad
                    GROUP BY curso_aca, titulacion, sexo
                    ORDER BY titulacion;
                    """,
# Consulta para obter el número de alumnos egresados por genero y titulación.
                "alumnos_egresados_genero_titulacion": """
SELECT
m.titulacion AS Titulacion,
m.sexo AS Genero,
COUNT(DISTINCT m.id) AS Cantidad
FROM public.egresados e
JOIN public.matricula m ON e.cod_plan = m.cod_plan AND
e.curso_aca = m.curso_aca AND e.id = m.id
WHERE e.cod_universidad = :cod_universidad AND
m.curso_aca = :curso_academico AND
m.titulacion IN :titulaciones
GROUP BY m.titulacion, m.sexo
ORDER BY m.titulacion, m.sexo;
""",
# Consulta para obtener el número de alumnos de nuevo ingreso por nacionalidad y titulación.
                "alumnos_nuevo_ingreso_nacionalidad_titulacion": """
SELECT titulacion, nacionalidad, COUNT(*) as num_alumnos
                    FROM matricula
                    WHERE curso_aca = :curso_academico
                    AND titulacion IN :titulaciones
                    AND nuevo_ingreso = 'si'
                    AND cod_universidad = :cod_universidad
                    GROUP BY titulacion, nacionalidad
                """,
# Consulta para obtener el número de alumnos egresados por nacionalidad y titulación.
                "alumnos_egresados_nacionalidad_titulacion": """
SELECT
m.titulacion AS Titulacion,
m.nacionalidad,
COUNT(DISTINCT m.id) AS Cantidad
FROM public.egresados e
```

```
JOIN public.matricula m ON e.cod_plan = m.cod_plan AND
e.curso_aca = m.curso_aca AND e.id = m.id
WHERE e.cod_universidad = :cod_universidad AND
m.curso_aca = :curso_academico AND
m.titulacion IN :titulaciones
GROUP BY m.titulacion, m.nacionalidad
ORDER BY m.titulacion, m.nacionalidad;
""",
},
"resultados": {
                # Consulta para obtener la nota media de acceso de cada titulación.
                "nota_media_acceso_titulacion": """
SELECT ma.curso_aca, ma.titulacion,
AVG(eb.nota_prue) AS nota_ebau_media
FROM matricula ma
JOIN ebau_prueba eb ON ma.id = eb.id
WHERE nuevo_ingreso = 'si' AND ma.cod_universidad = :cod_universidad
                    GROUP BY ma.curso_aca, ma.titulacion
                    ORDER BY ma.curso_aca, ma.titulacion;
                """,
# Consulta para obtener la duración media de los estudios con respecto a la nota media, por titulación y curso académico.
                "duracion_media_estudios_nota_gestor": """
WITH matriculas_por_estudiante AS (
SELECT
id,
titulacion,
COUNT(*) AS numero_matriculas
FROM matricula
GROUP BY id, titulacion
),
media_notas_y_duracion AS (
SELECT
e.curso_aca,
m.titulacion,
m.rama,
AVG(e.nota_media) AS nota_media_curso_aca,
ROUND(AVG(mpe.numero_matriculas)) AS duracion_media_estudios,
COUNT(DISTINCT e.id) AS numero_alumnos
FROM egresados e
JOIN matricula m ON e.id = m.id AND e.cod_plan = m.cod_plan AND e.cod_universidad = m.cod_universidad
JOIN matriculas_por_estudiante mpe ON mpe.id = m.id AND mpe.titulacion = m.titulacion
WHERE e.cod_universidad = :cod_universidad
GROUP BY e.curso_aca, m.titulacion, m.rama
)
SELECT
nota_media_curso_aca,
titulacion,
rama,
curso_aca,
duracion_media_estudios,
numero_alumnos
FROM media_notas_y_duracion
ORDER BY curso_aca;

""",
},
"riesgo_abandono": {
                # Consulta para obtener la tasa de abandono de una titulación.
                "tasa_abandono_titulacion_gestor": """
WITH ultima_matricula AS (
SELECT
m.id,
m.curso_aca,
m.titulacion,
t.abandona,
ROW_NUMBER() OVER (PARTITION BY m.id ORDER BY m.curso_aca DESC) AS row_num
FROM
public.matricula m
JOIN
public.alumnos t ON m.id = t.id
)

SELECT
m.curso_aca,
m.titulacion,
COUNT(DISTINCT m.id) AS numero_matriculados,
COUNT(DISTINCT CASE WHEN le.abandona = 'si' AND le.row_num = 1 THEN le.id END) AS numero_abandono
                            FROM
                                public.matricula m
                            LEFT JOIN ultima_matricula le ON m.id = le.id AND m.curso_aca = le.curso_aca
                            WHERE
                                m.cod_universidad = :cod_universidad
                                AND m.curso_aca IN :curso_academico
                            GROUP BY
                                m.curso_aca,
```

```
                                m.titulacion
                            ORDER BY
                                m.curso_aca,
                                m.titulacion;

                        """,
# Consulta para obtener la tasa de graduación de una titulación.
                "tasa_graduacion_titulacion_gestor": """
SELECT
COUNT(matricula.id) AS numero_matriculados,
COALESCE(COUNT( DISTINCT egresados.id), 0) AS numero_egresados,
egresados.curso_aca,
matricula.titulacion
FROM
matricula
LEFT JOIN
egresados ON matricula.id = egresados.id
WHERE
matricula.cod_universidad = :cod_universidad AND
matricula.curso_aca IN :curso_academico
GROUP BY
egresados.curso_aca,
matricula.titulacion
ORDER BY
titulacion,
curso_aca;
""",
},
},
},
}
```

6.247. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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.

```
from dash import html
import components.alumnado.utils.tabs_alumnado as tabs_alumnado


def alumno_layout():
    """
Retorna el layout del perfil "Alumno"

    Returns:
    html.Div: Layout del alumno
    """
return html.Div([tabs_alumnado.tabs_alumnado()])
```

6.249. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/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 from dash import html
00002 import components.docente.utils.tabs_docente as tabs_docente
00003
00004
00005 def docente_layout():
00006     """
00007 Retorna el layout del perfil "Docente"
00008
00009     Returns:
00010     html.Div: Layout del docente
00011     """
00012 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 from dash import html
00002 import components.gestor.utils.tabs_gestor as tabs_gestor
00003
00004
00005 def gestor_layout():
00006     """
00007 Retorna el layout del perfil "Gestor"
00008
00009     Returns:
00010     html.Div: Layout del gestor
00011     """
00012 return html.Div([tabs_gestor.tabs_gestor()])
```

6.253. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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_ebau', '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 = {}
- model.total
- model.desc
- model.grid_search = GridSearchCV(model, param_grids[key], cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)
- model.voting_clf
- string model.joblib_file = "trained_model.pkl"
- model.y_pred_voting = voting_clf.predict(X_test)
- model.predicciones
- 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 import pandas as pd
00002 import numpy as np
00003 from datetime import datetime
00004 from sklearn.impute import SimpleImputer
00005 from sklearn.preprocessing import StandardScaler, OneHotEncoder, PolynomialFeatures
00006 from sklearn.compose import ColumnTransformer
00007 from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold, cross_val_score
00008 from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier,
      AdaBoostClassifier
```

```
from sklearn.linear_model import LogisticRegression  # Corrección en la importación
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, f1_score, roc_auc_score
from sklearn.pipeline import Pipeline
import joblib
from tqdm import tqdm
from imblearn.over_sampling import SMOTE
import xgboost as xgb
from util import load_data_for_model # Corrección en la importación

data = load_data_for_model()  # Corrección en la carga de datos

current_year = datetime.now().year
data['edad_actual'] = current_year - data['anio_nac']

# Codificar variable objetivo
data['abandona'] = data['abandona'].map({'si': 1, 'no': 0})

# Definir columnas
columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
columnas_numericas = ['nota_def_ebau', 'nota_media', 'edad_actual']

# Preprocesamiento
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([('imputer', SimpleImputer(strategy='mean')), ('scaler', StandardScaler())]), columnas_numericas),
        ('cat', Pipeline([('imputer', SimpleImputer(strategy='constant', fill_value='Desconocido')),
                          ('encoder', OneHotEncoder(handle_unknown='ignore'))]), columnas_categoricas)
    ])

# Dividir datos en características y variable objetivo
X = data[columnas_categoricas + columnas_numericas]
y = data['abandona']
ids = data['id']

# Dividir los datos en conjuntos de entrenamiento y prueba
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)

# Aplicar preprocesamiento y SMOTE en el conjunto de entrenamiento
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)

smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)

# Feature engineering: polynomial features
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
X_train = poly.fit_transform(X_train)
X_test = poly.transform(X_test)

# Crear pipelines para cada modelo
modelos = {
    'rf': RandomForestClassifier(random_state=42, n_jobs=-1),
    'lr': LogisticRegression(max_iter=1000, random_state=42),
    'gb': GradientBoostingClassifier(random_state=42),
    'ab': AdaBoostClassifier(random_state=42),
    'xgb': xgb.XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss')
}

# Definir el rango de parámetros para la búsqueda
param_grids = {
    'rf': {
        'n_estimators': [100, 200, 300],
        'max_depth': [10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'bootstrap': [True, False]
    },
    'lr': {
        'C': [0.01, 0.1, 1.0, 10.0],
        'solver': ['liblinear', 'lbfgs']
    },
    'gb': {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.01, 0.05, 0.1],
        'max_depth': [3, 5, 7]
    },
    'ab': {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.05, 0.1]
    },
    'xgb': {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.01, 0.05, 0.1],
        'max_depth': [3, 5, 7]
```

```
    }
}

# Buscar los mejores hiperparámetros con GridSearchCV
best_estimators = {}
with tqdm(total=len(modelos), desc="Entrenando modelos") as pbar:
    for key, model in modelos.items():
        grid_search = GridSearchCV(model, param_grids[key], cv=StratifiedKFold(n_splits=5), scoring='accuracy', n_jobs=-1)
        grid_search.fit(X_train, y_train)
        best_estimators[key] = grid_search.best_estimator_
        pbar.update(1)

# Crear el VotingClassifier con los mejores estimadores
voting_clf = VotingClassifier(
    estimators=[(key, best_estimators[key]) for key in best_estimators],
    voting='soft',
    n_jobs=-1
)

# Entrenar el VotingClassifier
voting_clf.fit(X_train, y_train)

# Guardar el modelo entrenado
joblib_file = "trained_model.pkl"
joblib.dump(voting_clf, joblib_file)
print(f"Modelo guardado en {joblib_file}")

# Evaluar el modelo
y_pred_voting = voting_clf.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred_voting):.2f}")
print(f"Precision: {precision_score(y_test, y_pred_voting):.2f}")
print(f"Recall: {recall_score(y_test, y_pred_voting):.2f}")
print(f"F1 Score: {f1_score(y_test, y_pred_voting):.2f}")
print(f"ROC AUC: {roc_auc_score(y_test, y_pred_voting):.2f}")
print("Matriz de Confusión:")
print(confusion_matrix(y_test, y_pred_voting))

# Crear DataFrame con predicciones y IDs
predicciones = pd.DataFrame({
    'id': id_test,
    'abandona_real': y_test,
    'abandona_predicho': y_pred_voting
})

# Mostrar las predicciones junto con los IDs
print(predicciones)

# Validación cruzada
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)
print(f"Cross-Validation Accuracy: {cross_val_scores_voting.mean():.2f}")
```

6.255. Referencia del Archivo C:/Users/fabri/Documents/Fabrizzio/↩ 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 import random
00002 import pandas as pd
00003 import numpy as np
00004 import xgboost as xgb
00005 from datetime import datetime
00006 from sklearn.impute import SimpleImputer
00007 from sklearn.preprocessing import StandardScaler, OneHotEncoder, PolynomialFeatures
00008 from sklearn.compose import ColumnTransformer
00009 from sklearn.pipeline import Pipeline
00010 import joblib
00011 from data.queries import data_for_model
00012
00013
00014 # Función para convertir una lista en una tupla
00015 def list_to_tuple(value):
00016     """
00017 Función para convertir una lista en una tupla
00018
00019 Args:
00020 value: Lista o str a convertir en tupla
00021
00022 Returns:
00023 Tupla con el valor de entrada
00024 """
00025 if isinstance(value, str):
00026         return (value,)
00027     elif isinstance(value, list):
00028         return tuple(value)
00029     else:
00030         return ValueError(
00031             "Tipo de dato no soportado, se esperaba un list o str. {}".format(
00032                 type(value)
00033             )
00034         )
00035
00036
00037 def random_color(size):
00038     """
00039 Función para generar colores aleatorios en formato hexadecimal
00040
00041 Args:
00042 size: Cantidad de colores a generar
00043
00044 Returns:
00045 Lista de colores en formato hexadecimal
00046 """
00047 return ["#%06X"% random.randint(0, 0xFFFFFF) for _ in range(size)]
00048
00049
00050 # Configuración de los botones de la barra de herramientas de Plotly
00051 config_mode_bar_buttons_gestor = {
00052     "displayModeBar": True,
00053     "displaylogo": False,
00054     "scrollZoom": True,
00055     "modeBarButtonsToRemove": ["zoom2d", "lasso2d", "resetScale2d"],
00056     "modeBarButtonsToAdd": [
00057         "drawline",
00058         "drawcircle",
00059         "drawrect",
00060         "eraseshape",
00061         "toggleSpikelines",
00062     ],
00063 }
00064
00065
00066 def load_data_for_model():
00067     """
00068 Función para cargar los datos del modelo
00069
00070 Returns:
00071 pd.DataFrame: Datos del modelo
00072 """
00073 data = data_for_model()
00074 df = pd.DataFrame(data)
00075
00076 # Ajustar los tipos de datos de las columnas
00077     df = df.astype({
00078         'id': str,
00079         'anio_nac': int,
00080         'nacionalidad': str,
00081         'sexo': str,
00082         'titulacion': str,
```

```
        'nota_def_ebau': float,
        'nota_media': float,
        'abandona': str
    })

    return df

def load_model():

    model_file = "src/trained_model.pkl"
    voting_clf = joblib.load(model_file)
    data = load_data_for_model()

    current_year = datetime.now().year
    data['edad_actual'] = current_year - data['anio_nac']

    # Codificar variable objetivo
    data['abandona'] = data['abandona'].map({'si': 1, 'no': 0})

    # Definir columnas
    columnas_categoricas = ['nacionalidad', 'sexo', 'titulacion']
    columnas_numericas = ['nota_def_ebau', 'nota_media', 'edad_actual']

    # Imputación de datos
    num_imputer = SimpleImputer(strategy='mean')
    cat_imputer = SimpleImputer(strategy='constant', fill_value='Desconocido')

    # Preprocesamiento
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', Pipeline([('imputer', num_imputer), ('scaler', StandardScaler())]), columnas_numericas),
            ('cat', Pipeline([('imputer', cat_imputer), ('encoder', OneHotEncoder(handle_unknown='ignore'))]), columnas_categoricas)
        ]
    )

    # Dividir datos en características y variable objetivo
    X = data[columnas_categoricas + columnas_numericas]
    y = data['abandona']
    ids = data['id']

    # Filtrar los estudiantes que no han abandonado aún
    X_non_abandon = X[y == 0]
    data_non_abandon = data[y == 0].copy()  # Mantener una copia de los datos originales

    # Aplicar preprocesamiento
    X_non_abandon_preprocessed = preprocessor.fit_transform(X_non_abandon)

    # Feature engineering: polynomial features
    poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
    X_non_abandon_poly = poly.fit_transform(X_non_abandon_preprocessed)

    # Predecir probabilidades de abandono
    probabilidades_abandono = voting_clf.predict_proba(X_non_abandon_poly)[:, 1]

    # Añadir las probabilidades al DataFrame original de los estudiantes que no han abandonado
    data_non_abandon['probabilidad_abandono'] = probabilidades_abandono

    return data_non_abandon
```

Índice alfabético

__init__
data.db_connector.DatabaseConnector, 217
_favicon
app, 13

alumno_layout
layouts.alumno_layout, 203
alumnos_all
data.queries, 166
alumnos_df
data.generate_synthetic_data, 162
alumnos_egresados_genero_titulacion
data.queries, 167
alumnos_egresados_nacionalidad_titulacion
data.queries, 168
alumnos_genero_docente
data.queries, 169
alumnos_nota_cualitativa_docente
data.queries, 169
alumnos_nota_media_docente
data.queries, 170
alumnos_nuevo_ingreso_genero_titulacion
data.queries, 171
alumnos_nuevo_ingreso_nacionalidad_titulacion
data.queries, 172
alumnos_repetidores_nuevos
data.queries, 173
app, 13
_favicon, 13
app, 13
debug, 13
layout, 14
server, 14
title, 14
asignaturas_actas_titulacion
data.queries, 173
asignaturas_dict
data.generate_synthetic_data, 162
asignaturas_docente
data.queries, 174
asignaturas_matriculadas
data.queries, 175
asignaturas_matriculadas_df
data.generate_synthetic_data, 162
asignaturas_matriculadas_y_superadas
data.queries, 175
asignaturas_superadas
data.queries, 176
asignaturas_superadas_media_abandono
data.queries, 177

best_estimators
model, 206

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
221
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
229, 230
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
230, 231
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
231, 232
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
222
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
232, 233
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
234
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
236
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
238
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
239
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
241
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
242
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
243, 244
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
245
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
245, 246
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_
223

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_collapse.py, 247
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_layout.py, 247, 248
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_table.py, 248, 249
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/6_curso/BI_Universities_ 223
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/filtrado/6_curso/BI_Universities_ 224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/filtrado/6_curso/BI_Universities_callback.py, 249, 250
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/filtrado/6_curso/BI_Universities_ 250, 251
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/filtrado/6_curso/BI_Universities_ 251
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/filtrado/6_curso/BI_Universities_ 252
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/filtrado/6_curso/BI_Universities_ 253
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/6_curso/BI_Universities_ 224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/general/BI_Universities_ 224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/general/callback_graph.py, 254
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/general/callback_graph.py, 256
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/BI_Universities_ 224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 257
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 258, 259
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 259, 260
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 261
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/6_curso/BI_Universities_ 224
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/callbacks/Estudiantes.py, 262, 263
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/callbacks/Select_universities.py, 263, 264
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/callbacks/Tabs_universities.py, 264
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/6_curso/BI_Universities_ 225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/6_curso/BI_Universities_ 225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/callbacks/Filter_universities.py, 265, 266
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/callbacks/Filter_universities_de.py, 266, 267
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/callbacks/Filter_universities_es.py, 267, 268
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/grafos/6_curso/BI_Universities_ 225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/6_curso/BI_Universities_ 225
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/general/BI_Universities_ 226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/general/callback_graph.py, 291
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/general/callback_graph.py, 292, 293
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/BI_Universities_ 293
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 226
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/grafos/personal/callback_graph.py, 295
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/6_curso/BI_Universities_ 295, 296
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/callbacks/Estudiantes.py, 296, 297
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/callbacks/Select_universities.py, 227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/callbacks/Tabs_universities.py, 297
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/6_curso/BI_Universities_ 298
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/6_curso/BI_Universities_ 227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/callbacks/Filter_universities.py, 299
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/callbacks/Filter_universities_de.py, 300
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/filtrado/callbacks/Filter_universities_es.py, 300, 301
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_universidad/gestion/grafos/6_curso/BI_Universities_ 301

C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/create_graph.py, 227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/create_graph_universities.py, 302
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6.py, 303
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy.py, 303, 304
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy2.py, 304
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy3.py, 304, 305
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_data.py, 305, 306
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy.py, 306
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso.py, 227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/BI.py, 227
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_asignaturas.py, 307
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_curso_academico.py, 308
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_universidad.py, 308, 309
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_curso_academico.py, 309, 310
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_universidad.py, 310
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/BI_up.py, 228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/graphs/BI_universidad.py, 311
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/graphs/BI_universidad.py, 312
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/BI.py, 228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/recomendador_docente_queries, 312, 313
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/resumen_docente.py, 313, 314
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/select_docente.py, 314
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/tabs_docente.py, 315
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/__init__.py, 228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/__init__.py, 228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_all_curso_academico, 315, 316
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_curso_academico_universidad, 316, 317
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_titulaciones_gestor, 317
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/__init__.py, 228
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6.py, 318
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy.py, 319, 320
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy2.py, 320, 321
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy3.py, 229
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_data.py, 321, 322
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6copy.py, 322
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso.py, 323
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/BI.py, 324
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_asignaturas.py, 229
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_curso_academico.py, 324, 325
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_universidad.py, 326, 327
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_curso_academico.py, 331, 332
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_universidad.py, 338
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_curso_academico.py, 229
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/filtro/BI_universidad.py, 347
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/BI_up.py, 348
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/graphs/BI_universidad.py, 348
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/graphs/BI_universidad.py, 349
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/d6_curso/BI.py, 351, 352
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/recomendador_docente_queries, 178
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/resumen_docente.py, 178
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/select_docente.py, 179
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/docente/utils/tabs_docente.py, 180
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/__init__.py, 181
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/__init__.py, 181
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_all_curso_academico, 182
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_curso_academico_universidad, 14
callbacks.alumnado, 14
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/filters/filter_titulaciones_gestor, 15
callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnado, 15
C:/Users/fabri/Documents/Fabrizzio/Universidad/6_curso/BI_Universities_TFG/src/components/gestor/graphs/__init__.py, 228
update_filter_asignaturas_matri_alumnado, 15

callbacks.alumnado.filters.callback_filter_curso_cademico_alumnado, 16
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_suspendidas_alumnado, 19
update_graph_alumnado, 19
callbacks.alumnado.graphs.general.callback_graph_calif_cual_comparativa, 21
update_graph_alumnado, 21
callbacks.alumnado.graphs.general.callback_graph_calif_media_alumnado, 23
update_graph_alumnado, 23
callbacks.alumnado.graphs.personal, 25
callbacks.alumnado.graphs.personal.callback_graph_calif_cualitativa_alumnado, 25
update_graph_alumnado, 25
callbacks.alumnado.graphs.personal.callback_graph_calif_numerica_alumnado, 27
update_graph_alumnado, 27
callbacks.alumnado.graphs.personal.callback_graph_progreso_academico_alumnado, 29
update_graph_alumnado, 29
callbacks.alumnado.graphs.personal.callback_graph_tasa_rendimiento_alumnado, 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_sidebar_collapse, 36
sidebar_collapse, 37
callbacks.common.callback_update_layout, 37
update_layout, 37
callbacks.common.callback_user_role, 38
initialize_dropdown, 39
prevent_initial_call, 40
update_role, 39
callbacks.docente, 40
callbacks.docente.filters, 40
callbacks.docente.filters.callback_filter_all_asignaturas_titulacion_docente, 40
update_filter_asignaturas_docente, 40
callbacks.docente.filters.callback_filter_all_curso_academico, 42
update_filter_all_cursos_academicos_docente, 42
callbacks.docente.filters.callback_filter_asignaturas_docente, 43
update_filter_asignaturas_docente, 43
callbacks.docente.filters.callback_filter_curso_academico_docente, 44
update_filter_curso_academico_docente, 44
callbacks.docente.filters.callback_filter_titulacion_docente, 45
update_filter_titulacion_docente, 45
callbacks.docente.graphs, 46
callbacks.docente.graphs.general, 46
callbacks.docente.graphs.general.callback_graph_all_calif_cualitativa_docente, 46
update_graph_docente, 46
callbacks.docente.graphs.general.callback_graph_all_calif_media_docente, 48
update_graph_docente, 48
callbacks.docente.graphs.personal, 49
callbacks.docente.graphs.personal.callback_graph_alu_genero_docente, 50
update_graph_docente, 50
callbacks.docente.graphs.personal.callback_graph_alu_media_docente, 51
update_graph_docente, 51
callbacks.docente.graphs.personal.callback_graph_alu_nota_cualitativa_docente, 52
update_graph_docente, 53
callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevo_docente, 54
update_graph_docente, 54
callbacks.docente.utils, 56
callbacks.docente.utils.callback_resumen_docente, 56
not_data, 56
update_resumen_docente, 56
callbacks.docente.utils.callback_select_docente, 57
store_selected_docente, 58
callbacks.docente.utils.callback_tabs_docente, 58
render_content, 59
callbacks.gestor, 60
callbacks.gestor.filters, 60
callbacks.gestor.filters.callback_filter_all_curso_academico_gestor, 61
update_filter_all_curso_academico_gestor, 61
callbacks.gestor.filters.callback_filter_curso_academico_gestor, 62
update_filter_curso_academico_gestor, 62
callbacks.gestor.filters.callback_filter_titulaciones_gestor, 63
update_filter_titulaciones_gestor, 63
callbacks.gestor.graphs, 65
callbacks.gestor.graphs.indicadores, 65
callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor, 65
generate_csv, 65
get_data, 66
toggle_modal, 67
update_graph_gestor, 68
update_table, 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_nuevo_curso_gestor, 76
generate_csv, 76
get_data, 77
toggle_modal, 78
update_graph_gestor, 79
update_table, 81
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_curso_tasa_abandono_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_media_estudios_gestor, 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_grpht_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_graduacion_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, 183
close
data.db_connector.DatabaseConnector, 218
cod_asignaturas_dict
data.generate_synthetic_data, 162
cod_curso_dictero_gestor,
data.generate_synthetic_data, 163
columnas_categoricas
model, 206
columnas_numericas
model, 206
components, 114
components.alumnado, 114
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.filters.filter_titulacion_alumnado, 117
filter_titulacion_alumnado, 117
components.alumnado.graphs, 118
components.alumnado.graphs.graphs_general_alumnado, 118
graphs_general_alumnado, 118
components.alumnado.graphs.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.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_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_docente, 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, 139
components.docente.utils.recomendador_docente, 140
recomendador_docente, 140
components.docente.utils.resumen_docente, 141
resumen_docente, 141
components.docente.utils.select_docente, 141
select_docente, 141
components.docente.utils.tabs_docente, 142
tabs_docente, 142
components.gestor, 143
components.gestor.filters, 143
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, 148
graphs_resultados_gestor, 148
components.gestor.graphs.graphs_riesgo_abandono_gestor, 150
graphs_riesgo_abandono_gestor, 150
components.gestor.utils, 151
components.gestor.utils.recomendador_gestor, 151
recomendador_gestor, 152
components.gestor.utils.resumen_gestor, 152
resumen_gestor, 153
components.gestor.utils.select_gestor, 153
select_gestor, 153
components.gestor.utils.tabs_gestor, 154
tabs_gestor, 154
config
data.db_connector, 156
config_mode_bar_buttons_gestor
util, 216
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, 206
current_year
model, 206
curso_academico_actas_titulacion
data.queries, 184
curso_academico_alumnado
data.queries, 185
curso_academico_docente
data.queries, 186
curso_academico_universidad
data.queries, 186
cursos_academicos
data.generate_synthetic_data, 163
cursos_academicos_egresados
data.queries, 187

data, 155
model, 207
data.db_connector, 155
config, 156
db, 156
data.db_connector.DatabaseConnector, 217
__init__, 217
close, 218
db_url, 219
engine, 219
execute_query, 218
data.generate_synthetic_data, 156
alumnos_df, 162
asignaturas_dict, 162
asignaturas_matriculadas_df, 162
cod_asignaturas_dict, 162
cod_plan_dict, 163
cursos_academicos, 163
docentes_df, 163
ebau_prueba_df, 163
educacion, 163
egresados_df, 163
fake, 164
generate_alumnos, 157
generate_asignaturas, 157
generate_asignaturas_matriculadas, 158
generate_curso_aca, 158
generate_docentes, 158
generate_ebau_prueba, 159

generate_egresados, 159
generate_gestores, 160
generate_lineas_actas, 160
generate_matricula, 161
generate_unique_cod_asignaturas, 162
gestores_df, 164
index, 164
lineas_actas_df, 164
matricula_df, 164
num_alumnos, 164
num_docentes, 165
num_gestores, 165
universidades_dict, 165
data.queries, 165
alumnos_all, 166
alumnos_egresados_genero_titulacion, 167
alumnos_egresados_nacionalidad_titulacion, 168
alumnos_genero_docente, 169
alumnos_nota_cualitativa_docente, 169
alumnos_nota_media_docente, 170
alumnos_nuevo_ingreso_genero_titulacion, 171
alumnos_nuevo_ingreso_nacionalidad_titulacion, 172
alumnos_repetidores_nuevos, 173
asignaturas_actas_titulacion, 173
asignaturas_docente, 174
asignaturas_matriculadas, 175
asignaturas_matriculadas_y_superadas, 175
asignaturas_superadas, 176
asignaturas_superadas_media_abandono, 177
cache_query, 178
calif_all_cualitativa_asignaturas, 178
calif_cualitativa_alumno_asignaturas, 179
calif_cualitativa_asignatura, 180
calif_cualitativa_comparativa, 181
calif_media_asignaturas, 181
calif_numerica_asignatura, 182
check_data, 183
curso_academico_actas_titulacion, 184
curso_academico_alumnado, 185
curso_academico_docente, 186
curso_academico_universidad, 186
cursos_academicos_egresados, 187
data_for_model, 188
docentes_all, 189
duracion_media_estudios_nota_gestor, 189
gestores_all, 190
nota_media_acceso_titulacion, 191
nota_media_alumno_titulacion, 192
nota_media_general_mi_nota, 192
numero_alumnos_matriculados_universidad, 193
resumen_alumno, 194
resumen_docente, 195
resumen_gestor, 195
tasa_abandono_titulacion_gestor, 196
tasa_graduacion_titulacion_gestor, 197
titulacion_alumnado, 197
titulacion_docente, 198
titulaciones_universidad_gestor, 199
universidad_alumno, 199
universidades_docente, 200
universidades_gestor, 201
data.queries_dictionary, 202
queries, 203
data_for_model
data.queries, 188
db
data.db_connector, 156
db_url
data.db_connector.DatabaseConnector, 219
debug
app, 13
desc
model, 207
docente_layout
layouts.docente_layout, 204
docentes_all
data.queries, 189
docentes_df
data.generate_synthetic_data, 163
duracion_media_estudios_nota_gestor
data.queries, 189

ebau_prueba_df
data.generate_synthetic_data, 163
educacion
data.generate_synthetic_data, 163
egresados_df
data.generate_synthetic_data, 163
engine
data.db_connector.DatabaseConnector, 219
execute_query
data.db_connector.DatabaseConnector, 218

fake
data.generate_synthetic_data, 164
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, 157
generate_asignaturas
data.generate_synthetic_data, 157
generate_asignaturas_matriculadas
data.generate_synthetic_data, 158
generate_csv
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.callback_graph_duracion_media_estudios_nota_gestor, 88
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor, 93
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor, 98
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor, 104
generate_curso_aca
data.generate_synthetic_data, 158
generate_docentes
data.generate_synthetic_data, 158
generate_ebau_prueba
data.generate_synthetic_data, 159
generate_egresados
data.generate_synthetic_data, 159
generate_gestores
data.generate_synthetic_data, 160
generate_lineas_actas
data.generate_synthetic_data, 160
generate_matricula
data.generate_synthetic_data, 161
generate_unique_cod_asignaturas
data.generate_synthetic_data, 162
gestor_layout
layouts.gestor_layout, 204
gestores_all
data.queries, 190
gestores_df
data.generate_synthetic_data, 164
get_data
callbacks.gestor.graphs.indicadores.callback_graph_egresados_gene
66
callbacks.gestor.graphs.indicadores.callback_graph_egresados_naci
72
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_
77
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_
83
callbacks.gestor.graphs.resultados.callback_graph_duracion_media_
89
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titu
94
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_aban
99
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_grad
105
graphs_general_alumnado
components.alumnado.graphs.graphs_general_alumnado, 118
graphs_general_docente
components.docente.graphs.graphs_general_docente, 137
graphs_indicadores_gestor
components.gestor.graphs.graphs_indicadores_gestor, 147
graphs_personal_alumnado
components.alumnado.graphs.graphs_personal_alumnado, 119
graphs_personal_docente
components.docente.graphs.graphs_personal_docente, 139
graphs_resultados_gestor
components.gestor.graphs.graphs_resultados_gestor, 148
graphs_riesgo_abandono_gestor
components.gestor.graphs.graphs_riesgo_abandono_gestor, 150
grid_search
model, 207

header
components.common.header, 129

id_test
model, 207
id_train
model, 207
ids
model, 207
index
data.generate_synthetic_data, 164
initialize_dropdown
callbacks.common.callback_user_role, 39

joblib_file

model, 208

layout
app, 14
layouts, 203
layouts.alumno_layout, 203
alumno_layout, 203
layouts.docente_layout, 204
docente_layout, 204
layouts.gestor_layout, 204
gestor_layout, 204
lineas_actas_df
data.generate_synthetic_data, 164
list_to_tuple
util, 212
load_data_for_model
util, 213
load_model
util, 214

matricula_df
data.generate_synthetic_data, 164
model, 205
best_estimators, 206
columnas_categoricas, 206
columnas_numericas, 206
cross_val_scores_voting, 206
current_year, 206
data, 207
desc, 207
grid_search, 207
id_test, 207
id_train, 207
ids, 207
joblib_file, 208
modelos, 208
param_grids, 208
poly, 208
predicciones, 209
preprocessor, 209
random_state, 209
smote, 209
stratify, 209
test_size, 210
total, 210
voting_clf, 210
X, 210
X_test, 210
X_train, 210
y, 211
y_pred_voting, 211
y_test, 211
y_train, 211
modelos
model, 208

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, 191
nota_media_alumno_titulacion
data.queries, 192
nota_media_general_mi_nota
data.queries, 192
num_alumnos
data.generate_synthetic_data, 164
num_docentes
data.generate_synthetic_data, 165
num_gestores
data.generate_synthetic_data, 165
numero_alumnos_matriculados_universidad
data.queries, 193

param_grids
model, 208
poly
model, 208
predicciones
model, 209
preprocessor
model, 209
prevent_initial_call
callbacks.common.callback_user_role, 40

queries
data.queries_dictionary, 203

random_color
util, 215
random_state
model, 209
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, 194
resumen_docente
components.docente.utils.resumen_docente, 141
data.queries, 195

resumen_gestor
components.gestor.utils.resumen_gestor, 153
data.queries, 195

select_alumnado
components.alumnado.utils.select_alumnado, 122
select_docente
components.docente.utils.select_docente, 141
select_gestor
components.gestor.utils.select_gestor, 153
server
app, 14
sidebar
components.common.sidebar, 131
sidebar_collapse
callbacks.common.callback_sidebar_collapse, 37
smote
model, 209
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, 209

tabs_alumnado
components.alumnado.utils.tabs_alumnado, 123
tabs_docente
components.docente.utils.tabs_docente, 142
tabs_gestor
components.gestor.utils.tabs_gestor, 154
tasa_abandono_titulacion_gestor
data.queries, 196
tasa_graduacion_titulacion_gestor
data.queries, 197
test_size
model, 210
title
app, 14
titulacion_alumnado
data.queries, 197
titulacion_docente
data.queries, 198
titulaciones_universidad_gestor
data.queries, 199
toggle_modal
callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor, 67
callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor, 73
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor, 78
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor, 84
callbacks.gestor.graphs.resultados.callback_graph_duracion_media_ 90
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titul 95
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_aban 101
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_grad 106
total
model, 210

universidad_alumno
data.queries, 199
universidades_dict
data.generate_synthetic_data, 165
universidades_docente
data.queries, 200
universidades_gestor
data.queries, 201
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_do 40
callbacks.docente.filters.callback_filter_asignaturas_docente, 43
update_filter_asignaturas_matri_alumnado
callbacks.alumnado.filters.callback_filter_asignaturas_matri_alumnad 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
update_graph_alumnado
callbacks.alumnado.graphs.general.callback_graph_asig_superadas_ 19
callbacks.alumnado.graphs.general.callback_graph_calif_cual_compa 21
callbacks.alumnado.graphs.general.callback_graph_calif_media_mi_ 23

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
update_graph_docente
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.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, 53
callbacks.docente.graphs.personal.callback_graph_alu_repetidores_nuevos_docente, 54
update_graph_gestor
callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor, 68
callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor, 74
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor, 79
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor, 85
callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor, 91
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor, 101
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor, 107
update_grpht_gestor
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor, 96
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
update_table
callbacks.gestor.graphs.indicadores.callback_graph_egresados_genero_gestor, 69
callbacks.gestor.graphs.indicadores.callback_graph_egresados_nacionalidad_gestor, 75
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_genero_gestor, 81
callbacks.gestor.graphs.indicadores.callback_graph_nuevo_ingreso_nacionalidad_gestor, 86
callbacks.gestor.graphs.resultados.callback_graph_duracion_media_estudios_nota_gestor, 92
callbacks.gestor.graphs.resultados.callback_graph_nota_acceso_titulacion_gestor, 97
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_abandono_gestor, 103
callbacks.gestor.graphs.riesgo_abandono.callback_graph_tasa_graduacion_gestor, 108
utils, 211
config_mode_bar_buttons_gestor, 216
list_to_tuple, 212
load_data_for_model, 213
load_model, 214
random_color, 215
voting_clf
model, 210
X
model, 210
X_test
model, 210
X_train
model, 210
y
model, 211
y_pred_voting
model, 211
y_test
model, 211
y_train
model, 211