

# CLIPS

Communication & Localization with Indoor Positioning Systems

---

UNIVERSITÀ DI PADOVA

DEFINIZIONE DI PRODOTTO v1.00



[leaf.gruppo@gmail.com](mailto:leaf.gruppo@gmail.com)

<b>Versione</b>	1.00
<b>Data Redazione</b>	2016-04-10
<b>Redazione</b>	Marco Zanella Oscar Elia Conti Andrea Tombolato Federico Tavella Davide Castello Eduard Bicego
<b>Verifica</b>	Cristian Andrighetto
<b>Approvazione</b>	Oscar Conti
<b>Uso</b>	Esterno
<b>Distribuzione</b>	Prof. Vardanega Tullio Prof. Cardin Riccardo Miriade S.p.A.

## Diario delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
1.02	2016-04-28	Eduard Bicego	Progettista	Aggiunta sezione architettura applicazione e diagramma di sequenza Avvio bussola
1.01	2016-04-26	Eduard Bicego	Progettista	Migliorata la precisione dei riferimenti
1.00	2016-04-10	Oscar Elia Conti	Responsabile di progetto	Approvazione del documento
0.20	2016-04-10	Cristian Andrichetto	Verificatore	Verifica del documento
0.19	2016-04-10	Davide Castello	Progettista	Aggiunto tracciamento classi-requisiti e viceversa
0.18	2016-04-09	Eduard Bicego	Progettista	Correzioni delle classi ImageDetailActivity, ImageListFragment, ImageListFragmentViewImp, PreferenceViewImp, LoggingView, LoggingViewImp, DeveloperUnlockerViewImp, HelpViewImp, MapDownloaderActivity e LogInformationActivity

<b>Versione</b>	<b>Data</b>	<b>Autore</b>	<b>Ruolo</b>	<b>Descrizione</b>
0.17	2016-04-09	Eduard Bicego	Progettista	Correzioni minori nei diagrammi di sequenza, aggiunto appendice A
0.16	2016-04-08	Federico Tavella	Progettista	Correzioni nel package beacon
0.15	2016-04-08	Cristian Andrighetto	Verificatore	Verifica del documento
0.14	2016-04-08	Marco Zanella	Progettista	Correzioni generali su componenti del model
0.13	2016-04-08	Andrea Tombolato	Progettista	Aggiunte componenti view
0.12	2016-04-08	Oscar Elia Conti	Progettista	Aggiunte componenti presenter
0.11	2016-04-07	Cristian Andrighetto	Verificatore	Verifica del documento
0.10	2016-04-07	Marco Zanella	Progettista	Aggiunte componenti model
0.09	2016-04-06	Eduard Bicego	Progettista	Aggiunta sottosezione Metodo e formalismo di specifica
0.06	2016-04-06	Eduard Bicego	Progettista	Aggiungi diagrammi di sequenza Avvio Service, Ranging Beacons e avvio navigazione
0.05	2016-04-03	Eduard Bicego	Progettista	Completata sezione Standard di progetto

<b>Versione</b>	<b>Data</b>	<b>Autore</b>	<b>Ruolo</b>	<b>Descrizione</b>
0.04	2016-04-02	Eduard Bicego	Progettista	Aggiornata sezione Introduzione
0.03	2016-03-22	Oscar Elia Conti	Progettista	Aggiunta sezione "Specifica dei componenti"
0.02	2016-03-22	Oscar Elia Conti	Progettista	Aggiunta sezione "Standard di progetto"
0.01	2016-03-18	Oscar Elia Conti	Progettista	Definizione struttura documento

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento . . . . .	1
1.2	Scopo del prodotto . . . . .	1
1.3	Glossario . . . . .	1
1.4	Riferimenti utili . . . . .	1
1.4.1	Riferimenti normativi . . . . .	1
1.4.2	Riferimenti informativi . . . . .	1
<b>2</b>	<b>Standard di progetto</b>	<b>3</b>
2.1	Standard di documentazione del codice . . . . .	3
2.2	Standard di denominazione di entità e relazioni . . . . .	3
2.3	Standard di programmazione . . . . .	3
2.4	Strumenti di lavoro e procedure . . . . .	3
<b>3</b>	<b>Architettura applicazione</b>	<b>4</b>
3.1	Architettura ad alto livello . . . . .	4
<b>4</b>	<b>Specifiche dei componenti</b>	<b>5</b>
4.1	Metodo e formalismo di specifica . . . . .	5
4.2	Sistema CLIPS . . . . .	6
4.3	Componenti . . . . .	7
4.3.1	model . . . . .	7
4.3.2	model::beacon . . . . .	8
4.3.3	model::compass . . . . .	9
4.3.4	model::dataaccess . . . . .	9
4.3.5	model::dataaccess::dao . . . . .	10
4.3.6	model::dataaccess::service . . . . .	12
4.3.7	model::navigator . . . . .	13
4.3.8	model::navigator::algorithm . . . . .	14
4.3.9	model::navigator::graph . . . . .	15
4.3.10	model::navigator::graph::area . . . . .	16
4.3.11	model::navigator::graph::edge . . . . .	16
4.3.12	model::navigator::graph::navigationinformation . . . . .	17
4.3.13	model::navigator::graph::vertex . . . . .	18
4.3.14	model::usersetting . . . . .	18
4.3.15	presenter . . . . .	19
4.3.16	view . . . . .	20
4.4	Classi . . . . .	22
4.4.1	model::AbsBeaconReceiverManager . . . . .	22

4.4.2	model::InformationManager . . . . .	24
4.4.3	model::InformationManagerImp . . . . .	26
4.4.4	model::MessageSendType . . . . .	29
4.4.5	model::NavigationManager . . . . .	30
4.4.6	model::NavigationManagerImp . . . . .	32
4.4.7	model::NoBeaconSeenException . . . . .	35
4.4.8	model::ServiceConnectionImp . . . . .	36
4.4.9	model::beacon::BeaconManagerAdapter . . . . .	37
4.4.10	model::beacon::BeaconRanger . . . . .	40
4.4.11	model::beacon::LocalBinder . . . . .	41
4.4.12	model::beacon::Logger . . . . .	42
4.4.13	model::beacon::LoggerImp . . . . .	44
4.4.14	model::beacon::MyBeacon . . . . .	46
4.4.15	model::beacon::MyBeaconImp . . . . .	48
4.4.16	model::beacon::MyDistanceCalculator . . . . .	50
4.4.17	model::compass::Compass . . . . .	51
4.4.18	model::dataaccess::dao::BuildingContract . . . . .	53
4.4.19	model::dataaccess::dao::BuildingDao . . . . .	55
4.4.20	model::dataaccess::dao::BuildingTable . . . . .	57
4.4.21	model::dataaccess::dao::CategoryContract . . . . .	59
4.4.22	model::dataaccess::dao::CategoryDao . . . . .	60
4.4.23	model::dataaccess::dao::CategoryTable . . . . .	62
4.4.24	model::dataaccess::dao::CursorConverter . . . . .	63
4.4.25	model::dataaccess::dao::DaoFactoryHelper . . . . .	64
4.4.26	model::dataaccess::dao::EdgeContract . . . . .	65
4.4.27	model::dataaccess::dao::EdgeDao . . . . .	66
4.4.28	model::dataaccess::dao::EdgeTable . . . . .	68
4.4.29	model::dataaccess::dao::EdgeTypeContract . . . . .	70
4.4.30	model::dataaccess::dao::EdgeTypeDao . . . . .	71
4.4.31	model::dataaccess::dao::EdgeTypeTable . . . . .	73
4.4.32	model::dataaccess::dao::MapsDbContract . . . . .	74
4.4.33	model::dataaccess::dao::MapsDbHelper . . . . .	75
4.4.34	model::dataaccess::dao::PhotoContract . . . . .	77
4.4.35	model::dataaccess::dao::PhotoDao . . . . .	78
4.4.36	model::dataaccess::dao::PhotoTable . . . . .	80
4.4.37	model::dataaccess::dao::PointOfInterestContract . . . . .	81
4.4.38	model::dataaccess::dao::PointOfInterestDao . . . . .	82
4.4.39	model::dataaccess::dao::PointOfInterestTable . . . . .	84
4.4.40	model::dataaccess::dao::RegionOfInterestContract . . . . .	86
4.4.41	model::dataaccess::dao::RegionOfInterestDao . . . . .	87
4.4.42	model::dataaccess::dao::RegionOfInterestTable . . . . .	89

4.4.43	model::dataaccess::dao::RemoteBuildingDao . . . . .	90
4.4.44	model::dataaccess::dao::RemoteCategoryDao . . . . .	91
4.4.45	model::dataaccess::dao::RemoteDaoFactory . . . . .	93
4.4.46	model::dataaccess::dao::RemoteEdgeDao . . . . .	94
4.4.47	model::dataaccess::dao::RemoteEdgeTypeDao . . . . .	95
4.4.48	model::dataaccess::dao::RemotePhotoDao . . . . .	96
4.4.49	model::dataaccess::dao::RemotePointOfInterestDao . .	97
4.4.50	model::dataaccess::dao::RemoteRegionOfInterestDao .	98
4.4.51	model::dataaccess::dao::RemoteRoiPoiDao . . . . .	99
4.4.52	model::dataaccess::dao::RoiPoiContract . . . . .	100
4.4.53	model::dataaccess::dao::RoiPoiDao . . . . .	101
4.4.54	model::dataaccess::dao::RoiPoiTable . . . . .	103
4.4.55	model::dataaccess::dao::SQLDao . . . . .	104
4.4.56	model::dataaccess::dao::SQLiteBuildingDao . . . . .	107
4.4.57	model::dataaccess::dao::SQLiteCategoryDao . . . . .	110
4.4.58	model::dataaccess::dao::SQLiteDaoFactory . . . . .	112
4.4.59	model::dataaccess::dao::SQLiteEdgeDao . . . . .	114
4.4.60	model::dataaccess::dao::SQLiteEdgeTypeDao . . . . .	116
4.4.61	model::dataaccess::dao::SQLitePhotoDao . . . . .	118
4.4.62	model::dataaccess::dao::SQLitePointOfInterestDao . .	120
4.4.63	model::dataaccess::dao::SQLiteRegionOfInterestDao .	122
4.4.64	model::dataaccess::dao::SQLiteRoiPoiDao . . . . .	125
4.4.65	model::dataaccess::service::BuildingService . . . . .	127
4.4.66	model::dataaccess::service::DatabaseService . . . . .	131
4.4.67	model::dataaccess::service::EdgeService . . . . .	133
4.4.68	model::dataaccess::service::PhotoService . . . . .	136
4.4.69	model::dataaccess::service::PointOfInterestService . .	138
4.4.70	model::dataaccess::service::RegionOfInterestService .	141
4.4.71	model::dataaccess::service::ServiceHelper . . . . .	143
4.4.72	model::navigator::BuildingInformation . . . . .	144
4.4.73	model::navigator::BuildingMap . . . . .	146
4.4.74	model::navigator::BuildingMapImp . . . . .	148
4.4.75	model::navigator::NavigationExceptions . . . . .	152
4.4.76	model::navigator::Navigator . . . . .	153
4.4.77	model::navigator::NavigatorImp . . . . .	155
4.4.78	model::navigator::NoGraphSetException . . . . .	158
4.4.79	model::navigator::NoNavigationInformationException .	159
4.4.80	model::navigator::PathException . . . . .	161
4.4.81	model::navigator::ProcessedInformation . . . . .	162
4.4.82	model::navigator::ProcessedInformationImp . . . . .	163
4.4.83	model::navigator::algorithm::DijkstraPathFinder . . .	165

4.4.84 model::navigator::algorithm::PathFinder . . . . .	166
4.4.85 model::navigator::graph::MapGraph . . . . .	167
4.4.86 model::navigator::graph::area::PointOfInterest . . . . .	169
4.4.87 model::navigator::graph::area::PointOfInterestImp . . .	170
4.4.88 model::navigator::graph::area::PointOfInterestInformation	172
4.4.89 model::navigator::graph::area::RegionOfInterest . . . . .	174
4.4.90 model::navigator::graph::area::RegionOfInterestImp . .	176
4.4.91 model::navigator::graph::edge::AbsEnrichedEdge . . . . .	178
4.4.92 model::navigator::graph::edge::DefaultEdge . . . . .	181
4.4.93 model::navigator::graph::edge::Edge . . . . .	182
4.4.94 model::navigator::graph::edge::ElevatorEdge . . . . .	183
4.4.95 model::navigator::graph::edge::EnrichedEdge . . . . .	185
4.4.96 model::navigator::graph::edge::StairEdge . . . . .	186
4.4.97 model::navigator::graph::navigationinformation::BasicInformation	188
4.4.98 model::navigator::graph::navigationinformation::DetailedInformation	189
4.4.99 model::navigator::graph::navigationinformation::NavigationInformation	190
4.4.100 model::navigator::graph::navigationinformation::NavigationInformationImp	191
4.4.101 model::navigator::graph::navigationinformation::PhotoInformation	192
4.4.102 model::navigator::graph::navigationinformation::PhotoRef	193
4.4.103 model::navigator::graph::vertex::Vertex . . . . .	194
4.4.104 model::navigator::graph::vertex::VertexImp . . . . .	195
4.4.105 model::usersetting::DeveloperCodeManager . . . . .	196
4.4.106 model::usersetting::InstructionPreference . . . . .	197
4.4.107 model::usersetting::PathPreference . . . . .	197
4.4.108 model::usersetting::Setting . . . . .	198
4.4.109 model::usersetting::SettingImp . . . . .	200
4.4.110 presenter::Checker . . . . .	202
4.4.111 presenter::DatabaseServiceManager . . . . .	203
4.4.112 presenter::DetailedInformationActivity . . . . .	204
4.4.113 presenter::DeveloperUnlockerActivity . . . . .	205
4.4.114 presenter::HelpActivity . . . . .	206
4.4.115 presenter::HomeActivity . . . . .	208
4.4.116 presenter::ImageAdapter . . . . .	210
4.4.117 presenter::ImageDetailActivity . . . . .	212
4.4.118 presenter::ImageListFragment . . . . .	213
4.4.119 presenter::ImagePageAdapter . . . . .	215
4.4.120 presenter::InformationManagerPresenter . . . . .	216
4.4.121 presenter::LocalMapActivity . . . . .	217
4.4.122 presenter::LoggingActivity . . . . .	218
4.4.123 presenter::LogInformationActivity . . . . .	219
4.4.124 presenter::MainActivity . . . . .	220

4.4.125 presenter::MainDeveloperActivity . . . . .	221
4.4.126 presenter::MainDeveloperPresenter . . . . .	223
4.4.127 presenter::MapDownloaderActivity . . . . .	224
4.4.128 presenter::NavigationActivity . . . . .	225
4.4.129 presenter::NavigationAdapter . . . . .	226
4.4.130 presenter::NavigationManagerPresenter . . . . .	228
4.4.131 presenter::NearbyPoiActivity . . . . .	229
4.4.132 presenter::PoiCategoryActivity . . . . .	230
4.4.133 presenter::PreferencesActivity . . . . .	231
4.4.134 presenter::RemoteMapManagerActivity . . . . .	232
4.4.135 presenter::SearchSuggestionsProvider . . . . .	233
4.4.136 presenter::SettingManager . . . . .	235
4.4.137 view::DetailedInformationView . . . . .	236
4.4.138 view::DetailedInformationViewImp . . . . .	237
4.4.139 view::DeveloperUnlockerView . . . . .	238
4.4.140 view::DeveloperUnlockerViewImp . . . . .	239
4.4.141 view::HelpView . . . . .	241
4.4.142 view::HelpViewImp . . . . .	242
4.4.143 view::HomeView . . . . .	243
4.4.144 view::HomeViewImp . . . . .	245
4.4.145 view::ImageDetailView . . . . .	248
4.4.146 view::ImageDetailViewImp . . . . .	249
4.4.147 view::ImageListFragmentView . . . . .	250
4.4.148 view::ImageListFragmentViewImp . . . . .	251
4.4.149 view::LocalMapManagerView . . . . .	252
4.4.150 view::LocalMapManagerViewImp . . . . .	253
4.4.151 view::LoggingView . . . . .	254
4.4.152 view::LoggingViewImp . . . . .	255
4.4.153 view::LogInformationView . . . . .	256
4.4.154 view::LogInformationViewImp . . . . .	257
4.4.155 view::MainDeveloperView . . . . .	259
4.4.156 view::MainDeveloperViewImp . . . . .	260
4.4.157 view::MainView . . . . .	261
4.4.158 view::MainViewImp . . . . .	262
4.4.159 view::MapDownloaderView . . . . .	263
4.4.160 view::MapDownloaderViewImp . . . . .	264
4.4.161 view::NavigationView . . . . .	265
4.4.162 view::NavigationViewImp . . . . .	266
4.4.163 view::NearbyPoiView . . . . .	268
4.4.164 view::NearbyPoiViewImp . . . . .	269
4.4.165 view::PoiCategoryView . . . . .	270

4.4.166 view::PoiCategoryViewImp . . . . .	271
4.4.167 view::PreferencesView . . . . .	272
4.4.168 view::PreferencesViewImp . . . . .	273
4.4.169 view::RemoteMapManagerView . . . . .	275
4.4.170 view::RemoteMapManagerViewImp . . . . .	276
<b>5 Schema base di dati</b>	<b>278</b>
<b>6 Diagrammi di sequenza</b>	<b>279</b>
6.1 Avvio Service per il rilevamento beacon . . . . .	279
6.2 Elaborazione beacon rilevati e comunicazione broadcast . . . . .	281
6.3 Avvio navigazione . . . . .	283
6.4 Avvio della bussola . . . . .	284
<b>7 Tracciamento</b>	<b>285</b>
7.1 Tracciamento Classi-Requisiti . . . . .	285
7.2 Requisiti-Classi . . . . .	300
7.3 Tracciamento Metodi - test di unità . . . . .	318
<b>A Diagrammi riassuntivi package significativi</b>	<b>336</b>

## Elenco delle figure

1	Architettura a layer adottata nell'applicazione . . . . .	4
2	Diagramma dei package - sistema CLIPS . . . . .	6
3	Componente model . . . . .	7
4	Componente model::beacon . . . . .	8
5	Componente model::compass . . . . .	9
6	Componente model::dataaccess . . . . .	9
7	Componente model::dataaccess::dao . . . . .	10
8	Componente model::dataaccess::service . . . . .	12
9	Componente model::navigator . . . . .	13
10	Componente model::navigator::algorithm . . . . .	14
11	Componente model::navigator::graph . . . . .	15
12	Componente model::navigator::graph::area . . . . .	16
13	Componente model::navigator::graph::edge . . . . .	16
14	Componente model::navigator::graph::navigationinformation .	17
15	Componente model::navigator::graph::vertex . . . . .	18
16	Componente model::usersetting . . . . .	18
17	Componente presenter . . . . .	19
18	Componente view . . . . .	20
19	Classe astratta AbsBeaconReceiverManager . . . . .	22
20	Interfaccia InformationManager . . . . .	24
21	Classe InformationManagerImp . . . . .	26
22	Classe MessageSendType . . . . .	29
23	Interfaccia NavigationManager . . . . .	30
24	Classe NavigationManagerImp . . . . .	32
25	Classe NoBeaconSeenException . . . . .	35
26	Classe ServiceConnectionImp . . . . .	36
27	Classe BeaconManagerAdapter . . . . .	37
28	Interfaccia BeaconRanger . . . . .	40
29	Classe LocalBinder . . . . .	41
30	Interfaccia Logger . . . . .	42
31	Classe LoggerImp . . . . .	44
32	Interfaccia MyBeacon . . . . .	46
33	Classe MyBeaconImp . . . . .	48
34	Classe MyDistanceCalculator . . . . .	50
35	Classe Compass . . . . .	51
36	Classe BuildingContract . . . . .	53
37	Interfaccia BuildingDao . . . . .	55
38	Classe BuildingTable . . . . .	57
39	Classe CategoryContract . . . . .	59

40	Interfaccia CategoryDao . . . . .	60
41	Classe CategoryTable . . . . .	62
42	Interfaccia CursorConverter . . . . .	63
43	Classe DaoFactoryHelper . . . . .	64
44	Classe EdgeContract . . . . .	65
45	Interfaccia EdgeDao . . . . .	66
46	Classe EdgeTable . . . . .	68
47	Classe EdgeTypeContract . . . . .	70
48	Interfaccia EdgeTypeDao . . . . .	71
49	Classe EdgeTypeTable . . . . .	73
50	Classe MapsDbContract . . . . .	74
51	Classe MapsDbHelper . . . . .	75
52	Classe PhotoContract . . . . .	77
53	Interfaccia PhotoDao . . . . .	78
54	Classe PhotoTable . . . . .	80
55	Classe PointOfInterestContract . . . . .	81
56	Interfaccia PointOfInterestDao . . . . .	82
57	Classe PointOfInterestTable . . . . .	84
58	Classe RegionOfInterestContract . . . . .	86
59	Interfaccia RegionOfInterestDao . . . . .	87
60	Classe RegionOfInterestTable . . . . .	89
61	Classe RemoteBuildingDao . . . . .	90
62	Classe RemoteCategoryDao . . . . .	91
63	Classe RemoteDaoFactory . . . . .	93
64	Classe RemoteEdgeDao . . . . .	94
65	Classe RemoteEdgeTypeDao . . . . .	95
66	Classe RemotePhotoDao . . . . .	96
67	Classe RemotePointOfInterestDao . . . . .	97
68	Classe RemoteRegionOfInterestDao . . . . .	98
69	Classe RemoteRoiPoiDao . . . . .	99
70	Classe RoiPoiContract . . . . .	100
71	Interfaccia RoiPoiDao . . . . .	101
72	Classe RoiPoiTable . . . . .	103
73	Classe SQLDao . . . . .	104
74	Classe SQLiteBuildingDao . . . . .	107
75	Classe SQLiteCategoryDao . . . . .	110
76	Classe SQLiteDaoFactory . . . . .	112
77	Classe SQLiteEdgeDao . . . . .	114
78	Classe SQLiteEdgeTypeDao . . . . .	116
79	Classe SQLitePhotoDao . . . . .	118
80	Classe SQLitePointOfInterestDao . . . . .	120

81	Classe SQLiteRegionOfInterestDao . . . . .	122
82	Classe SQLiteRoiPoiDao . . . . .	125
83	Classe BuildingService . . . . .	127
84	Interfaccia DatabaseService . . . . .	131
85	Classe EdgeService . . . . .	133
86	Classe PhotoService . . . . .	136
87	Classe PointOfInterestService . . . . .	138
88	Classe RegionOfInterestService . . . . .	141
89	Classe ServiceHelper . . . . .	143
90	Classe BuildingInformation . . . . .	144
91	Interfaccia BuildingMap . . . . .	146
92	Classe BuildingMapImp . . . . .	148
93	Classe astratta NavigationExceptions . . . . .	152
94	Interfaccia Navigator . . . . .	153
95	Classe NavigatorImp . . . . .	155
96	Classe NoGraphSetException . . . . .	158
97	Classe NoNavigationInformationException . . . . .	159
98	Classe PathException . . . . .	161
99	Interfaccia ProcessedInformation . . . . .	162
100	Classe ProcessedInformationImp . . . . .	163
101	Classe DijkstraPathFinder . . . . .	165
102	Interfaccia PathFinder . . . . .	166
103	Classe MapGraph . . . . .	167
104	Interfaccia PointOfInterest . . . . .	169
105	Classe PointOfInterestImp . . . . .	170
106	Classe PointOfInterestInformation . . . . .	172
107	Interfaccia RegionOfInterest . . . . .	174
108	Classe RegionOfInterestImp . . . . .	176
109	Classe astratta AbsEnrichedEdge . . . . .	178
110	Classe DefaultEdge . . . . .	181
111	Interfaccia Edge . . . . .	182
112	Classe ElevatorEdge . . . . .	183
113	Interfaccia EnrichedEdge . . . . .	185
114	Classe StairEdge . . . . .	186
115	Classe BasicInformation . . . . .	188
116	Classe DetailedInformation . . . . .	189
117	Interfaccia NavigationInformation . . . . .	190
118	Classe NavigationInformationImp . . . . .	191
119	Classe PhotoInformation . . . . .	192
120	Classe PhotoRef . . . . .	193
121	Interfaccia Vertex . . . . .	194

122	Classe VertexImp . . . . .	195
123	Classe DeveloperCodeManager . . . . .	196
124	Classe InstructionPreference . . . . .	197
125	Classe PathPreference . . . . .	197
126	Interfaccia Setting . . . . .	198
127	Classe SettingImp . . . . .	200
128	Classe Checker . . . . .	202
129	Classe DatabaseServiceManager . . . . .	203
130	Classe DetailedInformationActivity . . . . .	204
131	Classe DeveloperUnlockerActivity . . . . .	205
132	Classe HelpActivity . . . . .	206
133	Classe HomeActivity . . . . .	208
134	Classe ImageAdapter . . . . .	210
135	Classe ImageDetailActivity . . . . .	212
136	Classe ImageListFragment . . . . .	213
137	Classe ImagePageAdapter . . . . .	215
138	Classe InformationManagerPresenter . . . . .	216
139	Classe LocalMapActivity . . . . .	217
140	Classe LoggingActivity . . . . .	218
141	Classe LogInformationActivity . . . . .	219
142	Classe MainActivity . . . . .	220
143	Classe MainDeveloperActivity . . . . .	221
144	Classe MainDeveloperPresenter . . . . .	223
145	Classe MapDownloaderActivity . . . . .	224
146	Classe NavigationActivity . . . . .	225
147	Classe NavigationAdapter . . . . .	226
148	Classe NavigationManagerPresenter . . . . .	228
149	Classe NearbyPoiActivity . . . . .	229
150	Classe PoiCategoryActivity . . . . .	230
151	Classe PreferencesActivity . . . . .	231
152	Classe RemoteMapManagerActivity . . . . .	232
153	Classe SearchSuggestionsProvider . . . . .	233
154	Classe SettingManager . . . . .	235
155	Interfaccia DetailedInformationView . . . . .	236
156	Classe DetailedInformationViewImp . . . . .	237
157	Interfaccia DeveloperUnlockerView . . . . .	238
158	Classe DeveloperUnlockerViewImp . . . . .	239
159	Interfaccia HelpView . . . . .	241
160	Classe HelpViewImp . . . . .	242
161	Interfaccia HomeView . . . . .	243
162	Classe HomeViewImp . . . . .	245

163	Interfaccia ImageDetailView . . . . .	248
164	Classe ImageDetailViewImp . . . . .	249
165	Interfaccia ImageListFragmentView . . . . .	250
166	Classe ImageListFragmentViewImp . . . . .	251
167	Interfaccia LocalMapManagerView . . . . .	252
168	Classe LocalMapManagerViewImp . . . . .	253
169	Interfaccia LoggingView . . . . .	254
170	Classe LoggingViewImp . . . . .	255
171	Interfaccia LogInformationView . . . . .	256
172	Classe LogInformationViewImp . . . . .	257
173	Interfaccia MainDeveloperView . . . . .	259
174	Classe MainDeveloperViewImp . . . . .	260
175	Interfaccia MainView . . . . .	261
176	Classe MainViewImp . . . . .	262
177	Interfaccia MapDownloaderView . . . . .	263
178	Classe MapDownloaderViewImp . . . . .	264
179	Interfaccia NavigationView . . . . .	265
180	Classe NavigationViewImp . . . . .	266
181	Interfaccia NearbyPoiView . . . . .	268
182	Classe NearbyPoiViewImp . . . . .	269
183	Interfaccia PoiCategoryView . . . . .	270
184	Classe PoiCategoryViewImp . . . . .	271
185	Interfaccia PreferencesView . . . . .	272
186	Classe PreferencesViewImp . . . . .	273
187	Interfaccia RemoteMapManagerView . . . . .	275
188	Classe RemoteMapManagerViewImp . . . . .	276
189	Schema UML - base di dati . . . . .	278
190	Diagramma di sequenza - Avvio di un service, per il rilevamento beacon . . . . .	280
191	Diagramma di sequenza - Elaborazione beacon rilevati e comunicazione broadcast . . . . .	282
192	Diagramma di sequenza - Avvio navigazione . . . . .	283
193	Diagramma di sequenza - Avvio della bussola . . . . .	284
194	Diagramma delle classi - model . . . . .	337
195	Diagramma delle classi - model::navigator . . . . .	338
196	Diagramma delle classi - model::dataaccess . . . . .	339
197	Diagramma delle classi - presenter . . . . .	340

## 1 Introduzione

### 1.1 Scopo del documento

Questo documento definisce nel dettaglio la struttura e le relazioni tra le parti del prodotto<sub>g</sub>, approfondendo ulteriormente dove ritenuto necessario. In particolare vengono descritti in dettaglio i package, le classi e le interfacce, concludendo con il tracciamento tra le classi e i requisiti analizzati nell'*Analisi dei requisiti v4.00*.

### 1.2 Scopo del prodotto

Lo scopo del prodotto<sub>g</sub> è implementare un metodo di navigazione indoor<sub>g</sub> che sia funzionale alla tecnologia Bluetooth Low Energy (BLE<sub>g</sub>). Il prodotto<sub>g</sub> comprenderà un prototipo software<sub>g</sub> che permetta la navigazione all'interno di un'area predefinita, basandosi sui concetti di Indoor Positioning System (IPS<sub>g</sub>) e smart place<sub>g</sub>.

### 1.3 Glossario

Allo scopo di rendere più semplice e chiara la comprensione dei documenti viene allegato il *Glossario v4.00* nel quale verranno raccolte le spiegazioni di terminologia tecnica o ambigua, abbreviazioni ed acronimi. Per evidenziare un termine presente in tale documento, esso verrà marcato con il pedice <sub>g</sub>.

### 1.4 Riferimenti utili

#### 1.4.1 Riferimenti normativi

- capitolato d'appalto C2: CLIPS<sub>g</sub> : Communication & Localization with Indoor Positioning Systems: <http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C2.pdf>;
- *Norme di progetto v4.00*.

#### 1.4.2 Riferimenti informativi

- Documentazione Android SDK: <http://developer.android.com/guide/index.html>;
- Documentazione AltBeacon Library: <https://altbeacon.github.io/android-beacon-library/documentation.html>;

- Documentazione SQLite: <https://www.sqlite.org/docs.html>;
- Documentazione JavaDoc JGraphT Library: <http://jgrapht.org/javadoc/>;
- Materiale di riferimento del corso di Ingegneria del Software<sub>g</sub> - Diagrammi delle classi: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E03.pdf>;
- Materiale di riferimento del corso di Ingegneria del Software<sub>g</sub> - Model View Presenter: [http://www.math.unipd.it/~rcardin/sweb/Design%20Pattern%20Architetturali%20-%20Model%20View%20Controller\\_4x4.pdf](http://www.math.unipd.it/~rcardin/sweb/Design%20Pattern%20Architetturali%20-%20Model%20View%20Controller_4x4.pdf);
- Materiale di riferimento del corso di Ingegneria del Software<sub>g</sub> - Layer Architecture: [http://www.math.unipd.it/~rcardin/sweb/Software%20Architecture%20Patterns\\_4x4.pdf](http://www.math.unipd.it/~rcardin/sweb/Software%20Architecture%20Patterns_4x4.pdf)
- Design Pattern: elementi per il riuso di software ad oggetti - Gamma, Helm, Johnson, Vlissides - editore Pearson - 2002: Part 1, capitoli: 5 - System modeling, 6 - Architectural design & 7 - Design and implementation;
- UML e ingegneria del software: dalla teoria alla pratica - Luca Vetti Tagliati - 2015: capitoli: 7 - Gli oggetti: una questione di classe & 9 - Diagrammi di interazione.

## 2 Standard di progetto

### 2.1 Standard di documentazione del codice

Per gli standard di documentazione del codice si fa riferimento al documento *Norme di progetto v4.00*.

### 2.2 Standard di denominazione di entità e relazioni

Per tutte le entità e le relazioni valgono gli standard di denominazione seguenti:

- per le entità definite come package, classi, attributi e metodi è necessario fornire denominazioni chiare e concise;
- per la denominazione delle entità sono da preferire i sostantivi mentre per le relazioni i verbi;
- eventuali abbreviazioni sono preferibilmente da evitare nonostante siano ammesse nei casi in cui siano comprensibili e non ambigue.
- per le regole tipografiche sui nomi si fa riferimento al documento *Norme di progetto v4.00*.

### 2.3 Standard di programmazione

Per gli standard di programmazione si fa riferimento al documento *Norme di progetto v4.00*.

### 2.4 Strumenti di lavoro e procedure

Per gli strumenti di lavoro e le procedure per la realizzazione del progetto si fa riferimento al documento *Norme di progetto v4.00*.

### 3 Architettura applicazione

#### 3.1 Architettura ad alto livello

L’architettura dell’applicazione rispecchia il pattern architettonico a livelli presentando quattro livelli:

**Presentation layer:** contiene le componenti dell’interfaccia grafica utente passiva e le componenti facenti parte dell’interfaccia di comunicazione tra vista e il layer sottostante;

**Business layer:** contiene le componenti logiche del model tranne quelle adibite all’accesso al database;

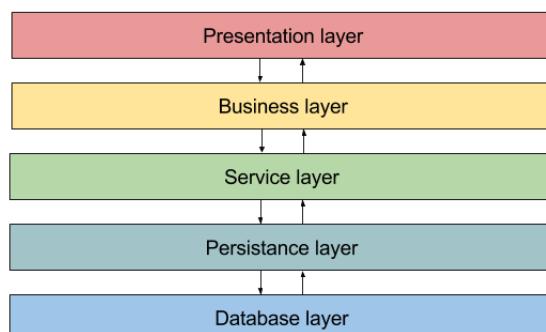
**Service layer:** contiene le componenti adibite a interfacciare le componenti della logica dell’applicativo con quelle che si interfacciano direttamente al database;

**Persistance layer:** contiene le componenti che comunicano direttamente con il database e vengono utilizzate dal Service layer;

**Database layer:** corrisponde al database SQLite all’interno del dispositivo mobile.

La scelta di tale pattern architettonico deriva dai vantaggi che ne derivano:

- Garantisce un ambiente facile da testare;
- Facilita lo sviluppo grazie alla sua semplicità teorica;
- Disaccoppiamento delle componenti con diversi scopi;



**Figura 1:** Architettura a layer adottata nell’applicazione

## 4 Specifica dei componenti

### 4.1 Metodo e formalismo di specifica

L'esposizione dell'architettura in dettaglio dell'applicazione è esposta di seguito seguendo un approccio top-down a livelli. Si descrive quindi l'architettura partendo dal generale esponendo inizialmente le componenti più teoriche: i package fino a quelle più concrete: le classi con i relativi metodi, attributi e relazioni di ereditarietà. Per distinguere in modo immediato le componenti di librerie dai componenti dell'applicativo si è deciso di associare ciascuna libreria ad un colore specifico:

- Android SDK: classi rappresentati in verde;
- JGraphT: classi rappresentate in grigio;
- AltBeacon: classi rappresentate in arancione;
- Java API: classi rappresentate in azzurro.

Mentre le classi dell'applicativo sono rappresentate nel classico giallo.

Per ogni package si specifica:

- il nome;
- una descrizione;
- il package da cui discende;
- le interazioni con gli altri package;
- gli eventuali package contenuti;
- le classi contenute affiancate da un riferimento alla descrizione completa.

Per ogni classe si specifica:

- il nome;
- il tipo;
- l'eventuale classe che estende;
- le eventuali interfacce che implementa;
- la visibilità;

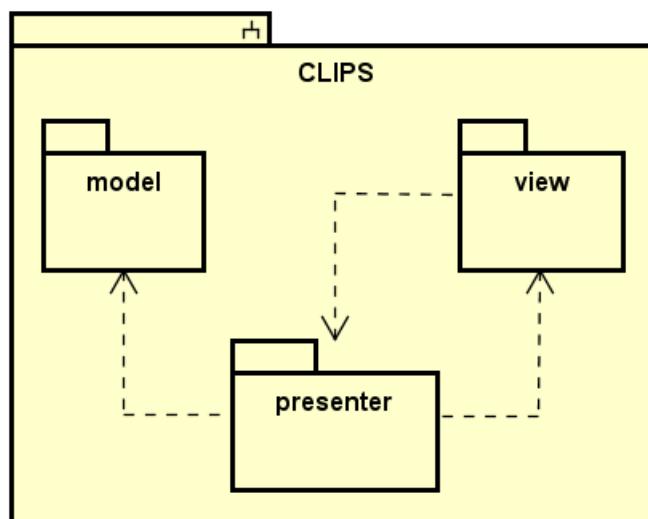
- una descrizione;
- la lista dettagliata degli attributi;
- la lista dettagliata dei metodi.

Per i diagrammi dei package e delle classi si utilizza il formalismo *UML 2.0*.

## 4.2 Sistema CLIPS

L'architettura dell'applicativo è basata sul pattern Model View Presenter MVP, in questo modo si preserva il mantenimento del componente model se la view cambiasse e viceversa. I package fondamentali sono:

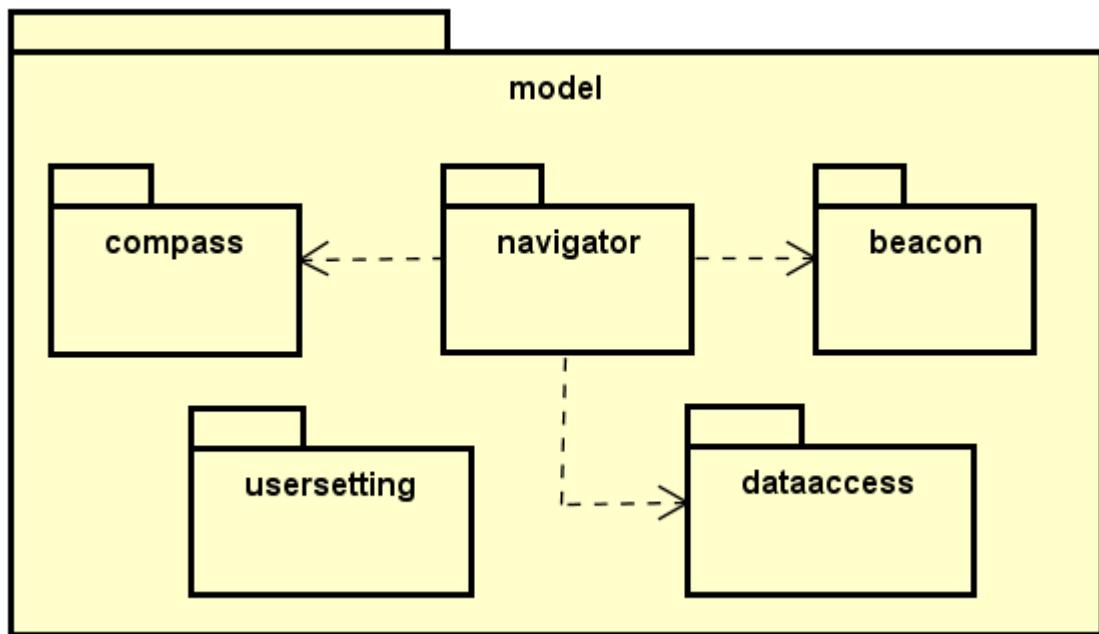
- **model**: contiene tutta la business logic dell'applicativo;
- **view**: contiene una serie di classi "passive" ossia assenti di logica e con relazioni minime tra di esse;
- **presenter**: contiene la logica che permette la comunicazione tra **view** e **model**, aggiorna la **view** ed elaborazione i segnali provenienti da essa.



**Figura 2:** Diagramma dei package - sistema CLIPS

## 4.3 Componenti

### 4.3.1 model



**Figura 3:** Componente model

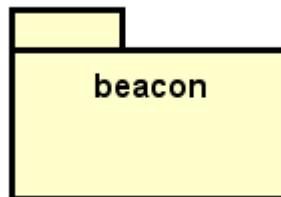
- **Descrizione:** Package per il componente Model del pattern architetture MVP. Questo pacchetto contiene tutte le classi che compongono la business logic;
- **Package Contenuti:**
  - beacon;
  - compass;
  - dataaccess;
  - navigator;
  - usersetting.
- **Classi Contenute:**
  - AbsBeaconReceiverManager;
  - InformationManagerImp;

- MessageSendType;
- NavigationManagerImp;
- NoBeaconSeenException;
- ServiceConnectionImp.

- **Interfacce Contenute:**

- InformationManager;
- NavigationManager.

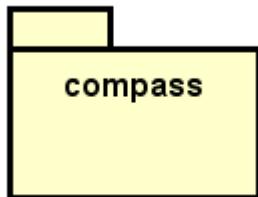
#### 4.3.2 model::beacon



**Figura 4:** Componente model::beacon

- **Descrizione:** Package contenente le classi che rappresentano o si occupano della rilevazione dei beacon. Questo package ha inoltre il compito di interfacciarsi con la libreria Altbeacon;
- **Padre:** model;
- **Classi Contenute:**
  - BeaconManagerAdapter;
  - LocalBinder;
  - LoggerImp;
  - MyBeaconImp;
  - MyDistanceCalculator.
- **Interfacce Contenute:**
  - BeaconRanger;
  - Logger;
  - MyBeacon.

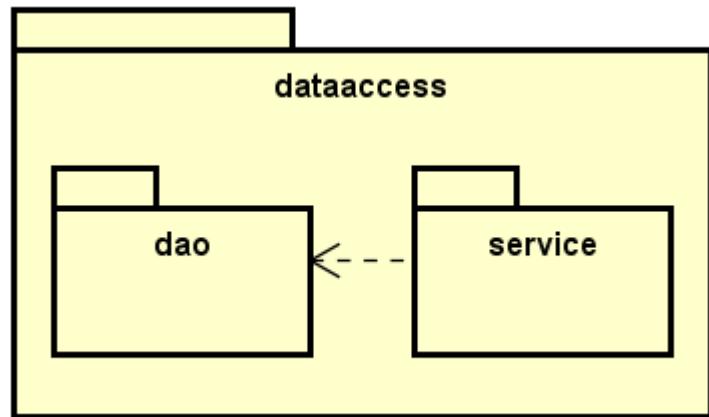
#### 4.3.3 model::compass



**Figura 5:** Componente model::compass

- **Descrizione:** Package per la gestione della bussola;
- **Padre:** model;
- **Classi Contenute:**
  - Compass.

#### 4.3.4 model::dataaccess

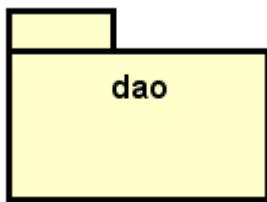


**Figura 6:** Componente model::dataaccess

- **Descrizione:** Package per la gestione dell'accesso ai dati del database locale e remoto;
- **Padre:** model;
- **Package Contenuti:**

- dao;
- service.

#### 4.3.5 model::dataaccess::dao



**Figura 7:** Componente model::dataaccess::dao

- **Descrizione:** Package che permette l'interazione diretta con il database e di costruire oggetti persistenti a partire dai risultati delle query sul database;
- **Padre:** dataaccess;
- **Classi Contenute:**
  - BuildingContract;
  - BuildingTable;
  - CategoryContract;
  - CategoryTable;
  - DaoFactoryHelper;
  - EdgeContract;
  - EdgeTable;
  - EdgeTypeContract;
  - EdgeTypeTable;
  - MapsDbContract;
  - MapsDbHelper;
  - PhotoContract;
  - PhotoTable;
  - PointOfInterestContract;

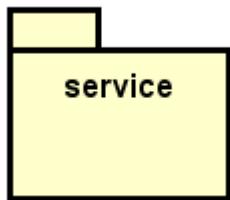
- PointOfInterestTable;
- RegionOfInterestContract;
- RegionOfInterestTable;
- RemoteBuildingDao;
- RemoteCategoryDao;
- RemoteDaoFactory;
- RemoteEdgeDao;
- RemoteEdgeTypeDao;
- RemotePhotoDao;
- RemotePointOfInterestDao;
- RemoteRegionOfInterestDao;
- RemoteRoiPoiDao;
- RoiPoiContract;
- RoiPoiTable;
- SQLDao;
- SQLiteBuildingDao;
- SQLiteCategoryDao;
- SQLiteDaoFactory;
- SQLiteEdgeDao;
- SQLiteEdgeTypeDao;
- SQLitePhotoDao;
- SQLitePointOfInterestDao;
- SQLiteRegionOfInterestDao;
- SQLiteRoiPoiDao.

- **Interfacce Contenute:**

- BuildingDao;
- CategoryDao;
- CursorConverter;
- EdgeDao;
- EdgeTypeDao;

- PhotoDao;
- PointOfInterestDao;
- RegionOfInterestDao;
- RoiPoiDao.

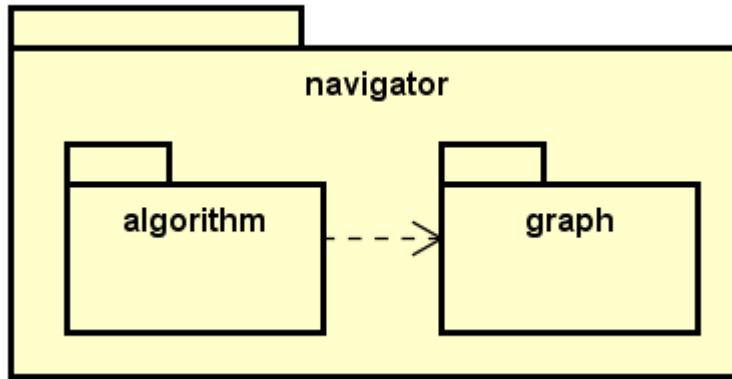
#### 4.3.6 model::dataaccess::service



**Figura 8:** Componente model::dataaccess::service

- **Descrizione:** Package per la creazione degli oggetti della business logic a partire da oggetti DAO;
- **Padre:** dataaccess;
- **Classi Contenute:**
  - BuildingService;
  - EdgeService;
  - PhotoService;
  - PointOfInterestService;
  - RegionOfInterestService;
  - ServiceHelper.
- **Interfacce Contenute:**
  - DatabaseService.

#### 4.3.7 model::navigator

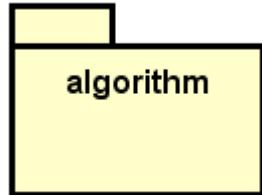


**Figura 9:** Componente model::navigator

- **Descrizione:** Package contenente le classi che permettono la navigazione all'interno degli edifici per cui è previsto il servizio e di accedere alle informazioni relative a tali edifici;
- **Padre:** model;
- **Package Contenuti:**
  - algorithm;
  - graph.
- **Classi Contenute:**
  - BuildingInformation;
  - BuildingMapImp;
  - NavigationExceptions;
  - NavigatorImp;
  - NoGraphSetException;
  - NoNavigationInformationException;
  - PathException;
  - ProcessedInformationImp.
- **Interfacce Contenute:**

- BuildingMap;
- Navigator;
- ProcessedInformation.

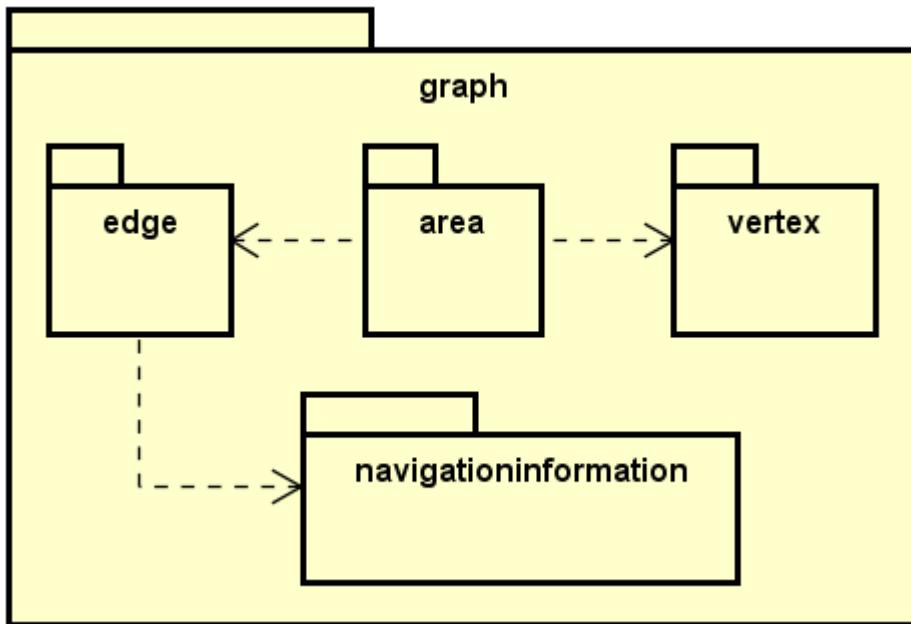
#### 4.3.8 model::navigator::algorithm



**Figura 10:** Componente model::navigator::algorithm

- **Descrizione:** Package contenente le classi che si occupano del calcolo dei percorsi da seguire per la navigazione;
- **Padre:** navigator;
- **Classi Contenute:**
  - DijkstraPathFinder.
- **Interfacce Contenute:**
  - PathFinder.

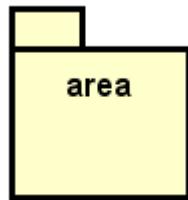
#### 4.3.9 model::navigator::graph



**Figura 11:** Componente `model::navigator::graph`

- **Descrizione:** Package contenente le classi che permettono la rappresentazione di un edificio sottoforma di grafo;
- **Padre:** `navigator`;
- **Package Contenuti:**
  - `area`;
  - `edge`;
  - `navigationinformation`;
  - `vertex`.
- **Classi Contenute:**
  - `MapGraph`.

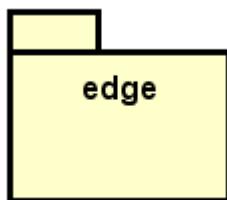
#### 4.3.10 model::navigator::graph::area



**Figura 12:** Componente model::navigator::graph::area

- **Descrizione:** Package contenente le classi per rappresentare le aree interne di un edificio;
- **Padre:** graph;
- **Interazione con componenti:**
  - model::navigator::graph::vertex
- **Classi Contenute:**
  - PointOfInterestImp;
  - PointOfInterestInformation;
  - RegionOfInterestImp.
- **Interfacce Contenute:**
  - PointOfInterest;
  - RegionOfInterest.

#### 4.3.11 model::navigator::graph::edge



**Figura 13:** Componente model::navigator::graph::edge

- **Descrizione:** Package contenenti le classi per la rappresentazione di un arco di un grafo. Questo package contiene inoltre le classi per rappresentare degli archi che contengono informazione;
- **Padre:** graph;
- **Classi Contenute:**
  - AbsEnrichedEdge;
  - DefaultEdge;
  - ElevatorEdge;
  - StairEdge.
- **Interfacce Contenute:**
  - Edge;
  - EnrichedEdge.

#### 4.3.12 model::navigator::graph::navigationinformation

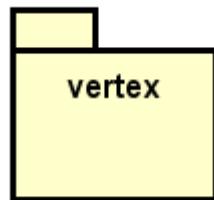


**Figura 14:** Componente model::navigator::graph::navigationinformation

- **Descrizione:** Package contenente le classi per la rappresentazione delle informazioni di navigazione;
- **Padre:** graph;
- **Classi Contenute:**
  - BasicInformation;
  - DetailedInformation;
  - NavigationInformationImp;
  - PhotoInformation;

- PhotoRef.
- **Interfacce Contenute:**
  - NavigationInformation.

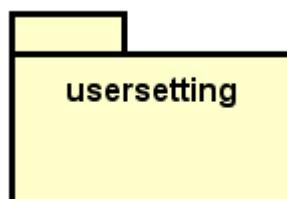
#### 4.3.13 model::navigator::graph::vertex



**Figura 15:** Componente model::navigator::graph::vertex

- **Descrizione:** Package contenente le classi per la rappresentazione di un vertice del grafo;
- **Padre:** graph;
- **Interazione con componenti:**
  - model::navigator::graph::area
- **Classi Contenute:**
  - VertexImp.
- **Interfacce Contenute:**
  - Vertex.

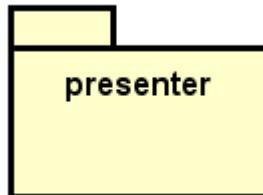
#### 4.3.14 model::usersetting



**Figura 16:** Componente model::usersetting

- **Descrizione:** Package contenente le classi che si occupano della gestione delle impostazioni e delle preferenze dell'utente. In particolare si occupano della gestione delle preferenze di navigazione e di fruizione delle informazioni e, inoltre, della gestione dei codici sviluppatore;
- **Padre:** model;
- **Classi Contenute:**
  - DeveloperCodeManager;
  - InstructionPreference;
  - PathPreference;
  - SettingImp.
- **Interfacce Contenute:**
  - Setting.

#### 4.3.15 presenter

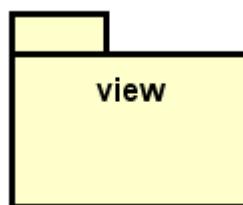


**Figura 17:** Componente presenter

- **Descrizione:** Package per il componente Presenter del pattern architetture MVP;
- **Classi Contenute:**
  - Checker;
  - DatabaseServiceManager;
  - DetailedInformationActivity;
  - DeveloperUnlockerActivity;
  - HelpActivity;
  - HomeActivity;

- ImageAdapter;
- ImageDetailActivity;
- ImageListFragment;
- ImagePageAdapter;
- InformationManagerPresenter;
- LocalMapActivity;
- LoggingActivity;
- LogInformationActivity;
- MainActivity;
- MainDeveloperActivity;
- MainDeveloperPresenter;
- MapDownloaderActivity;
- NavigationActivity;
- NavigationAdapter;
- NavigationManagerPresenter;
- NearbyPoiActivity;
- PoiCategoryActivity;
- PreferencesActivity;
- RemoteMapManagerActivity;
- SearchSuggestionsProvider;
- SettingManager.

#### 4.3.16 view



**Figura 18:** Componente view

- **Descrizione:** Package per il componente View del pattern architetturel MVP. Questo pacchetto contiene tutte le classi che compongono l'application logic;

- **Classi Contenute:**

- DetailedInformationViewImp;
- DeveloperUnlockerViewImp;
- HelpViewImp;
- HomeViewImp;
- ImageDetailViewImp;
- ImageListFragmentViewImp;
- LocalMapManagerViewImp;
- LoggingViewImp;
- LogInformationViewImp;
- MainDeveloperViewImp;
- MainViewImp;
- MapDownloaderViewImp;
- NavigationViewImp;
- NearbyPoiViewImp;
- PoiCategoryViewImp;
- PreferencesViewImp;
- RemoteMapManagerViewImp.

- **Interfacce Contenute:**

- DetailedInformationView;
- DeveloperUnlockerView;
- HelpView;
- HomeView;
- ImageDetailView;
- ImageListFragmentView;
- LocalMapManagerView;
- LoggingView;
- LogInformationView;
- MainDeveloperView;
- MainView;

- MapDownloaderView;
- NavigationView;
- NearbyPoiView;
- PoiCategoryView;
- PreferencesView;
- RemoteMapManagerView.

## 4.4 Classi

### 4.4.1 model::AbsBeaconReceiverManager

AbsBeaconReceiverManager
- context : Context {readOnly} - SERVICE_START : Intent - beaconRanger : BeaconRanger - serviceConnection : ServiceConnection
+ AbsBeaconReceiverManager(context : Context) + modifyScanningPeriod(p : long, backOrFore : boolean, betweenOrNot : boolean) : void + startService() : void + stopService() : void + onReceive() : void # getContext() : Context

**Figura 19:** Classe astratta AbsBeaconReceiverManager

**Nome:** *AbsBeaconReceiverManager*;

**Tipo:** Classe astratta;

**Estende:**

- ServiceConnectionImp.

**Componenti delle librerie utilizzate:**

- android.content.BroadcastReceiver (Android);
- android.content.Context (Android);
- android.content.IntentFilter (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per implementare metodi utili a tutte le classi che necessitano di ricevere e utilizzare beacon;

**Descrizione:** Classe base per la comunicazione con le classi che si occupano del rilevamento dei beacon;

#### Attributi:

- - `beaconManagerAdapter` : `BeaconRanger {readOnly}`  
Service che si occupa del rilevamento dei beacon
- - `context` : `Context {readOnly}`  
Contesto dell'applicazione
- - `serviceConnection` : `ServiceConnection`  
Connessione con il Service per la comunicazione con il service stesso
- - `SERVICE_START` : `Intent {readOnly}`  
Intent per ricevere I beacon inviati dal BeaconManagerAdapter

#### Metodi:

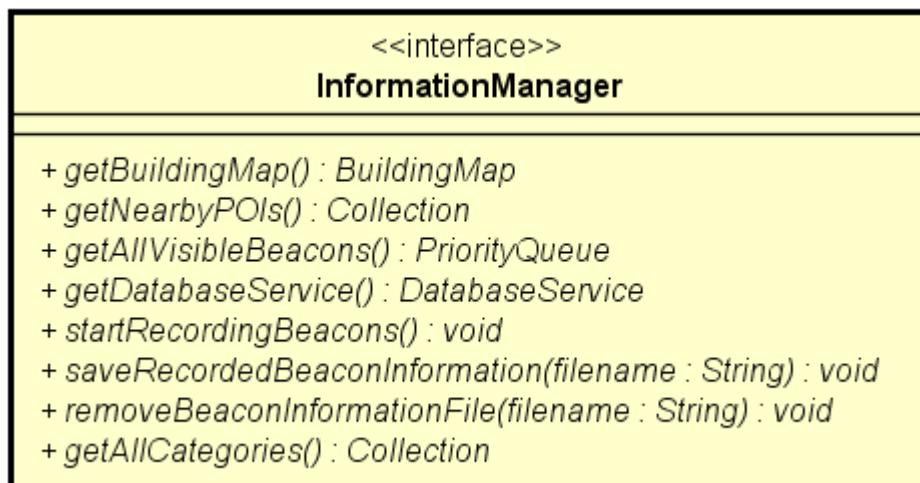
- + `AbsBeaconReceiverManager()`  
Costruttore della classe AbsBeaconReceiverManager
- # `getContext()` : `Context`  
Metodo che ritorna il contesto dell'applicazione
- + `modifyScanningPeriod(p : long, backOrFore : boolean, betweenOrNot : boolean) : void`  
Metodo che permette di modificare il tempo tra una scansione per la ricerca dei beacon e la successiva

#### Argomenti:

- `p` : `long`  
Periodo di scansione dei beacon da scansionare
- `backOrFore` : `boolean`  
Scelta se modificare il periodo di scansione quando l'applicazione è in background o in foreground
- `betweenOrNot` : `boolean`  
Scelta se modificare il periodo di scansione o la pausa tra due scansioni
- + `onReceive() : void`  
**Override** Metodo astratto che permette di eseguire delle azioni alla ricezione di una `PriorityQueue<MyBeacon>` contenente l'insieme dei beacon visibili in un determinato momento

- + `startService() : void`  
Metodo che permette di attivare il service che si occupa di fare le scansioni per trovare I beacon
- + `stopService() : void`  
Metodo che permette di fermare il service che si occupa di fare le scansioni per trovare I beacon

#### 4.4.2 model::InformationManager



**Figura 20:** Interfaccia InformationManager

**Nome:** *InformationManager*;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'accesso alle informazioni trattate dai pacchetti del Model da come questo è realizzato;

**Descrizione:** Interfaccia che si occupa di esporre tutti i metodi utili per accedere ad informazioni trattate dai vari pacchetti del Model;

**Metodi:**

- + `getAllCategories() : Collection<String>`  
Metodo che ritorna tutte le categorie di POI all'interno dell'edificio

- + *getAllVisibleBeacons()* : *PriorityQueue<MyBeacon>*  
Metodo che ritorna la PriorityQueue<MyBeacon>, eventualmente vuota, dei beacon visibili
- + *getBuildingMap()* : *BuildingMap*  
Metodo che ritorna la mappa dell'edificio se questa è già stata caricata dal database locale. Viene lanciata una eccezione di tipo NoBeaconSeenException nel caso in cui non sia stata caricata la mappa poiché non è stato ancora ricevuto alcun beacon
- + *getDatabaseService()* : *DatabaseService*  
Metodo che ritorna un oggetto DatabaseService che permette di interrogare il database
- + *getLogInfo(name : String)* : *String*  
Metodo che, dato il nome di un log, ritorna l'informazione in esso contenuta sotto forma di stringa

**Argomenti:**

- name : String  
Nome del file di log da cui reperire l'informazione

- + *getNearbyPOIs()* : *Collection<PointOfInterest>*  
Metodo che ritorna l'insieme di POI associati al beacon rilevato con il segnale più potente. Viene lanciata una eccezione di tipo NoBeaconSeenException nel caso in cui venga invocato il metodo ma non è stato rilevato ancora alcun beacon
- + *removeBeaconInformationFile(filename : String)* : void  
Metodo che permette di rimuovere un log delle informazioni dei beacon visibili

**Argomenti:**

- filename : String  
Nome del file da rimuovere

- + *saveRecordedBeaconInformation(filename : String)* : void  
Metodo che permette di salvare il log delle informazioni dei beacon visibili su file

**Argomenti:**

- filename : String  
Nome da dare al file da salvare

- + *startRecordingBeacons()* : void  
Metodo che permette di avviare il log delle informazioni dei beacon visibili

#### 4.4.3 model::InformationManagerImp

InformationManagerImp	
- map : BuildingMap	
- lastBeaconsSeen : PriorityQueue	
- activeLog : Logger	
- dbService : DatabaseService	
+ InformationManagerImp(dbService : DatabaseService, context : Context)	
+ getBuildingMap() : BuildingMap	
+ getNearbyPOIs() : Collection	
+ getAllVisibleBeacons() : PriorityQueue	
+ getDatabaseService() : DatabaseService	
+ startRecordingBeacons() : void	
+ saveRecordedBeaconInformation(filename : String) : void	
+ removeBeaconInformationFile(filename : String) : void	
+ onReceive() : void	
- setVisibleBeacon(beacons : PriorityQueue) : void	
- loadMap() : BuildingMapImp	
+ getAllCategories() : Collection	
+ getLogInfo(name : String) : String	

**Figura 21:** Classe InformationManagerImp

**Nome:** InformationManagerImp;

**Tipo:** Classe;

**Estende:**

- AbsBeaconReceiverManager.

**Implementa:**

- InformationManager.

**Visibilità:** public;

**Utilizzo:** È utilizzata per avere un unico punto di accesso alle informazioni del package Model. La classe si occupa di tenere un riferimento alla mappa a cui appartengono i beacon rilevati, un riferimento all'interfaccia che permette di accedere alle informazioni salvate nel database

locale o remoto e ti permette di salvare le informazioni dei beacon rilevati;

**Descrizione:** Classe che permette l'accesso alle informazioni trattate nel package Model;

#### Attributi:

- - `activeLog` : `Logger`  
Logger per la registrazione delle informazioni dei beacon rilevati
- - `dbService` : `DatabaseService {readOnly}`  
Oggetto per la gestione delle mappe nel database locale e per il recupero delle mappe nel database remoto
- - `lastBeaconsSeen` : `PriorityQueue<MyBeacon>`  
PriorityQueue, eventualmente vuota, contenente gli ultimi beacon rilevati
- - `map` : `BuildingMap`  
Mappa dell'edificio di cui sono stati rilevati I beacon

#### Metodi:

- + `getAllCategories()` : `Collection<String>`  
**Override** Metodo che ritorna tutte le categorie di POI presenti all'interno dell'edificio
- + `getAllVisibleBeacons()` : `PriorityQueue<MyBeacon>`  
**Override** Metodo che ritorna la `PriorityQueue<MyBeacon>`, eventualmente vuota, dei beacon visibili
- + `getBuildingMap()` : `BuildingMap`  
**Override** Metodo che ritorna la mappa dell'edificio se questa è già stata caricata dal database locale. Viene lanciata una eccezione di tipo `NoBeaconSeenException` nel caso in cui non sia stata caricata la mappa poiché non è stato ancora ricevuto alcun beacon
- + `getDatabaseService()` : `DatabaseService`  
**Override** Metodo che ritorna un oggetto `DatabaseService` che permette di interrogare il database
- + `getLogInfo(name : String)` : `String`  
**Override** Metodo che, dato il nome di un log, ritorna l'informazione in esso contenuta sotto forma di stringa

#### Argomenti:

– `name : String`

Nome del log da cui reperire l'informazione

- + `getNearbyPOIs() : Collection<PointOfInterest>`

**Override** Metodo che ritorna l'insieme di POI associati al beacon rilevato con il segnale più potente. Viene lanciata una eccezione di tipo NoBeaconSeenException nel caso in cui venga invocato il metodo ma non è stato rilevato ancora alcun beacon

- + `InformationManagerImp(dbService : DatabaseService, context : Context)`

**Override** Costruttore della classe InformationManagerImp

#### Argomenti:

– `dbService : DatabaseService`

Oggetto per la gestione delle mappe nel database locale e per il recupero delle mappe nel database remoto

– `context : Context`

Contesto dell'applicazione

- - `loadMap() : BuildingMap`

**Override** Metodo che permette di recuperare una mappa dal database in base al major dei beacon rilevati

- + `onReceive() : void`

**Override** Metodo che si occupa di settare il campo dati lastBeaconsSeen con la PriorityQueue<MyBeacon> contenente gli ultimi beacon rilevati. Nel caso in cui non sia stata ancora caricata una mappa dal database locale si occupa di caricare la mappa dell'edificio che contiene I beacon rilevati

- + `removeBeaconInformationFile(filename : String) : void`

**Override** Metodo che permette di rimuovere un log delle informazioni dei beacon visibili

#### Argomenti:

– `filename : String`

Nome del file da rimuovere

- + `saveRecordedBeaconInformation(filename : String) : void`

**Override** Metodo che permette di salvare il log delle informazioni dei beacon visibili su file

#### Argomenti:

– `filename : String`

Nome del file in cui salvare le informazioni dei beacon

- - `setVisibleBeacon(beacons : PriorityQueue<MyBeacon>) : void`  
**Override** Metodo che setta il campo dati lastBeaconsSeen  
**Argomenti:**
  - `beacons : PriorityQueue<MyBeacon>`  
Lista dei beacon visibili
- + `startRecordingBeacons() : void`  
**Override** Metodo che permette di avviare il log delle informazioni dei beacon visibili

#### 4.4.4 model::MessageSendType



**Figura 22:** Classe MessageSendType

**Nome:** MessageSendType;

**Tipo:** Classe;

**Visibilità:** public;

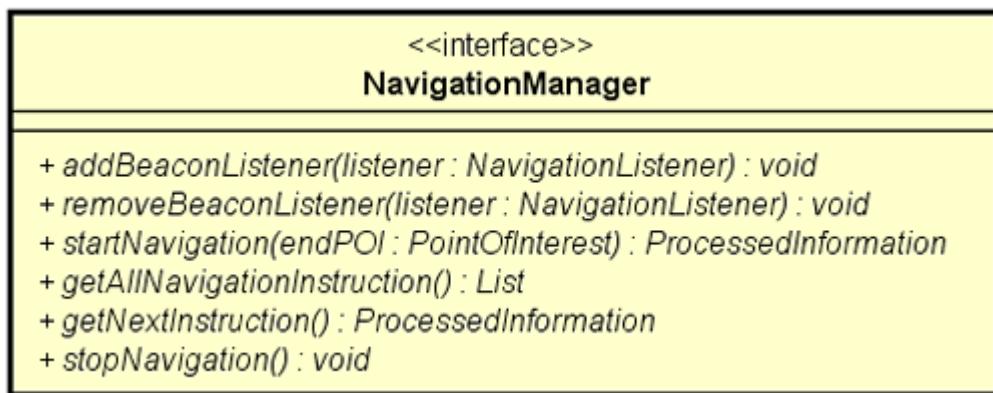
**Utilizzo:** È utilizzata dalla classe BeaconManagerAdapter per inviare messaggi contenenti la lista di beacon tramite una istanza della classe android.support.v4.content.LocalBroadcastManager, attraverso le classi android.content.IntentFilter e BeaconReceiver, per ricevere tali messaggi;

**Descrizione:** Classe che rappresenta l'etichetta di un messaggio scambiato all'interno dell'applicazione contenente una lista di beacon;

**Attributi:**

- - `VISIBLE_BEACON : String {readOnly}`  
Rappresenta il messaggio che viene scambiato all'interno dell'applicazione

#### 4.4.5 model::NavigationManager



**Figura 23:** Interfaccia NavigationManager

**Nome:** *NavigationManager*;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente il modo in cui la navigazione è utilizzata da come questi metodi sono implementati;

**Descrizione:** Interfaccia che si occupa di esporre tutti i metodi utili alla navigazione;

**Metodi:**

- + `addBeaconListener(listener : NavigationListener) : void`  
Metodo che permette di registrare un listener

**Argomenti:**

- `listener : NavigationListener`  
Listener che deve essere aggiunto alla lista di Navigation-Listener

- + `getAllNavigationInstruction() : List<ProcessedInformation>`  
Metodo che permette di recuperare tutte le istruzioni di navigazione per un percorso calcolato. Viene lanciata una eccezione di tipo NoNavigationInformationException nel caso in cui venga richiamato questo metodo senza aver prima avviato la navigazione

- + *getNextInstruction()* : *ProcessedInformation*  
Metodo che permette di recuperare tutte le istruzioni di navigazione per un percorso calcolato in base al beacon più potente ricalcato dalla PriorityQueue<MyBeacon> passata come argomento. Viene lanciata una eccezione di tipo NoNavigationInformationException nel caso in cui venga richiamato questo metodo senza aver prima avviato la navigazione.
- + *removeBeaconListener(listener : NavigationListener) : void*  
Metodo che permette di rimuovere un listener

**Argomenti:**

- *listener* : *NavigationListener*  
Listener che deve essere rimosso dalla lista di NavigationListener

- + *startCompass() : void*  
Metodo che permette di attivare il rilevamento dei dati dalla bussola
- + *startNavigation(endPOI : PointOfInterest) : ProcessedInformation*  
Metodo che permette di avviare la navigazione verso uno specifico POI

**Argomenti:**

- *endPOI* : *PointOfInterest*  
POI da raggiungere tramite navigazione

- + *stopCompass() : void*  
Metodo che permette di fermare il rilevamento dei dati ottenuti dalla bussola
- + *stopNavigation() : void*  
Metodo che permette di fermare la navigazione

#### 4.4.6 model::NavigationManagerImp

NavigationManagerImp	
- navigator : Navigator	
- listeners : Collection	
- graph : MapGraph	
- lastBeaconsSeen : PriorityQueue	
- compass : Compass	
+ NavigationManagerImp(graph : MapGraph, context : Context)	
+ addBeaconListener(listener : NavigationListener) : void	
+ removeBeaconListener(listener : NavigationListener) : void	
+ startNavigation(endPOI : PointOfInterest) : ProcessedInformation	
+ getAllNavigationInstruction() : List	
+ getNextInstruction() : ProcessedInformation	
+ stopNavigation() : void	
+ startCompass() : void	
+ stopCompass() : void	
- setVisibleBeacon(beacons : PriorityQueue) : void	

**Figura 24:** Classe NavigationManagerImp

**Nome:** NavigationManagerImp;

**Tipo:** Classe;

**Estende:**

- AbsBeaconReceiverManager.

**Implementa:**

- NavigationManager.

**Visibilità:** public;

**Utilizzo:** È utilizzata per fornire istruzioni di navigazioni agli oggetti che osservano le istanza di tale classe, in base ai beacon rilevati;

**Descrizione:** Classe che si occupa della gestione della navigazione;

**Attributi:**

- - compass : Compass {readOnly}  
Oggetto che permette di recuperare I dati della bussola
- - graph : MapGraph {readOnly}  
Grafo rappresentante la mappa dell'edificio
- - lastBeaconsSeen : PriorityQueue<MyBeacon>  
PriorityQueue, eventualmente vuota, contenente gli ultimi beacon rilevati
- - listeners : Collection<NavigationListener>  
Collezione contenenti tutti I listener da aggiornare ad ogni nuova istruzione da inviare
- - navigator : Navigator  
Oggetto per la navigazione

**Metodi:**

- + addBeaconListener(listener : NavigationListener) : void  
**Override** Metodo che permette di registrare un listener

**Argomenti:**

- listener : NavigationListener  
Listener che deve essere aggiunto alla lista di NavigationListener

- + getAllNavigationInstruction() : List<ProcessedInformation>  
**Override** Metodo che permette di recuperare tutte le istruzioni di navigazione per un percorso calcolato. Viene lanciata una eccezione di tipo NoNavigationInformationException nel caso in cui venga richiamato questo metodo senza aver prima avviato la navigazione

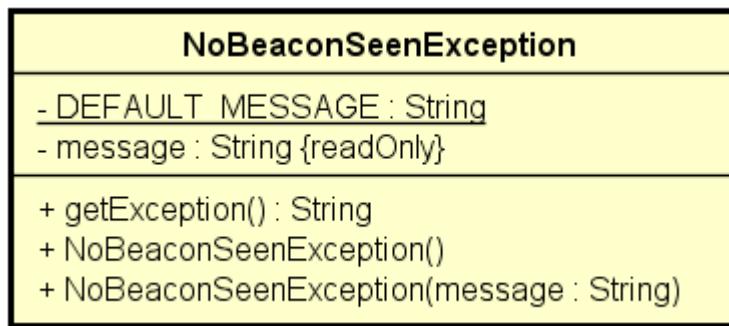
- + getNextInstruction() : ProcessedInformation  
**Override** Metodo che permette di recuperare tutte le istruzioni di navigazione per un percorso calcolato in base al beacon più potente ricavato dalla PriorityQueue<MyBeacon> passata come argomento. Viene lanciata una eccezione di tipo NoNavigationInformationException nel caso in cui venga richiamato questo metodo senza aver prima avviato la navigazione.

- + NavigationManagerImp(graph : MapGraph, context : Context)  
**Override** Costruttore della classe NavigationManagerImp

**Argomenti:**

- **graph** : `MapGraph`  
Grafo dell'edificio in cui si desidera navigare
  - **context** : `Context`  
Contesto dell'applicazione
- + **onReceive()** : `void`  
**Override** Metodo che si occupa di settare il campo dati `lastBeaconsSeen` con la `PriorityQueue<MyBeacon>` contenente gli ultimi beacon rilevati e di aggiornare tutti I listeners con le ultime istruzioni di navigazione
  - + **removeBeaconListener(listener : NavigationListener) : void**  
**Override** Metodo che permette di rimuovere un listener
- Argomenti:**
- **listener** : `NavigationListener`  
Listener che deve essere rimosso dalla lista di `NavigationListener`
- - **setVisibleBeacon(beacons : PriorityQueue<MyBeacon>) : void**  
**Override** Metodo che setta il campo dati `lastBeaconsSeen`
- Argomenti:**
- **beacons** : `PriorityQueue<MyBeacon>`  
Collection di beacon rilevati nell'area circostante
- + **startCompass() : void**  
**Override** Metodo che permette di fermare il rilevamento dei dati ottenuti dalla bussola
  - + **startNavigation() : ProcessedInformation**  
**Override** Metodo che permette di avviare la navigazione verso uno specifico POI
  - + **stopCompass() : void**  
**Override** Metodo che permette di attivare il rilevamento dei dati dalla bussola
  - + **stopNavigation() : void**  
**Override** Metodo che permette di fermare la navigazione

#### 4.4.7 model::NoBeaconSeenException



**Figura 25:** Classe NoBeaconSeenException

**Nome:** NoBeaconSeenException;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Eccezione lanciata nel caso in cui venga richiesta una operazione che coinvolge l'utilizzo dei beacon ma non ne sono stati rilevati;

**Descrizione:** Classe che rappresenta l'eccezione lanciata nel caso in cui non siano rilevati beacon;

**Attributi:**

- - **DEFAULT\_MESSAGE : String**  
Rappresenta il messaggio di default che viene mostrato quando non viene rilevato nessun beacon
- - **message : String {readOnly}**  
Rappresenta un messaggio qualsiasi quando non viene rilevato nessun beacon

**Metodi:**

- + **NoBeaconSeenException()**  
Costruttore della classe di default
- + **NoBeaconSeenException(message : String)**  
Costruttore della classe che richiede un messaggio come parametro

**Argomenti:**

– message : String

Questo parametro richiede che un messaggio di tipo String

#### 4.4.8 model::ServiceConnectionImp

ServiceConnectionImp
+ onServiceConnected(className : ComponentName, service : IBinder) : void
+ onServiceDisconnected(arg0 : ComponentName) : void

**Figura 26:** Classe ServiceConnectionImp

**Nome:** ServiceConnectionImp;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.content.ServiceConnection (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per comunicare con un'istanza della classe BeaconManagerAdapter, per dare la possibilità di ridurre il periodo di scansione per la ricerca dei beacon e per mettere in pausa la scansione qualora l'applicazione venisse messa in background;

**Descrizione:** Classe che implementa android.content.ServiceConnection, utile al fine di comunicare con la classe che si occupa del rilevamento dei beacon;

**Metodi:**

- + onServiceConnected(className : ComponentName, service : IBinder) : void

**Override** Questo metodo permette di specificare determinate azioni nel momento in cui un servizio(Service) viene connesso ad un componente

**Argomenti:**

– className : ComponentName

Questo parametro richiede il nome del servizio su cui si vuole eseguire la connessione

- **service : IBinder**  
Questo parametro richiede l'IBinder del servizio con cui si vuole effettuare la connessione
- + **onServiceDisconnected(className : ComponentName) : void**  
**Override** Questo metodo permette di eseguire delle azioni nel momento in cui un servizio (Service) viene interrotto o termina in seguito ad un errore.

**Argomenti:**

- **className : ComponentName**  
Questo parametro richiede il nome del servizio che è stato interrotto o è terminato in seguito ad errori

**4.4.9 model::beacon::BeaconManagerAdapter**

<b>BeaconManagerAdapter</b>	
- <u>locBinder : IBinder</u>	
- <u>beaconManager : BeaconManager</u>	
- <u>region : Region</u>	
- <u>beaconLayout : String {readOnly}</u>	
+ <u>onCreate() : void</u>	
+ <u>onDestroy() : void</u>	
+ <u>setBackgroundMode(mode : boolean) : void</u>	
- <u>startRanging() : void</u>	
+ <u>modifyScanPeriod(p : long, backOrFore : boolean, betweenOrNot : boolean) : void</u>	
+ <u>setRegionExitPeriod(p : long) : void</u>	
+ <u>onBind(intent : int) : IBinder</u>	
- <u>setMonitorNotifier(notifier : MonitorNotifier) : void</u>	
+ <u>didEnterRegion(region : Region) : void</u>	
+ <u>didExitRegion(region : Region) : void</u>	
+ <u>didDeterminateStateForRegion(i : int, region : Region) : void</u>	

**Figura 27:** Classe BeaconManagerAdapter**Nome:** BeaconManagerAdapter;**Tipo:** Classe;**Implementa:**

- **BeaconRanger.**

**Componenti delle librerie utilizzate:**

- android.app.Service (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per rilevare i beacon;

**Descrizione:** Classe che si occupa del rilevamento dei beacon. Estende la classe android.app.Service e implementa le interfacce org.altbeacon.beacon.BeaconConsumer e org.altbeacon.beacon.startup.BootstrapNotifier;

**Attributi:**

- - beaconLayout : String  
Rappresenta una stringa che identifica la marca del Beacon o del protocollo
- - beaconManager : BeaconManager  
Riferimento alla classe che permette di gestire il rilevamento dei beacon
- - locBinder : IBinder  
Riferimento al LocalBinder che detiene il collegamento con il Service
- - region : Region  
Rappresenta un criterio che serve ad eseguire il match con un beacon

**Metodi:**

- + didDeterminateStateForRegion(i : int , region : Region) : void

**Override** Metodo che determina se un dispositivo è presente all'interno di una Region

**Argomenti:**

- i : int  
Stato della Region che può essere MonitorNotifier.INSIDE o MonitorNotifier.OUTSIDE
- region : Region  
Criterio che serve ad eseguire il match con un beacon

- + didEnterRegion(region : Region) : void

**Override** Metodo che definisce delle azioni da eseguire nel momento in cui il dispositivo rileva uno o più beacon nella Region

**Argomenti:**

– `region : Region`

Criterio che serve ad eseguire il match con un beacon

- + `didExitRegion(region : Region) : void`

**Override** Metodo che definisce delle azioni da eseguire nel momento in cui il dispositivo non rileva più beacon nella Region

**Argomenti:**

– `region : Region`

Criterio che serve ad eseguire il match con un beacon

- + `modifyScanPeriod(p : long, backOrFore : boolean, betweenOrNot : boolean) : void`

Metodo che serve a modificare il periodo di scansione per il rilevamento dei beacon

**Argomenti:**

– `p : long`

Periodo di scansione

– `backOrFore : boolean`

Parametro per decidere se cambiare il periodo di scansione in Foreground o in Background

– `betweenOrNot : boolean`

Parametro che serve a decidere se modificare il periodo di scansione o di non scansione

- + `onBind(intent : Intent) : IBinder`

**Override** Metodo che serve a definire determinate azioni nel momento in cui una classe viene collegata ad un Service

**Argomenti:**

– `intent : Intent`

Intent del Service di cui si vuole fare il collegamento

- + `onCreate() : void`

**Override** Metodo che inizializza i parametri della classe alla creazione di un'istanza

- + `onDestroy() : void`

**Override** Metodo che esegue le azioni necessarie alla distruzione del Service

- + `setBackgroundMode(mode : boolean) : void`

Metodo per notificare al Service che l'applicazione sta andando in background

**Argomenti:**

- mode : boolean

Questo parametro serve per impostare se l'applicazione sta andando in background o no.

- • - setMonitorNotifier() : void

Metodo che imposta il monitor che rileva le notifiche

- • + setRegionExitPeriod(p : long) : void

Metodo per impostare il periodo che determina l'uscita di un beacon da una Region

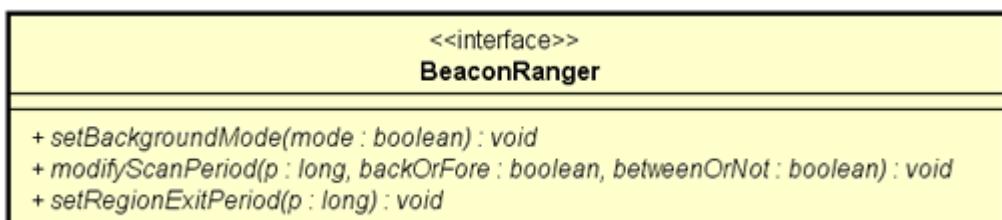
**Argomenti:**

- p : long

Questo parametro richiede il periodo in millisecondi

- • - startRanging() : void

Questo metodo serve per far partire il Ranging dei Beacon

**4.4.10 model::beacon::BeaconRanger**

**Figura 28:** Interfaccia BeaconRanger

**Nome:** BeaconRanger;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata al fine di rendere indipendente il rilevamento dei beacon dalla sua implementazione;

**Descrizione:** Interfaccia che espone tutti i metodi che possono essere invocati su di una classe che si occupa del rilevamento dei beacon ;

**Metodi:**

- + `modifyScanPeriod(p : long, backOrFore : boolean, betweenOrNot : boolean) : void`  
Metodo che serve a modificare il periodo di scansione per il rilevamento dei beacon
- Argomenti:**
  - `p : long`  
Periodo di scansione
  - `backOrFore : boolean`  
Parametro per decidere se cambiare il periodo di scansione in Foreground o in Background
  - `betweenOrNot : boolean`  
Parametro che serve a decidere se modificare il periodo di scansione o di non scansione
- + `setBackgroundMode() : void`  
Metodo per notificare al Service che l'applicazione sta andando in background
- + `setRegionExitPeriod() : void`  
Metodo per impostare il periodo che determina l'uscita di un beacon da una Region

#### 4.4.11 model::beacon::LocalBinder



**Figura 29:** Classe LocalBinder

**Nome:** LocalBinder;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.os.Binder` (Android).

**Visibilità:** public;

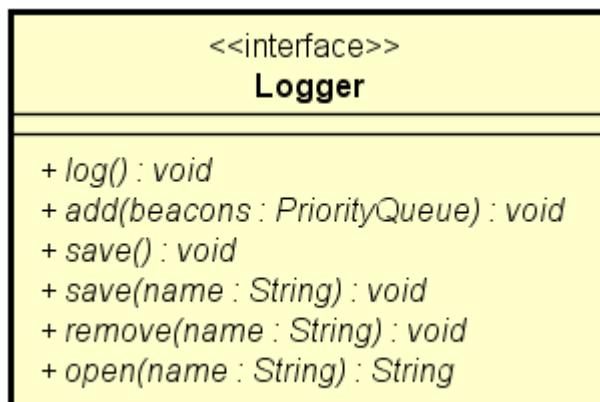
**Utilizzo:** È utilizzata per la comunicazione tra i processi;

**Descrizione:** Classe definita per permettere la comunicazione tra processi (IPC), in questo caso permette di comunicare con i metodi pubblici definiti internamente ad una classe Service. Estende la classe android.os.Binder;

**Metodi:**

- + getService() : BeaconManagerAdapter  
Questo metodo restituisce il riferimento al Service BeaconManagerAdapter

#### 4.4.12 model::beacon::Logger



**Figura 30:** Interfaccia Logger

**Nome:** Logger;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È necessaria per rendere indipendente l'utilizzo di un logger dalla sua implementazione;

**Descrizione:** Interfaccia che espone i metodi utili per accedere alle funzionalità di un logger;

**Metodi:**

- + *add(beacons : PriorityQueue<MyBeacon>) : void*

Metodo che aggiunge all'insieme di informazioni di beacon già eventualmente presenti le informazioni riguardanti i beacon passati in ingresso

**Argomenti:**

- *beacons : PriorityQueue<MyBeacon>*  
Insieme dei beacon di cui salvare le informazioni.

- + *log() : void*

Metodo che salva le informazioni contenute nell'oggetto su di un file

- + *open(name : String) : String*

Metodo che, dato il nome di un file di log, ritorna l'informazione in esso contenuta sotto forma di stringa

**Argomenti:**

- *name : String*  
Nome del file di log da cui reperire le informazioni

- + *remove(name : String) : void*

Metodo per la rimozione di un log precedentemente salvato

**Argomenti:**

- *name : String*  
Nome del log da rimuovere

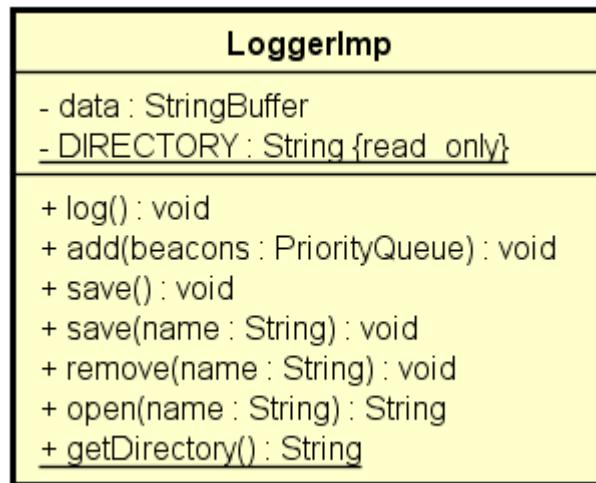
- + *save(filename : String) : void*

Metodo che salva le informazioni contenute nell'oggetto su di un file con nome uguale alla stringa passata come parametro

**Argomenti:**

- *filename : String*  
Nome da dare al file.

#### 4.4.13 model::beacon::LoggerImp



**Figura 31:** Classe LoggerImp

**Nome:** LoggerImp;

**Tipo:** Classe;

**Implementa:**

- Logger.

**Visibilità:** public;

**Utilizzo:** È utilizzata per raccogliere i dati dei beacon che devono essere salvati e per salvare tali dati in un file;

**Descrizione:** Classe che implementa Logger, per la gestione di un log;

**Attributi:**

- - data : StringBuffer  
Rappresenta il contenuto di un log
- - directory : String {readOnly}  
Path della directory in cui vengono salvati i log

**Metodi:**

- + add(beacons : PriorityQueue<MyBeacon>) : void  
**Override** Metodo che aggiunge all'insieme di informazioni di beacon già eventualmente presenti le informazioni riguardanti i beacon passati in ingresso

**Argomenti:**

- beacons : PriorityQueue<MyBeacon>  
Questo parametro richiede una lista di beacons

- + getDirectory() : String  
Metodo che restituisce il path della directory in cui vengono salvati i log

- + log() : void

**Override** Metodo che salva le informazioni contenute nell'oggetto su di un file

- + open(name : String) : String

**Override** Metodo che, dato il nome di un log, ritorna sotto forma di stringa l'informazione in esso contenuta

**Argomenti:**

- name : String  
Nome del log da cui reperire le informazioni

- + remove(name : String) : void

**Override** Metodo per la rimozione di un log precedentemente salvato

**Argomenti:**

- name : String  
Nome del log da rimuovere

- + save(name : String) : void

**Override** Metodo che salva le informazioni contenute nell'oggetto su di un file con nome uguale alla stringa passata come parametro

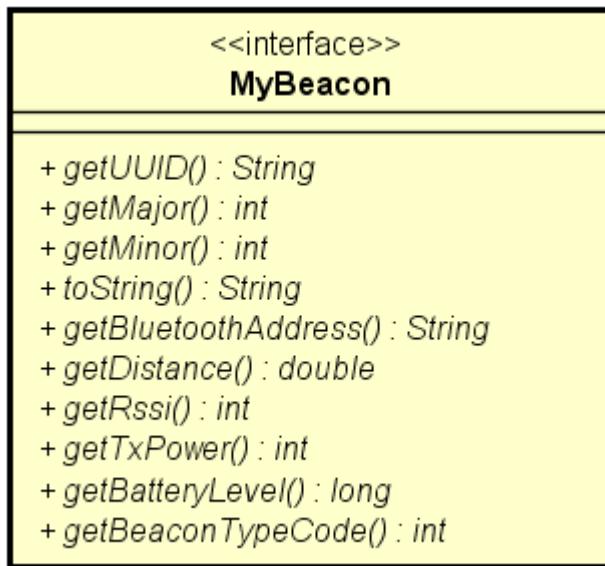
**Argomenti:**

- name : String  
Nome del file sul quale salvare il log

- + save() : void

Metodo che salva le informazioni contenute nell'oggetto su di un file

#### 4.4.14 model::beacon::MyBeacon



**Figura 32:** Interfaccia MyBeacon

**Nome:** MyBeacon;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata al fine di rendere indipendente l'accesso alle informazioni dei beacon da come questi sono rappresentati. È utile inoltre nel caso in cui non si voglia, in futuro, utilizzare più la libreria Altbeacon. Utilizzando questa interfaccia nel Model infatti, in caso di un cambiamento, non sarà necessario cambiare altre parti di model all'interno del package beacon, a meno di un ampliamento delle funzionalità;

**Descrizione:** Interfaccia che espone tutti i metodi che possono essere invocati su di un beacon gestito all'interno della nostra applicazione. Estende l'interfaccia java.io.Serializable;

**Metodi:**

- **+ getBatteryLevel() : long**

Metodo che ritorna il livello di batteria del beacon rilevato

- `+ getBeaconTypeCode() : int`  
Metodo che ritorna il codice rappresentante il tipo di beacon che è stato rilevato
- `+ getBluetoothAddress() : String`  
Metodo che ritorna l'indirizzo Bluetooth del beacon
- `+ getDistance() : double`  
Metodo che ritorna la distanza del beacon dal dispositivo che lo ha rilevato
- `+ getMajor() : int`  
Metodo che ritorna l'identificativo Major del beacon
- `+ getMinor() : int`  
Metodo che ritorna l'identificativo Minor del beacon
- `+ getRssi() : int`  
Metodo che ritorna la potenza ricevuta del segnale del beacon
- `+ getTxPower() : int`  
Metodo che ritorna la potenza di trasmissione del beacon
- `+ getUUID() : String`  
Metodo che ritorna l'identificativo UUID del beacon

#### 4.4.15 model::beacon::MyBeaconImp

MyBeaconImp	
+ CREATOR : Parcelable.Creator {readOnly}	
+ MyBeaconImp(beacon : Beacon)	
+ getDistance() : double	
+ getBluetoothAddress() : String	
+ toString() : String	
+ getRssi() : int	
- recalculateDistance() : void	
+ getTxPower() : int	
+ getUUID() : String	
+ getMajor() : int	
+ getMinor() : int	
+ getBatteryLevel() : long	
+ getBeaconTypeCode() : int	

**Figura 33:** Classe MyBeaconImp

**Nome:** MyBeaconImp;

**Tipo:** Classe;

**Implementa:**

- MyBeacon.

**Componenti delle librerie utilizzate:**

- org.altbeacon.beacon.Beacon (AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per accedere alle informazioni di un beacon sfruttando la classe org.altbeacon.beacon.Beacon, adattandola alle necessità dell'applicazione;

**Descrizione:** Classe che implementa l'interfaccia MyBeacon. Offre tutti i metodi per accedere alle informazioni di un beacon. Estende la classe org.altbeacon.beacon.Beacon;

**Attributi:**

- - `CREATOR : List<NavigationInformationImp, Parcelable.Creator<MyBeaconImp> {readOnly}`  
Attributo richiesto per rendere la classe Parcelable

**Metodi:**

- + `getBatteryLevel() : long`  
**Override** Metodo che ritorna il livello di batteria del beacon rilevato
- + `getBeaconTypeCode() : int`  
**Override** Metodo che ritorna il codice rappresentante il tipo di beacon che è stato rilevato
- + `getBluetoothAddress() : String`  
**Override** Metodo che ritorna l'indirizzo Bluetooth del beacon
- + `getDistance() : double`  
**Override** Metodo che ritorna la distanza del beacon dal dispositivo che lo ha rilevato
- + `getMajor() : int`  
**Override** Metodo che ritorna l'identificativo Major del beacon
- + `getMinor() : int`  
**Override** Metodo che ritorna l'identificativo Minor del beacon
- + `getRssi() : int`  
**Override** Metodo che ritorna la potenza ricevuta del segnale del beacon
- + `getTxPower() : int`  
**Override** Metodo che ritorna la potenza di trasmissione del beacon
- + `getUUID() : String`  
**Override** Metodo che ritorna l'identificativo UUID del beacon
- + `MyBeaconImp(beacon : Beacon)`  
Costruttore della classe

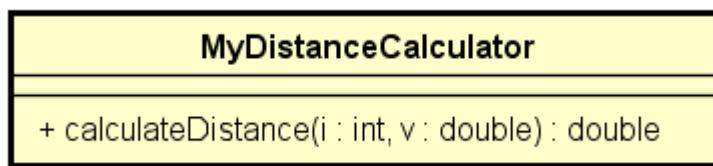
**Argomenti:**

- `beacon : Beacon`  
Questo parametro richiede un beacon
- + `recalculateDistance() : void`  
Questo metodo permette di ricalcolare la distanza tra il beacon e il dispositivo

- + `toString() : String`

Questo metodo permette di fornire una conversione di MyBeaconImp a tipo String

#### 4.4.16 model::beacon::MyDistanceCalculator



**Figura 34:** Classe MyDistanceCalculator

**Nome:** MyDistanceCalculator;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `org.altbeacon.beacon.distance.DistanceCalculator` (AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per calcolare la distanza di un beacon dal dispositivo che lo ho rilevato;

**Descrizione:** Classe che implementa l'interfaccia  
`org.altbeacon.beacon.distance.DistanceCalculator`;

**Metodi:**

- + `calculateDistance(i : int, v : double) : double`  
Questo metodo calcola la distanza di un beacon dal dispositivo

**Argomenti:**

- `i : int`

Questo parametro richiede la potenza del beacon di cui si vuole calcolare la distanza

- `v : double`

Questo parametro richiede il valore rssi del beacon di cui si vuole calcolare la distanza

#### 4.4.17 model::compass::Compass

Compass
- sensorManager : SensorManager - accelerometer : Sensor - magnetometer : Sensor - lastAccelerometerSet : boolean - lastMagnetometerSet : boolean - rotationMatrix : float[0..*] - orientation : float[0..*]
+ Compass() + onSensorChanged(event : SensorEvent) : void + onAccuracyChanged(sensor : Sensor, accuracy : int) : void + unregisterListener() : void + registerListener() : void + getLastCoordinate() : float

**Figura 35:** Classe Compass

**Nome:** Compass;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.hardware.Sensor (Android);
- android.hardware.SensorEventListener (Android);
- android.hardware.SensorManager (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per calcolare e restituire l'orientamento del dispositivo;

**Descrizione:** Classe che si occupa di gestire i dati ricavabili dai sensori e calcolare l'orientamento del device;

**Attributi:**

- - **accelerometer** : **Sensor**  
Sensore che misura l'accelerazione del device sui tre assi fisici
- - **lastAccelerometerSet** : **boolean**  
variabile di guardia per accertarsi il reperimento di almeno un dato dall'accelerometro
- - **lastMagnetometerSet** : **boolean**  
variabile di guardia per accertarsi il reperimento di almeno un dato dal magnetometro
- - **magnetometer** : **Sensor**  
Sensore che misura il campo magnetico per i tre assi fisici
- - **orientation** : **float[]**  
gradi di orientamento sui tre assi fisici
- - **rotationMatrix** : **float[]**  
matrice di rotazione ottenuta dai dati rilevati dai sensori
- - **sensorManager** : **SensorManager**  
oggetto fornito dal sistema Android per ottenere le istanze dei sensori

**Metodi:**

- + **Compass()**  
Costruttore della classe Compass
- + **getLastCoordinate() : float**  
Metodo che restituisce l'ultimo dato calcolato dai dati ricavati dai sensori che indica l'orientamento del dispositivo.
- + **onAccuracyChanged(sensor : Sensor, accuracy : int) : void**  
**Override** Metodo che viene chiamato nel caso in cui l'accuracy dei sensori è cambiata. Attualmente non viene utilizzato dall'applicativo

**Argomenti:**

- **sensor : Sensor**  
Riferimento al sensore che ha scatenato l'evento
  - **accuracy : int**  
Nuova accuratezza impostata al sensore
- + **onSensorChanged(event : SensorEvent) : void**  
**Override** Metodo invocato ad ogni evento generato dai sensori attivi di Compass. Fornisce quindi i dati misurati dal sensore e elaborandoli ne ricava l'orientamento del device

**Argomenti:**

– event : SensorEvent

Rappresenta un evento scatenato da un sensore del dispositivo e detiene al suo interno tutti i dati rilevati da quel sensore

- + registerListener() : void

Metodo che permette all'oggetto Compass di ricevere dati dai sensori e quindi accenderli

- + unregisterListener() : void

Metodo che permette all'oggetto Compass di smettere di ricevere dati dai sensori e quindi spegnerli

#### 4.4.18 model::dataaccess::dao::BuildingContract

<b>BuildingContract</b>	
+ TABLE_NAME : String = "Building"	
+ COLUMN_ID : String = "id"	
+ COLUMN_UUID : String = "uuid"	
+ COLUMN_NAME : String = "name"	
+ COLUMN_DESCRIPTION : String = "description"	
+ COLUMN_OPENINGHOURS : String = "openingHours"	
+ COLUMN_ADDRESS : String = "address"	
+ COLUMN_MAPVERSION : String = "mapVersion"	
+ COLUMN_MAPSIZE : String = "mapSize"	
+ BUILDING_TABLE_CREATE : String = indef	

**Figura 36:** Classe BuildingContract

**Nome:** BuildingContract;

**Tipo:** Classe;

**Visibilità:** public;

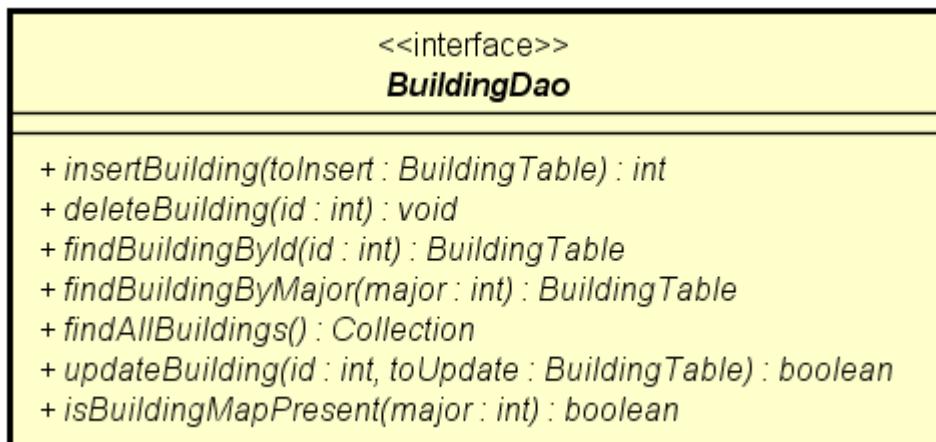
**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella Building del database locale;

**Descrizione:** Classe che contiene le informazioni corrette della tabella Building del database locale;

**Attributi:**

- + BUILDING\_TABLE\_CREATE : String {readOnly}  
Query per la creazione della tabella
- + COLUMN\_ADDRESS : String {readOnly}  
Valore della colonna address. Valore di default "address"
- + COLUMN\_DESCRIPTION : String {readOnly}  
Valore della colonna description. Valore di default "description"
- + COLUMN\_ID : String {readOnly}  
Valore della colonna id. Valore di default "id"
- + COLUMN\_MAPVERSION : String {readOnly}  
Valore della colonna mapVersion. Valore di default "mapVersion"
- + COLUMN\_NAME : String {readOnly}  
name
- + COLUMN\_OPENINGHOURS : String {readOnly}  
Valore della colonna openingHours. Valore di default "openingHours"
- + COLUMN\_UUID : String {readOnly}  
Valore della colonna uuid. Valore di default "uuid"
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "Building"

#### 4.4.19 model::dataaccess::dao::BuildingDao



**Figura 37:** Interfaccia BuildingDao

**Nome:** BuildingDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "Building" del database locale dalla loro implementazione;

**Descrizione:** Interfaccia che espone i metodi per un DAO per accedere alla tabella "Building" del database locale;

**Metodi:**

- + *deleteBuilding(id : int) : void*

Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "Building" del database locale

**Argomenti:**

- id : int

Identificativo dell'edificio di cui rimuovere le informazioni dal database locale

- + *findAllBuildings() : Collection<BuildingTable>*

Metodo che viene utilizzato per recuperare le informazioni di tutti gli edifici presenti nella tabella "Building" del database locale

- + *findBuildingById(id : int) : BuildingTable*

Metodo che viene utilizzato per recuperare le informazioni di tutti gli edifici presenti nella tabella "Building" del database locale

**Argomenti:**

- *id : int*

Identificativo dell'edificio di cui recuperare le informazioni

- + *findBuildingByMajor(major : int) : BuildingTable*

Metodo per recuperare le informazioni di un edificio dal database locale tramite il suo major, sotto forma di oggetto BuildingTable

**Argomenti:**

- *major : int*

Major identificante l'edificio che deve essere recuperato dal database

- + *insertBuilding(toInsert : BuildingTable) : int*

Metodo che permette l'inserimento delle informazioni di un edificio in una entry della tabella "Building" del database locale

**Argomenti:**

- *toInsert : BuildingTable*

Oggetto di tipo BuildingTable che contiene le informazioni dell'edificio

- + *isBuildingMapPresent(major : int) : boolean*

Metodo per verificare la presenza nel database locale delle informazioni di un edificio

**Argomenti:**

- *major : int*

major dell'edificio

- + *updateBuilding(id : int, toUpdate : BuildingTable) : boolean*

Metodo per aggiornare le informazioni di un edificio nella tabella "Building" del database locale

**Argomenti:**

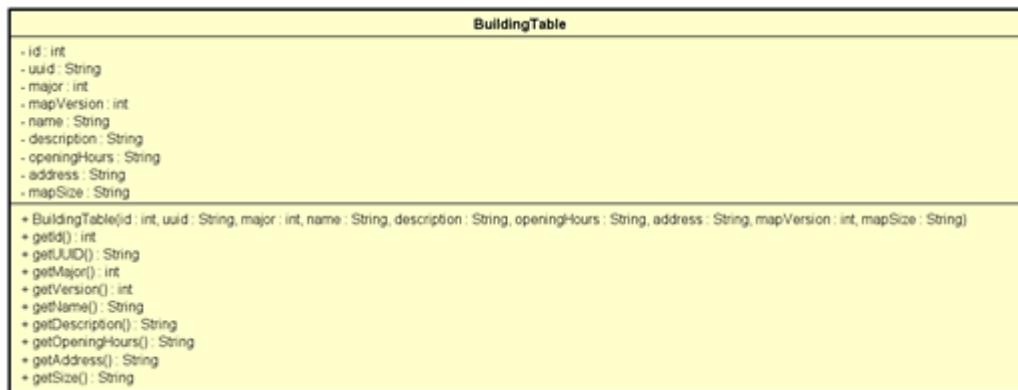
- *id : int*

Identificativo dell'edificio di cui aggiornare le informazioni

- *toUpdate : BuildingTable*

Oggetto che contiene le informazioni aggiornate dell'edificio

#### 4.4.20 model::dataaccess::dao::BuildingTable



**Figura 38:** Classe BuildingTable

**Nome:** BuildingTable;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella Building del database locale ;

**Descrizione:** Classe che rappresenta una ennupla della tabella Building del database locale;

**Attributi:**

- - address : String  
Indirizzo dell'edificio
- - description : String  
Descrizione dell'edificio
- - id : int  
Identificativo dell'edificio
- - major : int  
Major dell'edificio
- - mapSize : String  
Dimensione della mappa (in MB)
- - mapVersion : int  
Versione corrente della mappa

- - `name : String`  
Nome dell'edificio
- - `openingHours : String`  
Orario dell'apertura dell'edificio
- - `uuid : String`  
Identificativo dell'applicazione

**Metodi:**

- + `BuildingTable(id : int, uuid : String, major : int, name : String, description : String, openingHours : String, address : String, mapVersion : int, mapSize : String)`  
Costruttore della classe BuildingTable

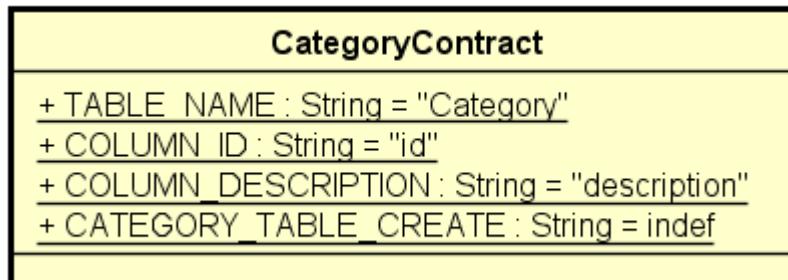
**Argomenti:**

- `id : int`  
Identificativo numerico dell'oggetto BuildingTable
- `uuid : String`  
Identificativo univoco
- `major : int`  
Major dell'edificio
- `name : String`  
Nome dell'edificio mappato
- `description : String`  
Descrizione dell'edificio mappato
- `openingHours : String`  
Orari di apertura dell'edificio mappato
- `address : String`  
Indirizzo dell'edificio mappato
- `mapVersion : int`  
Versione della mappa
- `mapSize : String`  
Dimensione della mappa (espressa in MB)

- + `getAddress() : String`  
Metodo che ritorna l'indirizzo dell'edificio
- + `getDescription() : String`  
Metodo che ritorna la descrizione dell'edificio
- + `getId() : int`  
Metodo che ritorna l'identificativo dell'edificio

- + `getMajor() : int`  
Metodo che ritorna il major dell'edificio
- + `getName() : String`  
Metodo che ritorna il nome dell'edificio
- + `getOpeningHours() : String`  
Metodo che ritorna l'orario di apertura dell'edificio
- + `getSize() : String`  
Metodo che ritorna la dimensione della mappa (in MB)
- + `getUUID() : String`  
Metodo che ritorna l'identificativo dell'applicazione
- + `getVersion() : int`  
Metodo che ritorna la versione della mappa dell'edificio

#### 4.4.21 model::dataaccess::dao::CategoryContract



**Figura 39:** Classe CategoryContract

**Nome:** CategoryContract;

**Tipo:** Classe;

**Visibilità:** public;

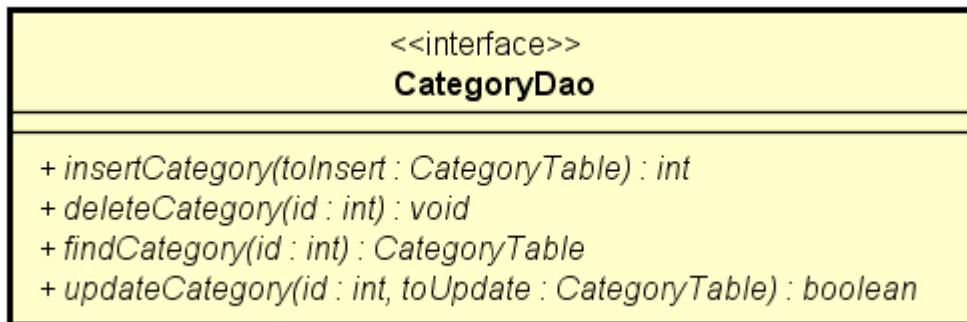
**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella Category del database locale;

**Descrizione:** Classe che contiene le informazioni corrette della tabella Category del database locale;

**Attributi:**

- + CATEGORY\_TABLE\_CREATE : String {readOnly}  
Query per la creazione della tabella
- + COLUMN\_DESCRIPTION : String {readOnly}  
Valore della colonna description. Valore di default "description"
- + COLUMN\_ID : String {readOnly}  
Valore della colonna id. Valore di default "id"
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "Category"

#### 4.4.22 model::dataaccess::dao::CategoryDao



**Figura 40:** Interfaccia CategoryDao

**Nome:** CategoryDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "Category" del database locale dalla loro implementazione;

**Descrizione:** .Interfaccia che espone i metodi per un DAO per accedere alla tabella "Category" del database locale;

**Metodi:**

- + *deleteCategory(id : int) : void*  
Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "Category" del database locale

**Argomenti:**

– `id : int`

Identificativo della categoria da rimuovere dal database locale

- + `findCategory(id : int) : CategoryTable`

Metodo per recuperare le informazioni di una categoria dal database locale tramite il suo identificativo, sotto forma di oggetto CategoryTable

**Argomenti:**

– `id : int`

Identificativo della categoria di cui recuperare le informazioni

- + `insertCategory(toInsert : CategoryTable) : int`

Metodo che permette l'inserimento di una categoria nella tabella "Category" del database locale

**Argomenti:**

– `toInsert : CategoryTable`

Oggetto di tipo CategoryTable che contiene le informazioni della categoria

- + `updateCategory(id : int, toUpdate : CategoryTable) : boolean`

Metodo per aggiornare le informazioni di una categoria nella tabella "Category" del database locale

**Argomenti:**

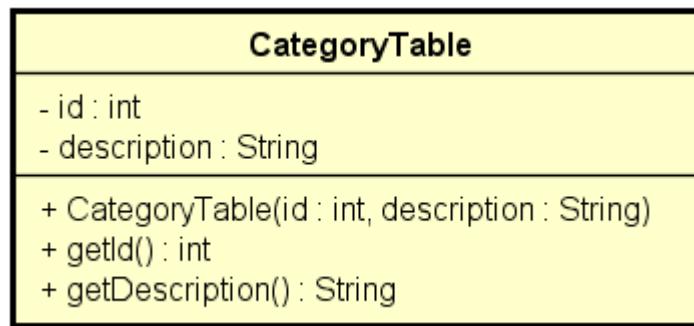
– `id : int`

Identificativo della categoria di cui aggiornare le informazioni

– `toUpdate : CategoryTable`

Oggetto che contiene le informazioni aggiornate della categoria

#### 4.4.23 model::dataaccess::dao::CategoryTable



**Figura 41:** Classe CategoryTable

**Nome:** CategoryTable;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella Category del database locale ;

**Descrizione:** Classe che rappresenta una ennupla della tabella Category del database locale;

**Attributi:**

- - description : String  
Nome della categoria
- - id : int  
Identificativo numerico dell'oggetto CategoryTable

**Metodi:**

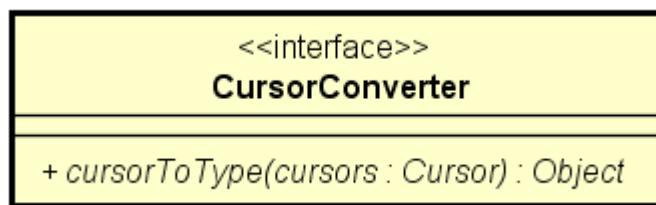
- + CategoryTable(id : int, description : String)  
Costruttore della classe CategoryTable

**Argomenti:**

- id : int  
Identificativo numerico dell'oggetto CategoryTable
- description : String  
Nome della categoria

- + `getDescription() : String`  
Metodo che restituisce il nome della categoria
- + `getId() : int`  
Metodo che restituisce l'identificativo numerico dell'oggetto CategoryTable

#### 4.4.24 model::dataaccess::dao::CursorConverter



**Figura 42:** Interfaccia CursorConverter

**Nome:** CursorConverter;

**Tipo:** Interfaccia;

**Componenti delle librerie utilizzate:**

- `android.database.Cursor` (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere definire una unica firma per la conversione delle informazioni prese dal database in oggetti;

**Descrizione:** Interfaccia base per la conversione di un Cursor in un oggetto;

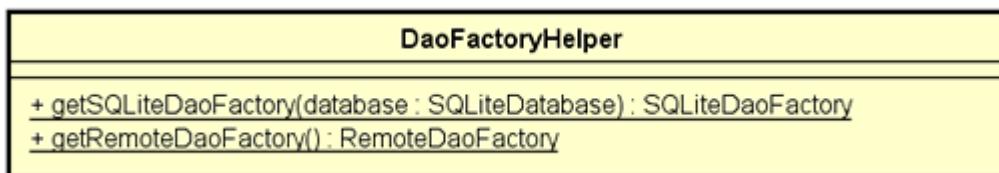
**Metodi:**

- + `cursorToType(cursor : Cursor) : Object`  
Metodo che viene utilizzato per convertire il risultato di una query sul database locale in un oggetto

**Argomenti:**

- `cursor : Cursor`  
Risultato della query sul database locale

#### 4.4.25 model::dataaccess::dao::DaoFactoryHelper



**Figura 43:** Classe DaoFactoryHelper

**Nome:** DaoFactoryHelper;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per ottenere un'istanza di SQLiteDaoFactory o di RemoteDaoFactory;

**Descrizione:** Classe che rappresenta un aiutante per ottenere un'istanza di una delle due Factory di DAO (locali o remoti);

**Metodi:**

- + getRemoteDaoFactory() : RemoteDaoFactory  
Metodo che viene utilizzato per ottenere un'istanza di RemoteDaoFactory
- + getSQLiteDaoFactory(database : SQLiteDatabase) : SQLiteDaoFactory  
Metodo che viene utilizzato per ottenere un'istanza di SQLiteDaoFactory

**Argomenti:**

- database : SQLiteDatabase  
Il database locale

#### 4.4.26 model::dataaccess::dao::EdgeContract

EdgeContract
<pre>+ TABLE NAME : String = "Edge" + COLUMN ID : String = "id" + COLUMN STARTROI : String = "startROI" + COLUMN ENDROI : String = "endROI" + COLUMN DISTANCE : String = "distance" + COLUMN COORDINATE : String = "coordinate" + COLUMN TYPEID : String = "typeId" + COLUMN ACTION : String = "action" + COLUMN LONGDESCRIPTION : String = "longDescription" + EDGE TABLE CREATE : String = indef</pre>

**Figura 44:** Classe EdgeContract

**Nome:** EdgeContract;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella Edge del database locale;

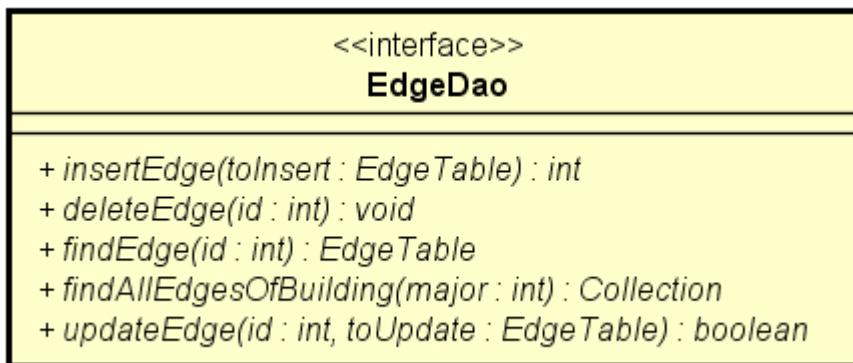
**Descrizione:** Classe che contiene le informazioni corrette della tabella Edge del database locale;

**Attributi:**

- + COLUMN\_ACTION : String {readOnly}  
Valore della colonna action. Valore di default "action"
- + COLUMN\_COORDINATE : String {readOnly}  
Valore della colonna coordinate. Valore di default "coordinate"
- + COLUMN\_DISTANCE : String {readOnly}  
Valore della colonna distance. Valore di default "distance"
- + COLUMN\_ENDROI : String {readOnly}  
Valore della colonna endROI. Valore di default "endROI"

- + COLUMN\_ID : String {readOnly}  
Valore della colonna id. Valore di default "id"
- + COLUMN\_LONGDESCRIPTION : String {readOnly}  
Valore della colonna longDescription. Valore di default "longDescription"
- + COLUMN\_STARTROI : String {readOnly}  
Valore della colonna startROI. Valore di default "startROI"
- + COLUMN\_TYPEID : String {readOnly}  
Valore della colonna typeId. Valore di default "typeId"
- + EDGE\_TABLE\_CREATE : String {readOnly}  
Query per la creazione della tabella
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "Edge"

#### 4.4.27 model::dataaccess::dao::EdgeDao



**Figura 45:** Interfaccia EdgeDao

**Nome:** EdgeDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "Edge" del database locale dalla loro implementazione;

**Descrizione:** Interfaccia che espone i metodi per un DAO per accedere alla tabella "Edge" del database locale;

**Metodi:**

- + *deleteEdge(id : int) : void*

Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "Edge" del database locale

**Argomenti:**

- *id : int*

Identificativo dell'arco di cui rimuovere le informazioni dal database locale

- + *findAllEdgesOfBuilding(major : int) : Collection<EdgeTable>*

Metodo che viene utilizzato per recuperare le informazioni di tutti gli archi presenti nella tabella "Edge" del database locale

**Argomenti:**

- *major : int*

Identificativo major dell'edificio di cui si vogliono recuperare tutti gli archi

- + *findEdge(id : int) : EdgeTable*

Metodo per recuperare le informazioni di un arco dal database locale tramite il suo identificativo, sotto forma di oggetto EdgeTable

**Argomenti:**

- *id : int*

Identificativo dell'arco di cui recuperare le informazioni

- + *insertEdge(toInsert : EdgeTable) : int*

Metodo che permette l'inserimento delle informazioni di un edificio in una entry della tabella "Edge" del database locale

**Argomenti:**

- *toInsert : EdgeTable*

Oggetto di tipo EdgeTable che contiene le informazioni dell'arco

- + *updateEdge(id : int, toUpdate : EdgeTable) : boolean*

Metodo per aggiornare le informazioni di un edificio nella tabella "Edge" del database locale

**Argomenti:**

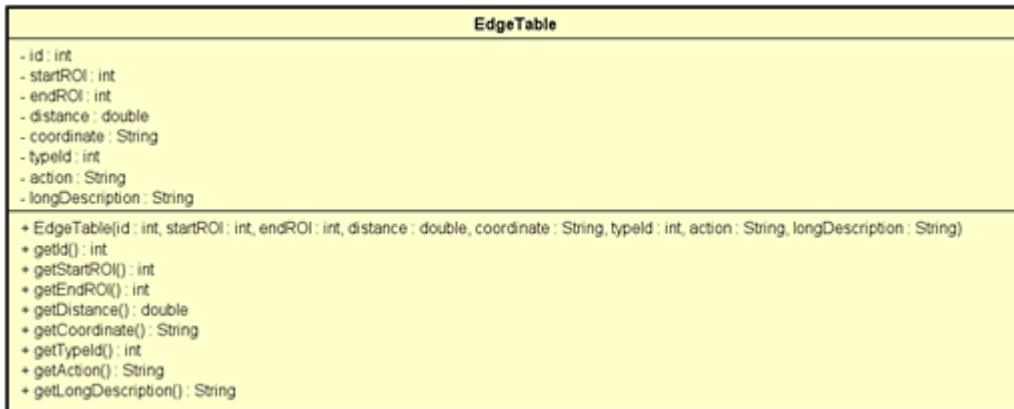
- *id : int*

Identificativo dell'arco di cui aggiornare le informazioni

– **toUpdate** : EdgeTable

Oggetto che contiene le informazioni aggiornate dell'arco

#### 4.4.28 model::dataaccess::dao::EdgeTable



**Figura 46:** Classe EdgeTable

**Nome:** EdgeTable;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella Edge del database locale ;

**Descrizione:** Classe che rappresenta una ennupla della tabella Edge del database locale;

**Attributi:**

- **- action : String**  
Descrizione dell'azione da compiere per arrivare dallo startROI all'endROI
- **- coordinate : String**  
Angolo rispetto al Nord polare presente lo startROI e l'endROI
- **- distance : double**  
Distanza tra lo startROI e l'endROI

- - `endROI : int`  
Nodo d'arrivo dell'arco
- - `id : int`  
Identificativo dell'Edge
- - `longDescription : String`  
Descrizione testuale estesa per raggiungere l'endROI a partire dallo startROI
- - `startROI : int`  
Nodo di partenza dell'arco
- - `typeId : int`  
Identificativo del tipo di Edge

**Metodi:**

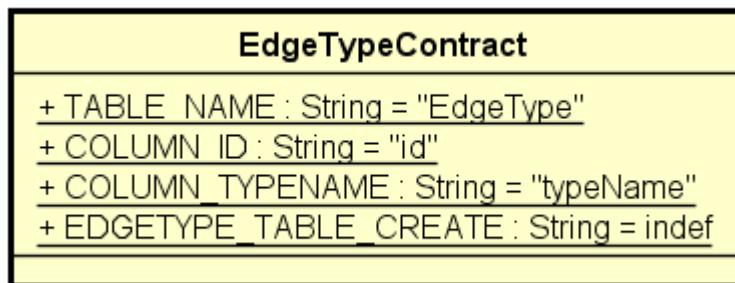
- + `EdgeTable(id : int, startROI : int, endROI : int, distance : double, coordinate : String, typeId : int, action : String, longDescription : String)`  
Costruttore della classe EdgeTable

**Argomenti:**

- `id : int`  
Identificativo dell'Edge
- `startROI : int`  
Nodo di partenza dell'arco
- `endROI : int`  
Nodo d'arrivo dell'arco
- `distance : double`  
Distanza tra lo startROI e l'endROI
- `coordinate : String`  
Angolo rispetto al Nord polare presente lo startROI e l'endROI
- `typeId : int`  
Identificativo del tipo di Edge
- `action : String`  
Descrizione dell'azione da compiere per arrivare dallo startROI all'endROI
- `longDescription : String`  
Descrizione testuale estesa per raggiungere l'endROI a partire dallo startROI

- + `getAction() : String`  
Metodo che ritorna la descrizione dell'azione da compiere per arrivare dallo startROI all'endROI
- + `getCoordinate() : String`  
Metodo che ritorna l'angolo rispetto al Nord polare presente lo startROI e l'endROI
- + `getDistance() : double`  
Metodo che ritorna la distanza tra lo startROI e l'endROI
- + `getEndROI() : int`  
Metodo che ritorna il nodo d'arrivo dell'arco
- + `getId() : int`  
Metodo che ritorna l'identificativo dell'Edge
- + `getLongDescription() : String`  
Metodo che ritorna la descrizione testuale estesa per raggiungere l'endROI a partire dallo startROI
- + `getStartROI() : int`  
Metodo che ritorna il nodo di partenza dell'arco
- + `getTypeId() : int`  
Metodo che ritorna l'identificativo del tipo di Edge

#### 4.4.29 model::dataaccess::dao::EdgeTypeContract



**Figura 47:** Classe EdgeTypeContract

**Nome:** EdgeTypeContract;

**Tipo:** Classe;

**Visibilità:** public;

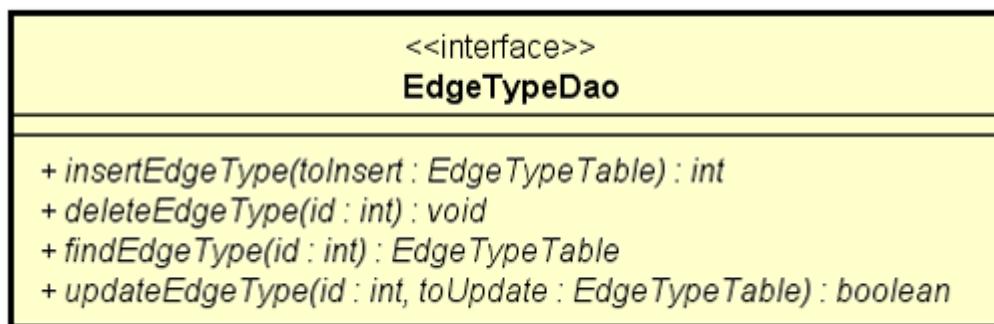
**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella EdgeType del database locale;

**Descrizione:** Classe che contiene le informazioni corrette della tabella EdgeType del database locale;

**Attributi:**

- + COLUMN\_ID : String {readOnly}  
Valore della colonna id. Valore di default "id"
- + COLUMN\_TYPENAME : String {readOnly}  
Valore della colonna typeName. Valore di default "typeName"
- + EDGETYPE\_TABLE\_CREATE : String {readOnly}  
Query per la creazione della tabella
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "EdgeType"

#### 4.4.30 model::dataaccess::dao::EdgeTypeDao



**Figura 48:** Interfaccia EdgeTypeDao

**Nome:** EdgeTypeDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "EdgeType" del database locale dalla loro implementazione;

**Descrizione:** Interfaccia che espone i metodi per un DAO per accedere alla tabella "EdgeType" del database locale;

**Metodi:**

- + *deleteEdgeType(id : int) : void*

Metodo che permette la rimozione delle informazioni di un tipo di Edge dalla tabella "EdgeType" del database locale

**Argomenti:**

- *id : int*

Identificativo del tipo di Edge di cui rimuovere le informazioni dal database locale

- + *findEdgeType(id : int) : EdgeTypeTable*

Metodo per recuperare le informazioni di un tipo di Edge dal database locale tramite il suo identificativo, sotto forma di oggetto EdgeTypeTable

**Argomenti:**

- *id : int*

Identificativo del tipo di Edge di cui recuperare le informazioni

- + *insertEdgeType(toInsert : EdgeTypeTable) : int*

Metodo che permette l'inserimento delle informazioni del tipo di Edge in una entry della tabella "EdgeType" del database locale

**Argomenti:**

- *toInsert : EdgeTypeTable*

Oggetto di tipo EdgeTypeTable che contiene le informazioni di un tipo di Edge

- + *updateEdgeType(id : int, toUpdate : EdgeTypeTable) : boolean*

Metodo per aggiornare le informazioni di un tipo di Edge nella tabella "EdgeType" del database locale

**Argomenti:**

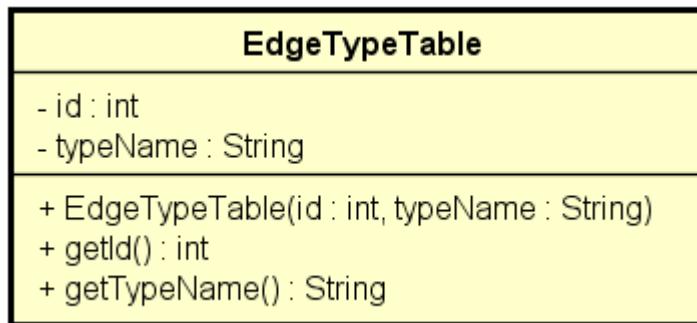
- *id : int*

Identificativo del tipo di Edge di cui aggiornare le informazioni

- *toUpdate : EdgeTypeTable*

Oggetto che contiene le informazioni aggiornate del tipo di Edge

#### 4.4.31 model::dataaccess::dao::EdgeTypeTable



**Figura 49:** Classe EdgeTypeTable

**Nome:** EdgeTypeTable;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella EdgeType del database locale ;

**Descrizione:** Classe che rappresenta una ennupla della tabella EdgeType del database locale;

**Attributi:**

- - id : int  
Identificativo numerico dell'oggetto EdgeTypeTable
- - typeName : String  
Identificativo numerico che permette di identificare il tipo di Edge

**Metodi:**

- + EdgeTypeTable(id : int, typeName : String)  
Costruttore della classe EdgeTypeTable

**Argomenti:**

- id : int  
Identificativo numerico dell'oggetto EdgeTypeTable

- `typeName : String`  
Identificativo numerico che permette di identificare il tipo di Edge
- + `getId() : int`  
Metodo che restituisce l'identificativo numerico dell'oggetto EdgeTypeTable
- + `get TypeName() : String`  
Metodo che restituisce l'identificativo numerico che permette di identificare il tipo di Edge

#### 4.4.32 model::dataaccess::dao::MapsDbContract



**Figura 50:** Classe MapsDbContract

**Nome:** MapsDbContract;

**Tipo:** Classe;

**Visibilità:** public;

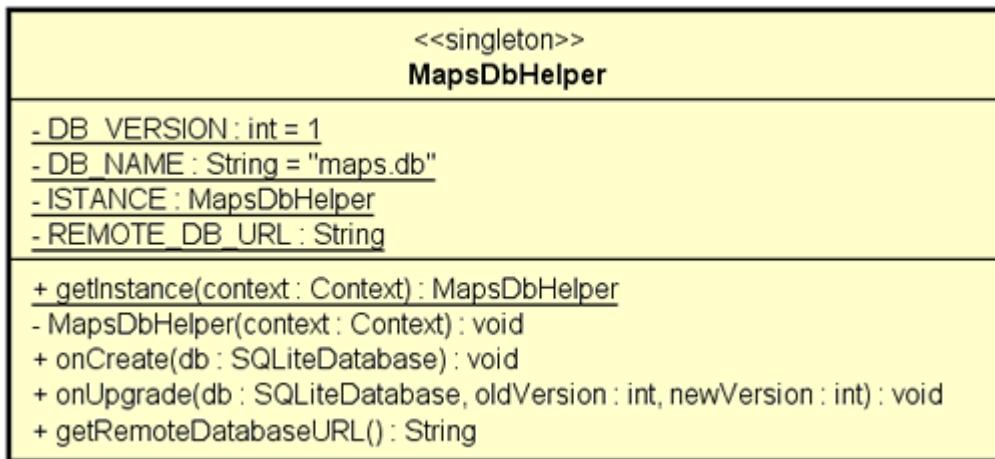
**Utilizzo:** Viene utilizzata per reperire la query corretta per creare tutte le tabelle del database locale;

**Descrizione:** Classe che contiene la query corretta per creare tutte le tabelle del database locale;

**Attributi:**

- `ALL_TABLES_CREATE : String {readOnly}`  
Query per la creazione di tutte le tabelle del database locale

#### 4.4.33 model::dataaccess::dao::MapsDbHelper



**Figura 51:** Classe MapsDbHelper

**Nome:** MapsDbHelper;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.database.sqlite.SQLiteOpenHelper` (Android).

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per ottenere un'istanza di SQLiteDatabase o l'URL del database remoto;

**Descrizione:** Classe che rappresenta un aiutante per ottenere informazioni su come accedere al database locale e remoto;

**Attributi:**

- - DB\_NAME : String {readOnly}  
Nome del database locale. Valore di default: "maps.db"
- - DB\_VERSION : int {readOnly}  
Numero di versione del database locale. Valore di default: "1"
- - INSTANCE : MapsDbHelper {readOnly}  
Istanza di MapsDbHelper salvata per poter essere condivisa

- - REMOTE\_DB\_URL : String {readOnly}

URL del database remoto

#### Metodi:

- + getInstance() : MapsDbHelper

Metodo che ritorna una istanza di MapsDbHelper

- + getRemoteDatabaseURL() : String

Metodo che ritorna l'URL del database remoto

- - MapsDbHelper()

Costruttore della classe MapsDbHelper

- + onCreate(db : SQLiteDatabase) : void

**Override** Metodo che viene chiamato la prima volta che viene creato il database

#### Argomenti:

- db : SQLiteDatabase

Riferimento al database

- + onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void

**Override** Metodo che viene chiamato per effettuare l'upgrade del database

#### Argomenti:

- db : SQLiteDatabase

Riferimento al database

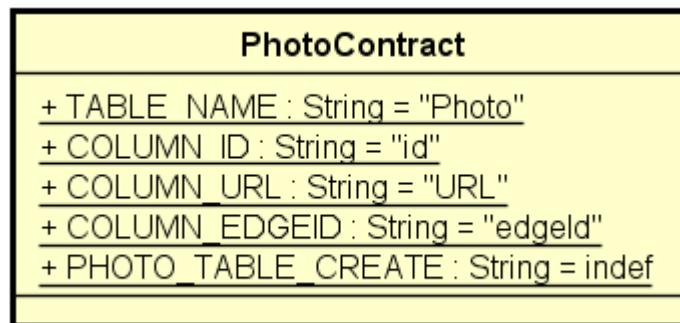
- oldVersion : int

Numero di versione del vecchio database

- newVersion : int

Numero di versione del nuovo database

#### 4.4.34 model::dataaccess::dao::PhotoContract



**Figura 52:** Classe PhotoContract

**Nome:** PhotoContract;

**Tipo:** Classe;

**Visibilità:** public;

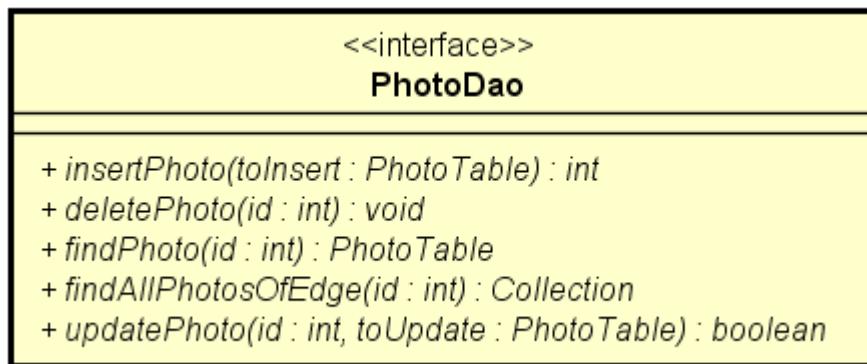
**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella Photo del database locale;

**Descrizione:** Classe che contiene le informazioni corrette della tabella Photo del database locale;

**Attributi:**

- + COLUMN\_EDGEID : String {readOnly}  
Valore della colonna edgeId. Valore di default "edgeId"
- + COLUMN\_ID : String {readOnly}  
Valore della colonna id. Valore di default "id"
- + COLUMN\_URL : String {readOnly}  
Valore della colonna url. Valore di default "url"
- + PHOTO\_TABLE\_CREATE : String {readOnly}  
Query per la creazione della tabella
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "Photo"

#### 4.4.35 model::dataaccess::dao::PhotoDao



**Figura 53:** Interfaccia PhotoDao

**Nome:** PhotoDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "Photo" del database locale dalla loro implementazione;

**Descrizione:** Interfaccia che espone i metodi per un DAO per accedere alla tabella "Photo" del database locale;

**Metodi:**

- `+ deletePhoto(id : int) : void`  
Metodo che permette la rimozione delle informazioni di una foto dalla tabella "Photo" del database locale

**Argomenti:**

– `id : int`

Identificativo della foto di cui rimuovere le informazioni dal database locale

- `+ findAllPhotosOfEdge(id : int) : Collection<PhotoTable>`  
Metodo che viene utilizzato per recuperare le informazioni di tutte foto associate ad un Edge presenti nella tabella "Photo" del database locale

**Argomenti:**

– `id : int`

Identificativo dell'Edge

- `+ findPhoto(id : int) : PhotoTable`

Metodo per recuperare le informazioni di una foto dal database locale tramite il suo identificativo, sotto forma di oggetto PhotoTable

**Argomenti:**

– `id : int`

Identificativo della foto

- `+ insertPhoto(toInser : PhotoTable) : int`

Metodo che permette l'inserimento delle informazioni di una foto in una entry della tabella "Photo" del database locale

**Argomenti:**

– `toInser : PhotoTable`

Oggetto di tipo Photo che contiene le informazioni della foto

- `+ updatePhoto(id : int, toUpdate : PhotoTable) : boolean`

Metodo per aggiornare le informazioni di una foto nella tabella "Photo" del database locale

**Argomenti:**

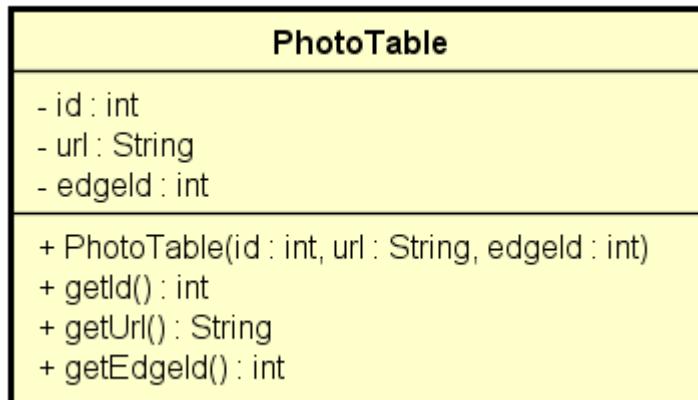
– `id : int`

Identificativo della foto di cui aggiornare le informazioni

– `toUpdate : PhotoTable`

Oggetto che contiene le informazioni aggiornate della foto

#### 4.4.36 model::dataaccess::dao::PhotoTable



**Figura 54:** Classe PhotoTable

**Nome:** PhotoTable;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella Photo del database locale ;

**Descrizione:** Classe che rappresenta una ennupla della tabella Photo del database locale;

**Attributi:**

- - edgeId : int  
Identificativo numerico dell'Edge a cui fa riferimento la foto
- - id : int  
Identificativo numerico dell'oggetto PhotoTable
- - url : String  
Stringa che rappresenta l'URL dove si può reperire la foto

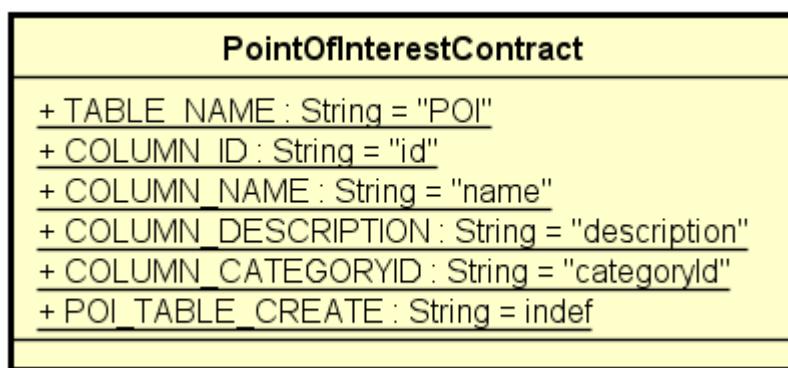
**Metodi:**

- + getEdgeId() : int  
Metodo che viene utilizzato per recuperare l'identificativo numerico dell'Edge a cui fa riferimento la foto

- + getId() : int  
Metodo che viene utilizzato per recuperare l'identificativo numerico della foto nel database
- + getUrl() : String  
Metodo per recuperare la stringa che rappresenta l'URL dove è possibile reperire la foto
- + PhotoTable(id : int, url : String, edgeId : int)  
Costruttore della classe PhotoTable

**Argomenti:**

- id : int  
Identificativo numerico della foto nel database locale
- url : String  
Stringa che rappresenta l'URL dove si può reperire la foto
- edgeId : int  
Identificativo numerico dell'Edge a cui fa riferimento la foto

**4.4.37 model::dataaccess::dao::PointOfInterestContract**

**Figura 55:** Classe PointOfInterestContract

**Nome:** PointOfInterestContract;

**Tipo:** Classe;

**Visibilità:** public;

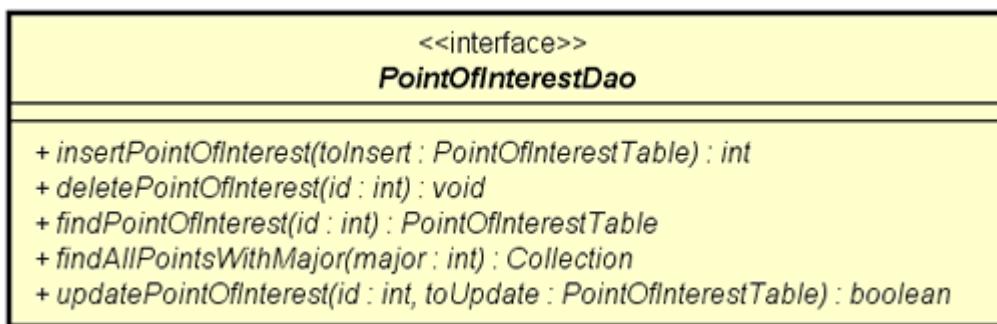
**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella POI del database locale;

**Descrizione:** Classe che contiene le informazioni corrette della tabella POI del database locale;

**Attributi:**

- + COLUMN\_CATEGORYID : String {readOnly}  
Valore della colonna categoryId. Valore di default "categoryId"
- + COLUMN\_DESCRIPTION : String {readOnly}  
Valore della colonna description. Valore di default "description"
- + COLUMN\_ID : String {readOnly}  
Valore della colonna id. Valore di default "id"
- + COLUMN\_NAME : String {readOnly}  
Valore della colonna name. Valore di default "name"
- + POI\_TABLE\_CREATE : String {readOnly}  
Contiene la query di creazione della tabella
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "POI"

#### 4.4.38 model::dataaccess::dao::PointOfInterestDao



**Figura 56:** Interfaccia PointOfInterestDao

**Nome:** PointOfInterestDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "POI" del database locale dalla loro implementazione;

**Descrizione:** Interfaccia che espone i metodi per un DAO per accedere alla tabella "POI" del database locale;

**Metodi:**

- + *deletePointOfInterest(id : int) : void*

Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "POI" del database locale

**Argomenti:**

- *id : int*

Identificativo del POI di cui rimuovere le informazioni dal database locale

- + *findAllPointsWithMajor(major : int) : Collection<PointOfInterestTable>*

Metodo che viene utilizzato per recuperare le informazioni di tutti gli edifici presenti nella tabella "POI" del database locale

**Argomenti:**

- *major : int*

Identificativo Major associato a tutti i beacon presenti in uno stesso edificio

- + *findPointOfInterest(id : int) : PointOfInterestTable*

Metodo per recuperare le informazioni di un edificio dal database locale tramite il suo identificativo, sotto forma di oggetto PointOfInterestTable

**Argomenti:**

- *id : int*

Identificativo del POI di cui recuperare le informazioni

- + *insertPointOfInterest(toInsert : PointOfInterestTable) : int*

Metodo che permette l'inserimento delle informazioni di un edificio in una entry della tabella "POI" del database locale

**Argomenti:**

- *toInsert : PointOfInterestTable*

Oggetto di tipo PointOfInterestTable che contiene le informazioni dell'edificio

- + *updatePointOfInterest(id : int, toUpdate : PointOfInterestTable) : boolean*

Metodo per aggiornare le informazioni di un edificio nella tabella "POI" del database locale

**Argomenti:**

- **id** : int  
Identificativo del POI di cui aggiornare le informazioni
- **toUpdate** : PointOfInterestTable  
Oggetto che contiene le informazioni aggiornate del POI

**4.4.39 model::dataaccess::dao::PointOfInterestTable**

PointOfInterestTable	
- id : int	
- name : String	
- description : String	
- categoryId : int	
+ PointOfInterestTable(id : int, name : String, description : String, categoryId : int)	
+ getId() : int	
+ getName() : String	
+ getDescription() : String	
+ getCategoryID() : int	

**Figura 57:** Classe PointOfInterestTable**Nome:** PointOfInterestTable;**Tipo:** Classe;**Visibilità:** public;**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella PointOfInterest del database locale ;**Descrizione:** Classe che rappresenta una ennupla della tabella PointOfInterest del database locale;**Attributi:**

- - **categoryId** : int  
Identificativo della categoria a cui appartiene il POI
- - **description** : String  
Descrizione del POI
- - **id** : int  
Identificativo del POI

- - `name` : `String`

Nome del POI

#### Metodi:

- + `getCategoryId()` : `int`

Metodo che ritorna l'identificativo del POI

- + `getDescription()` : `String`

Metodo che ritorna la descrizione del POI

- + `getId()` : `int`

Metodo che ritorna l'identificativo del POI

- + `getName()` : `String`

Metodo che ritorna il nome dell'edificio

- + `PointOfInterestTable(description : String, id : int, name : String, category : int)`

Costruttore della classe PointOfInterestTable

#### Argomenti:

- `description` : `String`

Descrizione del POI

- `id` : `int`

Identificativo del POI

- `name` : `String`

Nome del POI

- `category` : `int`

Identificativo della categoria a cui appartiene il POI

#### 4.4.40 model::dataaccess::dao::RegionOfInterestContract



**Figura 58:** Classe RegionOfInterestContract

**Nome:** RegionOfInterestContract;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella ROI del database locale;

**Descrizione:** Classe che contiene le informazioni corrette della tabella ROI del database locale;

**Attributi:**

- + COLUMN\_ID : String {readOnly}  
Valore della colonna id. Valore di default "id"
- + COLUMN\_MAJOR : String {readOnly}  
Valore della colonna major. Valore di default "major"
- + COLUMN\_MINOR : String {readOnly}  
Valore della colonna minor. Valore di default "minor"
- + COLUMN\_UUID : String {readOnly}  
Valore della colonna uuid. Valore di default "uuid"
- + ROI\_TABLE\_CREATE : String {readOnly}  
Contiene la query di creazione della tabella
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "ROI"

#### 4.4.41 model::dataaccess::dao::RegionOfInterestDao



**Figura 59:** Interfaccia RegionOfInterestDao

**Nome:** RegionOfInterestDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "ROI" del database locale dalla loro implementazione;

**Descrizione:** Interfaccia che espone i metodi per un DAO per accedere alla tabella "ROI" del database locale;

**Metodi:**

- + `deleteRegionOfInterest(id : int) : void`

Metodo che permette la rimozione delle informazioni di una RegionOfInterest dalla tabella "ROI" del database locale

**Argomenti:**

- `id : int`

Identificativo della RegionOfInterest di cui rimuovere le informazioni dal database locale

- + `findAllRegionsWithMajor(major : int) : Collection<RegionOfInterestTable>`

Metodo che viene utilizzato per recuperare le informazioni di tutte le RegionOfInterest associato ad certo edificio, dato il major dell'edificio

**Argomenti:**

- `major : int`  
Major dell'edificio

- + `findRegionOfInterest(id : int) : RegionOfInterestTable`  
Metodo per recuperare le informazioni di una RegionOfInterest dal database locale tramite il suo identificativo, sotto forma di oggetto RegionOfInterestTable

**Argomenti:**

- `id : int`  
Identificativo della RegionOfInterest di cui recuperare le informazioni

- + `insertRegionOfInterest(toInsert : RegionOfInterestTable) : int`

Metodo che permette l'inserimento delle informazioni di una RegionOfInterest in una entry della tabella "ROI" del database locale

**Argomenti:**

- `toInsert : RegionOfInterestTable`  
Oggetto di tipo RegionOfInterestTable che contiene le informazioni della RegionOfInterest

- + `updateRegionOfInterest(id : int, toUpdate : RegionOfInterestTable) : boolean`

Metodo per aggiornare le informazioni di una RegionOfInterest nella tabella "ROI" del database locale

**Argomenti:**

- `id : int`  
Identificativo della RegionOfInterest di cui aggiornare le informazioni
- `toUpdate : RegionOfInterestTable`  
Oggetto che contiene le informazioni aggiornate della RegionOfInterest

#### 4.4.42 model::dataaccess::dao::RegionOfInterestTable



**Figura 60:** Classe RegionOfInterestTable

**Nome:** RegionOfInterestTable;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella RegionOfInterest del database locale ;

**Descrizione:** Classe che rappresenta una ennupla della tabella RegionOfInterest del database locale;

**Attributi:**

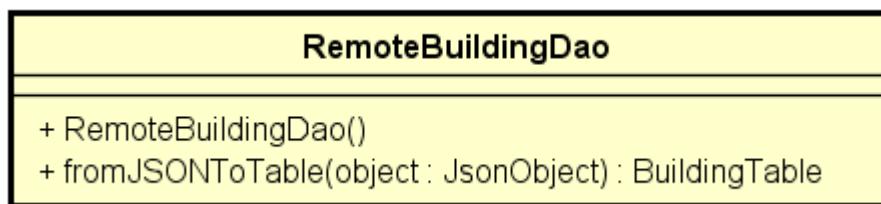
- - `id : int`  
Identificativo della RegionOfInterest
- - `major : int`  
Major dell'edificio
- - `minor : int`  
Identificativo del beacon associato alla ROI
- - `uuid : String`  
UUID dell'applicazione

**Metodi:**

- + getId() : int  
Metodo che ritorna l'identificativo della ROI
- + getMajor() : int  
Metodo che ritorna il major dell'edificio
- + getMinor() : int  
Metodo che ritorna l'identificativo del beacon associato alla ROI
- + getUUID() : String  
Metodo che ritorna l'UUID dell'applicazione
- + RegionOfInterestTable(id : int, uuid : String, major : int, minor : int)  
Costruttore della classe RegionOfInterestTable

**Argomenti:**

- id : int  
Identificativo della RegionOfInterest
- uuid : String  
Identificativo dell'applicazione
- major : int  
Major dell'edificio
- minor : int  
Identificativo del beacon associato alla ROI

**4.4.43 model::dataaccess::dao::RemoteBuildingDao****Figura 61:** Classe RemoteBuildingDao**Nome:** RemoteBuildingDao;**Tipo:** Classe;**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;

- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti BuildingTable che rappresentano la tabella "Building" del database locale;

**Descrizione:** Classe di utility per la conversione da JSON a BuildingTable;

**Metodi:**

- + fromJSONToTable(object : JsonObject) : BuildingTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto BuildingTable, che viene ritornato

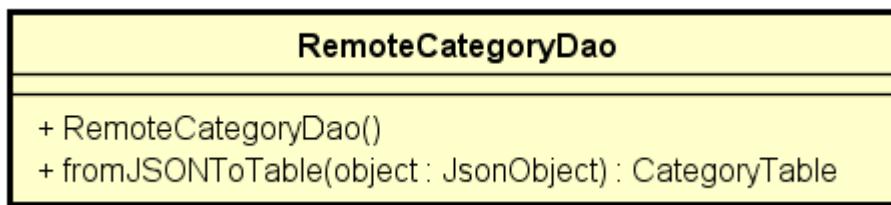
**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo BuildingTable

- + RemoteBuildingDao()

Costruttore di default per la classe RemoteBuildingDao

#### 4.4.44 model::dataaccess::dao::RemoteCategoryDao



**Figura 62:** Classe RemoteCategoryDao

**Nome:** RemoteCategoryDao;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti CategoryTable che rappresentano la tabella "Category" del database locale;

**Descrizione:** Classe di utility per la conversione da JSON a CategoryTable;

**Metodi:**

- + fromJSONToTable(object : JsonObject) : CategoryTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto CategoryTable, che viene ritornato

**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo CategoryTable

- + RemoteCategoryDao()  
Costruttore di default per la classe RemoteCategoryDao

#### 4.4.45 model::dataaccess::dao::RemoteDaoFactory

RemoteDaoFactory
+ RemoteDaoFactory() + getBuildingDao() : RemoteBuildingDao + getPointOfInterestDao() : RemotePointOfInterestDao + getRoiPoiDao() : RemoteRoiPoiDao + getEdgeDao() : RemoteEdgeDao + getCategoryDao() : RemoteCategoryDao + getEdgeTypeDao() : RemoteEdgeTypeDao + getPhotoDao() : RemotePhotoDao + getRegionOfInterestDao() : RemoteRegionOfInterestDao

**Figura 63:** Classe RemoteDaoFactory

**Nome:** RemoteDaoFactory;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per creare e ottenere oggetti DAO remoti;

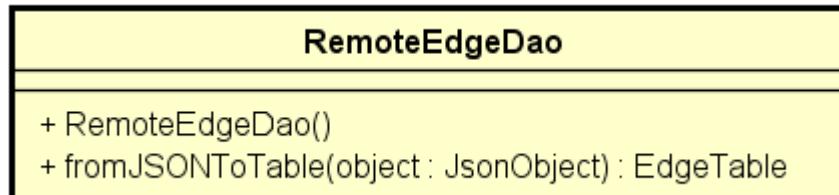
**Descrizione:** Classe che rappresenta la factory per creare tutti gli oggetti DAO remoti;

**Metodi:**

- + getBuildingDao() : RemoteBuildingDao  
Metodo che viene utilizzato per ottenere un'istanza di RemoteBuildingDao
- + getCategoryDao() : RemoteCategoryDao  
Metodo che viene utilizzato per ottenere un'istanza di RemoteCategoryDao
- + getEdgeDao() : RemoteEdgeDao  
Metodo che viene utilizzato per ottenere un'istanza di RemoteEdgeDao

- + `getEdgeTypeDao()` : `RemoteEdgeTypeDao`  
Metodo che viene utilizzato per ottenere un'istanza di `RemoteEdgeTypeDao`
- + `getPhotoDao()` : `RemotePhotoDao`  
Metodo che viene utilizzato per ottenere un'istanza di `RemotePhotoDao`
- + `getPointOfInterestDao()` : `RemotePointOfInterestDao`  
Metodo che viene utilizzato per ottenere un'istanza di `RemotePointOfInterestDao`
- + `getRegionOfInterestDao()` : `RemoteRegionOfInterestDao`  
Metodo che viene utilizzato per ottenere un'istanza di `RemoteRegionOfInterestDao`
- + `getRoiPoiDao()` : `RemoteRoiPoiDao`  
Metodo che viene utilizzato per ottenere un'istanza di `RemoteRoiPoiDao`
- + `RemoteDaoFactory()`  
Costruttore di default per la classe `RemoteDaoFactory`

#### 4.4.46 model::dataaccess::dao::`RemoteEdgeDao`



**Figura 64:** Classe `RemoteEdgeDao`

**Nome:** `RemoteEdgeDao`;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `com.google.api.client.json.JsonObjectParser` ;
- `com.google.gson.JsonArray`;
- `com.google.gson.JsonElement`;

- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti EdgeTable che rappresentano la tabella "Edge" del database locale;

**Descrizione:** Classe di utility per la conversione da JSON a EdgeTable;

**Metodi:**

- + fromJSONToTable(object : JsonObject) : EdgeTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto EdgeTable, che viene ritornato

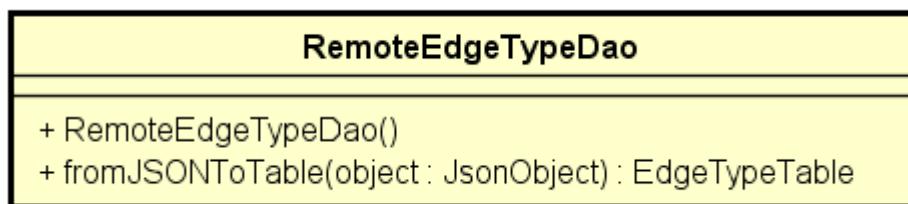
**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo EdgeTable

- + RemoteEdgeDao()

Costruttore di default per la classe RemoteEdgeDao

#### 4.4.47 model::dataaccess::dao::RemoteEdgeTypeDao



**Figura 65:** Classe RemoteEdgeTypeDao

**Nome:** RemoteEdgeTypeDao;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;

- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti EdgeTypeTable che rappresentano la tabella "EdgeType" del database locale;

**Descrizione:** Classe di utility per la conversione da JSON a EdgeTypeTable;

**Metodi:**

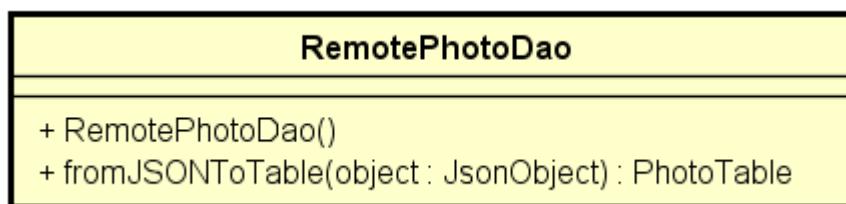
- + fromJSONToTable(object : JsonObject) : EdgeTypeTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto EdgeTypeTable, che viene ritornato

**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo EdgeTypeTable

- + RemoteEdgeTypeDao()  
Costruttore di default per la classe RemoteEdgeTypeDao

#### 4.4.48 model::dataaccess::dao::RemotePhotoDao



**Figura 66:** Classe RemotePhotoDao

**Nome:** RemotePhotoDao;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;

- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti PhotoTable che rappresentano la tabella "Photo" del database locale;

**Descrizione:** Classe di utility per la conversione da JSON a PhotoTable;

**Metodi:**

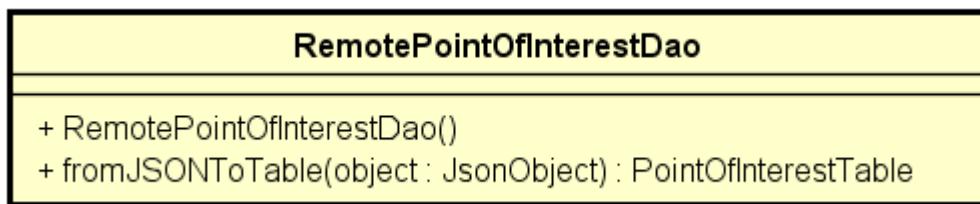
- + fromJSONToTable(object : JsonObject) : PhotoTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto PhotoTable, che viene ritornato

**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo PhotoTable

- + RemotePhotoDao()  
Costruttore di default per la classe RemotePhotoDao

#### 4.4.49 model::dataaccess::dao::RemotePointOfInterestDao



**Figura 67:** Classe RemotePointOfInterestDao

**Nome:** RemotePointOfInterestDao;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti PointOfInterestTable che rappresentano la tabella "POI" del database locale;

**Descrizione:** Classe di utility per la conversione da JSON a PointOfInterestTable;

**Metodi:**

- + fromJSONToTable(object : JsonObject) : PointOfInterestTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto PointOfInterestTable, che viene ritornato

**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo PointOfInterestTable

- + RemotePointOfInterestDao()  
Costruttore di default per la classe RemotePointOfInterestDao

#### 4.4.50 model::dataaccess::dao::RemoteRegionOfInterestDao



**Figura 68:** Classe RemoteRegionOfInterestDao

**Nome:** RemoteRegionOfInterestDao;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

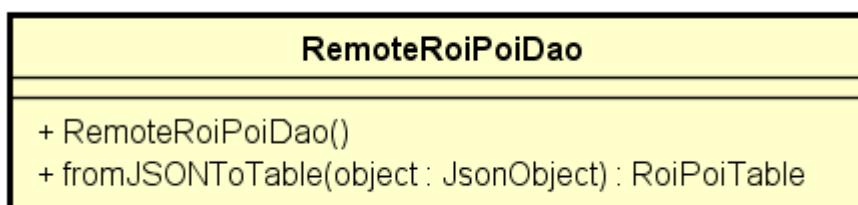
**Visibilità:** public;**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti RegionOfInterestTable che rappresentano la tabella "ROI" del database locale;**Descrizione:** Classe di utility per la conversione da JSON a RegionOfInterestTable;**Metodi:**

- + fromJSONToTable(object : JsonObject) : RegionOfInterestTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto RegionOfInterestTable, che viene ritornato

**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo RegionOfInterestTable

- + RemoteRegionOfInterestDao()  
Costruttore di default per la classe RemoteRegionOfInterestDao

**4.4.51 model::dataaccess::dao::RemoteRoiPoiDao****Figura 69:** Classe RemoteRoiPoiDao**Nome:** RemoteRoiPoiDao;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti RoiPoiTable che rappresentano la tabella "ROIPOI" del database locale;

**Descrizione:** Classe di utility per la conversione da JSON a RoiPoiTable;

**Metodi:**

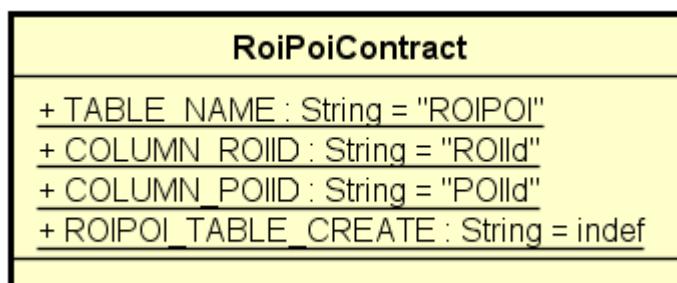
- + fromJSONToTable(object : JsonObject) : RoiPoiTable  
Metodo utilizzato per la conversione di un oggetto JsonObject in un oggetto RoiPoiTable, che viene ritornato

**Argomenti:**

- object : JsonObject  
Oggetto JSON che rappresenta un oggetto di tipo RoiPoiTable

- + RemoteRoiPoiDao()  
Costruttore di default per la classe RemoteRoiPoiDao

#### 4.4.52 model::dataaccess::dao::RoiPoiContract



**Figura 70:** Classe RoiPoiContract

**Nome:** RoiPoiContract;

**Tipo:** Classe;

**Visibilità:** public;

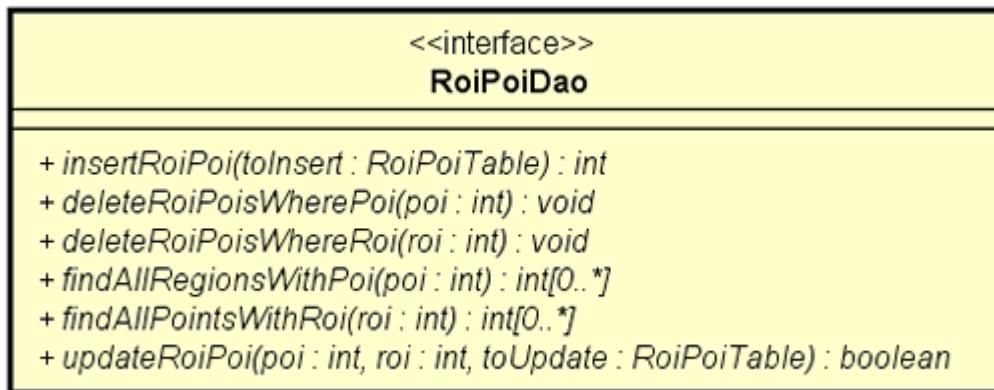
**Utilizzo:** Utilizzata per reperire le informazioni corrette della tabella ROI-POI del database locale;

**Descrizione:** Classe che contiene le informazioni corrette della tabella ROI-POI del database locale;

**Attributi:**

- + COLUMN\_POIID : String {readOnly}  
Valore della colonna poiId. Valore di default "poiId"
- + COLUMN\_ROIID : String {readOnly}  
Valore della colonna roiId. Valore di default "roiId"
- + ROIPOI\_TABLE\_CREATE : String {readOnly}  
Query per la creazione della tabella
- + TABLE\_NAME : String {readOnly}  
Nome della tabella. Valore di default "ROIPOI"

#### 4.4.53 model::dataaccess::dao::RoiPoiDao



**Figura 71:** Interfaccia RoiPoiDao

**Nome:** RoiPoiDao;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per rendere indipendenti l'invocazione dei metodi per effettuare le operazioni CRUD sulla tabella "ROIPOI" del database locale dalla loro implementazione;

**Descrizione:** Interfaccia che espone i metodi per un DAO per accedere alla tabella "ROIPOI" del database locale;

**Metodi:**

- + *deleteRoiPoisWherePoi(poi : int) : void*

Metodo che permette la rimozione delle associazioni tra un ROI e i POI ad esso associato dalla tabella "ROIPOI" del database locale

**Argomenti:**

- poi : int

Identificativo del POI di cui rimuovere le associazioni con i ROI dal database locale

- + *deleteRoiPoisWhereRoi(roi : int) : void*

Metodo che permette la rimozione delle associazioni tra un POI e i ROI ad esso associato dalla tabella "ROIPOI" del database locale

**Argomenti:**

- roi : int

Identificativo del ROI di cui rimuovere le associazioni con i POI dal database locale

- + *findAllPointsWithRoi(roi : int) : int[]*

Metodo per recuperare tutti gli identificativi dei POI associati ad un ROI

**Argomenti:**

- roi : int

Identificativo del ROI di cui recuperare gli identificativi di tutti i POI associati

- + *findAllRegionsWithPoi(poi : int) : int[]*

Metodo per recuperare tutti gli identificativi dei ROI associati ad un POI

**Argomenti:**

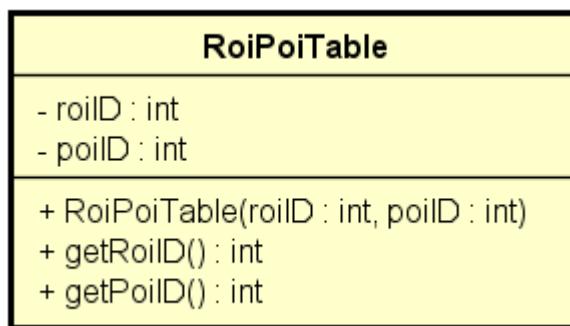
- **poi : int**  
Identificativo del POI di cui recuperare gli identificativi di tutti i ROI associati
- + **insertRoiPoi(toInsert : RoiPoiTable) : int**  
Metodo che permette l'inserimento tra ROI ed POI nel database locale utilizzando un oggetto RoiPoiTable

**Argomenti:**

- **toInsert : RoiPoiTable**  
Oggetto di tipo RoiPoiTable che contiene le associazioni tra ROI e POI
- + **updateRoiPoi(poi : int, roi : int, toUpdate : RoiPoiTable) : boolean**  
Metodo per aggiornare le associazioni tra POI e ROI

**Argomenti:**

- **poi : int**  
Identificativo del POI di cui aggiungere una associazione con un ROI
- **roi : int**  
Identificativo del ROI di cui aggiungere una associazione con un POI
- **toUpdate : RoiPoiTable**  
Oggetto che contiene le associazioni tra ROI e POI

**4.4.54 model::dataaccess::dao::RoiPoiTable****Figura 72:** Classe RoiPoiTable**Nome:** RoiPoiTable;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella RoiPoi del database locale ;

**Descrizione:** Classe che rappresenta una ennupla della tabella RoiPoi del database locale;

**Attributi:**

- - poiID : int  
Identificativo del POI
- - roiID : int  
Identificativo del ROI

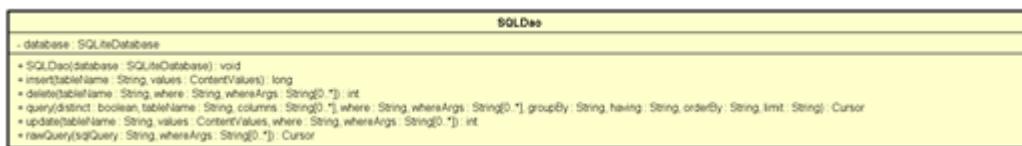
**Metodi:**

- + getPoIID() : int  
Metodo che restituisce l'identificativo del POI
- + getRoiID() : int  
Metodo che restituisce l'identificativo del ROI
- + RoiPoiTable(roiID : int, poiID : int)  
Costruttore della classe RoiPoiTable

**Argomenti:**

- roiID : int  
Identificativo del ROI
- poiID : int  
Identificativo del POI

#### 4.4.55 model::dataaccess::dao::SQLDao



**Figura 73:** Classe SQLDao

**Nome:** SQLDao;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.content.ContentValues (Android);
- android.database.Cursor (Android).

**Visibilità:** public;

**Utilizzo:** Utilizzata dai DAO locali per effettuare operazioni CRUD sul database;

**Descrizione:** Classe che contiene le operazioni di query dirette;

**Attributi:**

- - database : SQLiteDatabase {readOnly}  
Database locale

**Metodi:**

- + delete(tableName : String, where : String, whereArgs : String[]) : int  
Metodo per la rimozione di valori dal database locale. Ritorna il numero delle righe rimosse

**Argomenti:**

- tableName : String  
Nome della tabella su cui eseguire l'operazione
- where : String  
Condizioni utilizzate per filtrare le righe su cui effettuare l'operazione
- whereArgs : String[]  
Valori delle condizioni where

- + insert(tableName : String, values : ContentValues) : long  
Metodo per l'inserimento di valori in una tabella del database locale. Ritorna l'id della riga inserita

**Argomenti:**

- tableName : String  
Nome della tabella su cui effettuare l'operazione
- values : ContentValues  
Valori da inserire nella tabella

- + `query(distinct : boolean, tableName : String, columns : String[], where : String, whereArgs : String[], groupBy : String, having : String, orderBy : String, limit : String) : Cursor`

Metodo per effettuare una query sul database locale

**Argomenti:**

- `distinct : boolean`  
Parametro che indica se applicare o meno la clausola DISTINCT alla query
- `tableName : String`  
Nome della tabella su cui effettuare la query
- `columns : String[]`  
Lista delle colonne da ritornare
- `where : String`  
Condizioni utilizzate per filtrare le righe su cui effettuare l'operazione
- `whereArgs : String[]`  
Valori delle condizioni where
- `groupBy : String`  
Parametro su cui effettuare il raggruppamento dei risultati della query
- `having : String`  
Condizioni utilizzate per filtrare le righe dopo aver applicato la clausola HAVING
- `orderBy : String`  
Parametro su cui effettuare l'ordinamento dei risultati della query
- `limit : String`  
Limite di righe che la query può restituire

- + `rawQuery(sqlQuery : String, whereArgs : String[]) : Cursor`

Metodo per eseguire una query fornendola sotto forma di stringa

**Argomenti:**

- `sqlQuery : String`  
Query da eseguire sotto forma di stringa
- `whereArgs : String[]`  
Argomenti della clausola WHERE

- + SQLDao(database : SQLiteDatabase)

Costruttore della classe SQLDao

**Argomenti:**

- database : SQLiteDatabase

Database locale dell'applicazione

- + update(tableName : String, values : ContentValues, where

: String, whereArgs : String[]) : int

Metodo per l'aggiornamento di valori in una tabella del database locale. Ritorna il numero di righe modificate

**Argomenti:**

- tableName : String

Nome della tabella su cui eseguire l'operazione

- values : ContentValues

Valori aggiornati che sostituiranno i presenti

- where : String

Condizioni utilizzate per filtrare le righe su cui effettuare l'operazione

- whereArgs : String[]

Valori delle condizioni where

#### 4.4.56 model::dataaccess::dao::SQLiteBuildingDao

SQLiteBuildingDao
- sqlDao : SQLDao
+ SQLiteBuildingDao(database : SQLiteDatabase)
+ insertBuilding(toInsert : BuildingTable) : int
+ deleteBuilding(id : int) : void
+ findBuildingByld(id : int) : BuildingTable
+ findBuildingByMajor(major : int) : BuildingTable
+ findAllBuildings() : Collection
+ updateBuilding(id : int, toUpdate : BuildingTable) : boolean
+ cursorToType(cursor : Cursor) : BuildingTable
+ isBuildingMapPresent(major : int) : boolean

**Figura 74:** Classe SQLiteBuildingDao

**Nome:** SQLiteBuildingDao;

**Tipo:** Classe;

**Implementa:**

- BuildingDao.

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "Building" del database locale;

**Descrizione:** Classe che rappresenta un DAO per la tabella "Building" del database locale;

**Metodi:**

- + cursorToType(cursor : Cursor) : BuildingTable

**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "Building" del database locale in un oggetto BuildingTable

**Argomenti:**

- cursor : Cursor

Risultato della query sulla tabella "Building" del database locale

- + deleteBuilding(id : int) : void

**Override** Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "Building" del database locale

**Argomenti:**

- id : int

Identificativo dell'edificio di cui rimuovere le informazioni dal database locale

- + findAllBuildings() : Collection<BuildingTable>

**Override** Metodo che viene utilizzato per recuperare le informazioni di tutti gli edifici presenti nella tabella "Building" del database locale

- + findBuildingById(id : int) : BuildingTable

**Override** Metodo per recuperare le informazioni di un edificio dal database locale tramite il suo identificativo, sotto forma di oggetto BuildingTable

**Argomenti:**

– id : int

Identificativo dell'edificio di cui recuperare le informazioni

- • + findBuildingByMajor(major : int) : BuildingTable  
**Override** Metodo per recuperare le informazioni di un edificio dal database locale tramite il suo major, sotto forma di oggetto BuildingTable

**Argomenti:**

– major : int

Major dell'edificio di cui recuperare le informazioni

- • + insertBuilding(toInsert : BuildingTable) : int  
**Override** Metodo che permette l'inserimento delle informazioni di un edificio in una entry della tabella "Building" del database locale

**Argomenti:**

– toInsert : BuildingTable

Oggetto di tipo BuildingTable che contiene le informazioni dell'edificio

- • + isBuildingMapPresent(major : int) : boolean  
Metodo per verificare la presenza nel database locale delle informazioni di un edificio

**Argomenti:**

– major : int

major dell'edificio

- • + SQLiteBuildingDao(database : SQLiteDatabase)  
Costruttore della classe SQLiteBuildingDao

**Argomenti:**

– database : SQLiteDatabase

Il database locale

- • + updateBuilding(id : int, toUpdate : BuildingTable) : boolean  
**Override** Metodo per aggiornare le informazioni di un edificio nella tabella "Building" del database locale

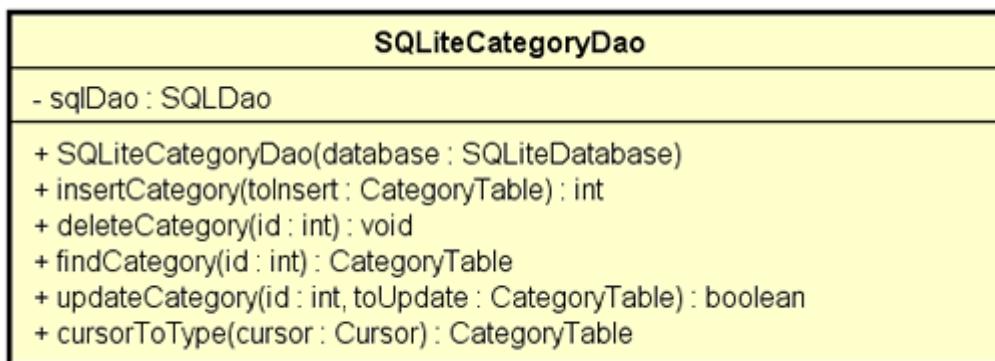
**Argomenti:**

– id : int

Identificativo dell'edificio di cui aggiornare le informazioni

- **toUpdate** : BuildingTable  
Oggetto che contiene le informazioni aggiornate dell’edificio

#### 4.4.57 model::dataaccess::dao::SQLiteCategoryDao



**Figura 75:** Classe SQLiteCategoryDao

**Nome:** SQLiteCategoryDao;

**Tipo:** Classe;

**Implementa:**

- CategoryDao.

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "Category" del database locale;

**Descrizione:** Classe che rappresenta un DAO per la tabella "Category" del database locale;

**Metodi:**

- + cursorToType(cursor : Cursor) : CategoryTable  
**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "Category" del database locale in un oggetto CategoryTable

**Argomenti:**

- `cursor : Cursor`

Risultato della query sulla tabella "Category" del database locale

- • `+ deleteCategory(id : int) : void`

**Override** Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "Category" del database locale

**Argomenti:**

- `id : int`

Identificativo della categoria da rimuovere dal database locale

- • `+ findCategory(id : int) : CategoryTable`

**Override** Metodo per recuperare le informazioni di una categoria dal database locale tramite il suo identificativo, sotto forma di oggetto CategoryTable

**Argomenti:**

- `id : int`

Identificativo della categoria di cui recuperare le informazioni

- • `+ insertCategory(toInsert : CategoryTable) : int`

**Override** Metodo che permette l'inserimento di una categoria nella tabella "Category" del database locale

**Argomenti:**

- `toInsert : CategoryTable`

Oggetto di tipo CategoryTable che contiene le informazioni della categoria

- • `+ SQLiteCategoryDao(database : SQLiteDatabase)`

Costruttore della classe SQLiteCategoryDao

**Argomenti:**

- `database : SQLiteDatabase`

Il database locale

- • `+ updateCategory(id : int, toUpdate : CategoryTable) : boolean`

**Override** Metodo per aggiornare le informazioni di una categoria nella tabella "Category" del database locale

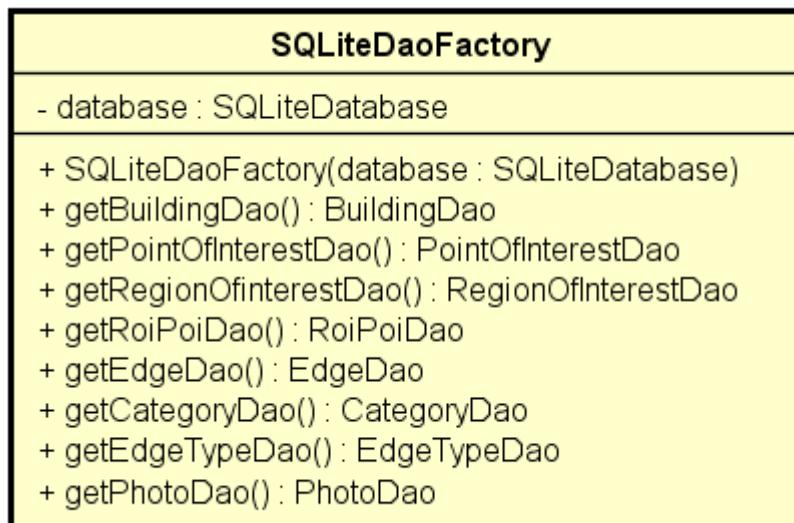
**Argomenti:**

- `id : int`

Identificativo della categoria di cui aggiornare le informazioni

- toUpdate : CategoryTable  
Oggetto che contiene le informazioni aggiornate della categoria

#### 4.4.58 model::dataaccess::dao::SQLiteDaoFactory



**Figura 76:** Classe SQLiteDaoFactory

**Nome:** SQLiteDaoFactory;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per creare e ottenere oggetti DAO locali;

**Descrizione:** Classe che rappresenta la factory per creare tutti gli oggetti DAO locali;

**Attributi:**

- - database : SQLiteDatabase {readOnly}  
Il database locale

**Metodi:**

- + `getBuildingDao() : BuildingDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLiteBuildingDao
- + `getCategoryDao() : CategoryDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLiteCategoryDao
- + `getEdgeDao() : EdgeDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLiteEdgeDao
- + `getEdgeTypeDao() : EdgeTypeDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLiteEdgeTypeDao
- + `getPhotoDao() : PhotoDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLitePhotoDao
- + `getPointOfInterestDao() : PointOfInterestDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLitePointOfInterestDao
- + `getRegionOfInterestDao() : RegionOfInterestDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLiteRegionOfInterestDao
- + `getRoiPoiDao() : RoiPoiDao`  
Metodo che viene utilizzato per ottenere un'istanza di SQLiteRoiPoiDao
- + `SQLiteDaoFactory(database : SQLiteDatabase)`  
Costruttore della classe SQLiteDaoFactory

**Argomenti:**

- `database : SQLiteDatabase`  
Il database locale

#### 4.4.59 model::dataaccess::dao::SQLiteEdgeDao

SQLiteEdgeDao	
-	sqlDao : SQLDao
+	SQLiteEdgeDao(database : SQLiteDatabase)
+	insertEdge(toInsert : EdgeTable) : int
+	deleteEdge(id : int) : void
+	findEdge(id : int) : EdgeTable
+	findAllEdgesOfBuilding(major : int) : Collection
+	updateEdge(id : int, toUpdate : EdgeTable) : boolean
+	cursorToType(cursor : Cursor) : EdgeTable

**Figura 77:** Classe SQLiteEdgeDao

**Nome:** SQLiteEdgeDao;

**Tipo:** Classe;

**Implementa:**

- EdgeDao.

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "Edge" del database locale;

**Descrizione:** Classe che rappresenta un DAO per la tabella "Edge" del database locale;

**Metodi:**

- + cursorToType(cursor : Cursor) : EdgeTable

**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "Edge" del database locale in un oggetto EdgeTable

**Argomenti:**

- cursor : Cursor

Risultato della query sulla tabella "Edge" del database locale

- + `deleteEdge(id : int) : void`

**Override** Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "Edge" del database locale

**Argomenti:**

- `id : int`

Identificativo dell'arco di cui rimuovere le informazioni dal database locale

- + `findAllEdgesOfBuilding(major : int) : Collection<EdgeTable>`

**Override** Metodo che viene utilizzato per recuperare le informazioni di tutti gli archi presenti nella tabella "Edge" del database locale

**Argomenti:**

- `major : int`

Identificativo major dell'edificio di cui si vogliono recuperare tutti gli archi

- + `findEdge(id : int) : EdgeTable`

**Override** Metodo per recuperare le informazioni di un arco dal database locale tramite il suo identificativo, sotto forma di oggetto EdgeTable

**Argomenti:**

- `id : int`

Identificativo dell'arco di cui recuperare le informazioni

- + `insertEdge(toInsert : EdgeTable) : int`

**Override** Metodo che permette l'inserimento delle informazioni di un edificio in una entry della tabella "Edge" del database locale

**Argomenti:**

- `toInsert : EdgeTable`

Oggetto di tipo EdgeTable che contiene le informazioni dell'arco

- + `SQLiteEdgeDao(database : SQLiteDatabase)`

Costruttore della classe SQLiteEdgeDao

**Argomenti:**

- `database : SQLiteDatabase`

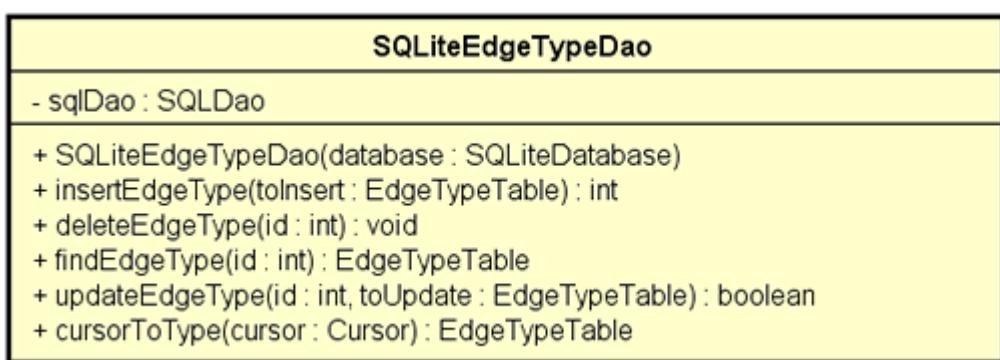
Il database locale

- + `updateEdge(id : int, toUpdate : EdgeTable) : boolean`

**Override** Metodo per aggiornare le informazioni di un edificio nella tabella "Edge" del database locale

**Argomenti:**

- `id : int`  
Identificativo dell'arco di cui aggiornare le informazioni
- `toUpdate : EdgeTable`  
Oggetto che contiene le informazioni aggiornate dell'arco

**4.4.60 model::dataaccess::dao::SQLiteEdgeTypeDao**

**Figura 78:** Classe SQLiteEdgeTypeDao

**Nome:** `SQLiteEdgeTypeDao`;

**Tipo:** Classe;

**Implementa:**

- `EdgeTypeDao`.

**Visibilità:** `public`;

**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "EdgeType" del database locale;

**Descrizione:** Classe che rappresenta un DAO per la tabella "EdgeType" del database locale;

**Metodi:**

- `+ cursorToType(cursor : Cursor) : EdgeTypeTable`  
**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "EdgeType" del database locale in un oggetto `EdgeTypeTable`

**Argomenti:**

– cursor : Cursor

Risultato della query sulla tabella "EdgeType" del database locale

- + deleteEdgeType(id : int) : void

**Override** Metodo che permette la rimozione delle informazioni di un tipo di Edge dalla tabella "EdgeType" del database locale

**Argomenti:**

– id : int

Identificativo del tipo di Edge di cui rimuovere le informazioni dal database locale

- + findEdgeType(id : int) : EdgeTypeTable

**Override** Metodo per recuperare le informazioni di un tipo di Edge dal database locale tramite il suo identificativo, sotto forma di oggetto EdgeTypeTable

**Argomenti:**

– id : int

Identificativo del tipo di Edge di cui recuperare le informazioni

- + insertEdgeType(toInsert : EdgeTypeTable) : int

**Override** Metodo che permette l'inserimento delle informazioni del tipo di Edge in una entry della tabella "EdgeType" del database locale

**Argomenti:**

– toInsert : EdgeTypeTable

Oggetto di tipo EdgeTypeTable che contiene le informazioni di un tipo di Edge

- + SQLiteEdgeTypeDao(database : SQLiteDatabase)

Costruttore della classe SQLiteEdgeTypeDao

**Argomenti:**

– database : SQLiteDatabase

Il database locale

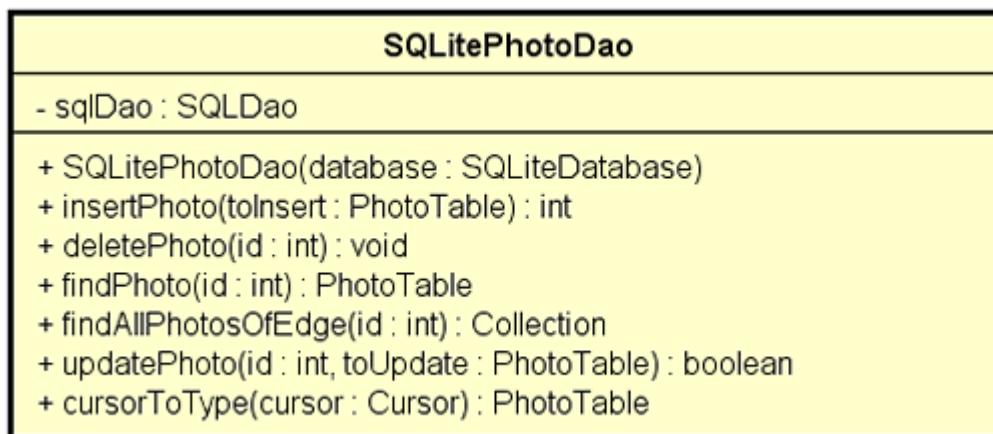
- + updateEdgeType(id : int, toUpdate : EdgeTypeTable) : boolean

**Override** Metodo per aggiornare le informazioni di un tipo di Edge nella tabella "EdgeType" del database locale

**Argomenti:**

- **id : int**  
Identificativo del tipo di Edge di cui aggiornare le informazioni
- **toUpdate : EdgeTypeTable**  
Oggetto che contiene le informazioni aggiornate del tipo di Edge

#### 4.4.61 model::dataaccess::dao::SQLitePhotoDao



**Figura 79:** Classe SQLitePhotoDao

**Nome:** SQLitePhotoDao;

**Tipo:** Classe;

**Implementa:**

- PhotoDao.

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "Photo" del database locale;

**Descrizione:** Classe che rappresenta un DAO per la tabella "Photo" del database locale;

**Metodi:**

- + cursorToType(cursor : Cursor) : PhotoTable

**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "Photo" del database locale in un oggetto PhotoTable

**Argomenti:**

- cursor : Cursor

Risultato della query sulla tabella "Photo" del database locale

- + deletePhoto(id : int) : void

**Override** Metodo che permette la rimozione delle informazioni di una foto dalla tabella "Photo" del database locale

**Argomenti:**

- id : int

Identificativo della foto di cui rimuovere le informazioni dal database locale

- + findAllPhotosOfEdge(id : int) : Collection<PhotoTable>

**Override** Metodo che viene utilizzato per recuperare le informazioni di tutte foto associate ad un Edge presenti nella tabella "Photo" del database locale

**Argomenti:**

- id : int

Identificativo dell'Edge

- + findPhoto(id : int) : PhotoTable

**Override** Metodo per recuperare le informazioni di una foto dal database locale tramite il suo identificativo, sotto forma di oggetto PhotoTable

**Argomenti:**

- id : int

Identificativo della foto

- + insertPhoto(toInsert : PhotoTable) : int

**Override** Metodo che permette l'inserimento delle informazioni di una foto in una entry della tabella "Photo" del database locale

**Argomenti:**

- toInsert : PhotoTable

Oggetto di tipo Photo che contiene le informazioni della foto

- + `SQLitePhotoDao(database : SQLiteDatabase)`

Costruttore della classe SQLitePhotoDao

**Argomenti:**

- `database : SQLiteDatabase`  
Il database locale

- + `updatePhoto(id : int, toUpdate : PhotoTable) : boolean`

**Override** Metodo per aggiornare le informazioni di una foto nella tabella "Photo" del database locale

**Argomenti:**

- `id : int`  
Identificativo della foto di cui aggiornare le informazioni
- `toUpdate : PhotoTable`  
Oggetto che contiene le informazioni aggiornate della foto

#### 4.4.62 model::dataaccess::dao::SQLitePointOfInterestDao

SQLitePointOfInterestDao
- <code>sqlDao : SQLDao</code>
+ <code>SQLitePointOfInterestDao(database : SQLiteDatabase)</code>
+ <code>insertPointOfInterest(toInsert : PointOfInterestTable) : int</code>
+ <code>deletePointOfInterest(id : int) : void</code>
+ <code>findPointOfInterest(id : int) : PointOfInterestTable</code>
+ <code>findAllPointsWithMajor(major : int) : Collection</code>
+ <code>updatePointOfInterest(id : int, toUpdate : PointOfInterestTable) : boolean</code>
+ <code>cursorToType(cursor : Cursor) : PointOfInterestTable</code>

**Figura 80:** Classe SQLitePointOfInterestDao

**Nome:** SQLitePointOfInterestDao;

**Tipo:** Classe;

**Implementa:**

- `PointOfInterestDao.`

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "POI" del database locale;

**Descrizione:** Classe che rappresenta un POI per la tabella "POI" del database locale;

**Metodi:**

- + cursorToType(cursor : Cursor) : PointOfInterestTable  
**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "POI" del database locale in un oggetto PointOfInterestTable

**Argomenti:**

- cursor : Cursor  
Risultato della query sulla tabella "POI" del database locale

- + deletePointOfInterest(id : int) : void  
**Override** Metodo che permette la rimozione delle informazioni di un edificio dalla tabella "POI" del database locale

**Argomenti:**

- id : int  
Identificativo del POI di cui rimuovere le informazioni dal database locale

- + findAllPointsWithMajor(major : int) : Collection<PointOfInterestTable>  
**Override** Metodo che viene utilizzato per recuperare le informazioni di tutti gli edifici presenti nella tabella "POI" del database locale

**Argomenti:**

- major : int  
Identificativo Major associato a tutti i beacon presenti in uno stesso edificio

- + findPointOfInterest(id : int) : PointOfInterestTable  
**Override** Metodo per recuperare le informazioni di un edificio dal database locale tramite il suo identificativo, sotto forma di oggetto PointOfInterestTable

**Argomenti:**

- id : int  
Identificativo del POI di cui recuperare le informazioni

- + insertPointOfInterest(toInsert : PointOfInterestTable) : int

**Override** Metodo che permette l'inserimento delle informazioni di un edificio in una entry della tabella "POI" del database locale

**Argomenti:**

- `toInsert : PointOfInterestTable`

Oggetto di tipo `PointOfInterestTable` che contiene le informazioni dell'edificio

- + `SQLitePointOfInterestDao(database : SQLiteDatabase)`  
Costruttore della classe `SQLitePointOfInterestDao`

**Argomenti:**

- `database : SQLiteDatabase`

Il database locale

- + `updatePointOfInterest(id : int, toUpdate : PointOfInterestTable) : boolean`

**Override** Metodo per aggiornare le informazioni di un edificio nella tabella "POI" del database locale

**Argomenti:**

- `id : int`

Identificativo del POI di cui aggiornare le informazioni

- `toUpdate : PointOfInterestTable`

Oggetto che contiene le informazioni aggiornate del POI

#### 4.4.63 model::dataaccess::dao::SQLiteRegionOfInterestDao

SQLiteRegionOfInterestDao
- <code>sqlDao : SQLDao</code>
+ <code>SQLiteRegionOfInterestDao(database : SQLiteDatabase)</code>
+ <code>insertRegionOfInterest(toInsert : RegionOfInterestTable) : int</code>
+ <code>deleteRegionOfInterest(id : int) : void</code>
+ <code>findRegionOfInterest(id : int) : RegionOfInterestTable</code>
+ <code>findAllRegionsWithMajor(major : int) : Collection</code>
+ <code>updateRegionOfInterest(id : int, toUpdate : RegionOfInterestTable) : boolean</code>
+ <code>cursorToType(cursor : Cursor) : RegionOfInterestTable</code>

**Figura 81:** Classe SQLiteRegionOfInterestDao

**Nome:** `SQLiteRegionOfInterestDao`;

**Tipo:** Classe;

**Implementa:**

- RegionOfInterestDao.

**Visibilità:** public;**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "ROI" del database locale;**Descrizione:** Classe che rappresenta un DAO per la tabella "ROI" del database locale;**Metodi:**

- + cursorToType(cursor : Cursor) : RegionOfInterestTable  
**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "ROI" del database locale in un oggetto RegionOfInterestTable

**Argomenti:**

- cursor : Cursor  
Risultato della query sulla tabella "ROI" del database locale

- + deleteRegionOfInterest(id : int) : void  
**Override** Metodo che permette la rimozione delle informazioni di una RegionOfInterest dalla tabella "ROI" del database locale

**Argomenti:**

- id : int  
Identificativo della RegionOfInterest di cui rimuovere le informazioni dal database locale

- + findAllRegionsWithMajor(major : int) : Collection<RegionOfInterestTable>  
**Override** Metodo che viene utilizzato per recuperare le informazioni di tutte le RegionOfInterest associato ad certo edificio, dato il major dell'edificio

**Argomenti:**

- major : int  
Major dell'edificio

- + findRegionOfInterest(id : int) : RegionOfInterestTable  
**Override** Metodo per recuperare le informazioni di una RegionOfInterest dal database locale tramite il suo identificativo, sotto forma di oggetto RegionOfInterestTable

**Argomenti:**

– `id : int`

Identificativo della RegionOfInterest di cui recuperare le informazioni

- • `+ insertRegionOfInterest(toInsert : RegionOfInterestTable) : int`

**Override** Metodo che permette l'inserimento delle informazioni di una RegionOfInterest in una entry della tabella "ROI" del database locale

**Argomenti:**

– `toInsert : RegionOfInterestTable`

Oggetto di tipo RegionOfInterestTable che contiene le informazioni della RegionOfInterest

- • `+ SQLiteRegionOfInterestDao(database : SQLiteDatabase)`  
Costruttore della classe SQLiteRegionOfInterestDao

**Argomenti:**

– `database : SQLiteDatabase`

Il database locale

- • `+ updateRegionOfInterest(id : int, toUpdate : RegionOfInterestTable) : boolean`

**Override** Metodo per aggiornare le informazioni di una RegionOfInterest nella tabella "ROI" del database locale

**Argomenti:**

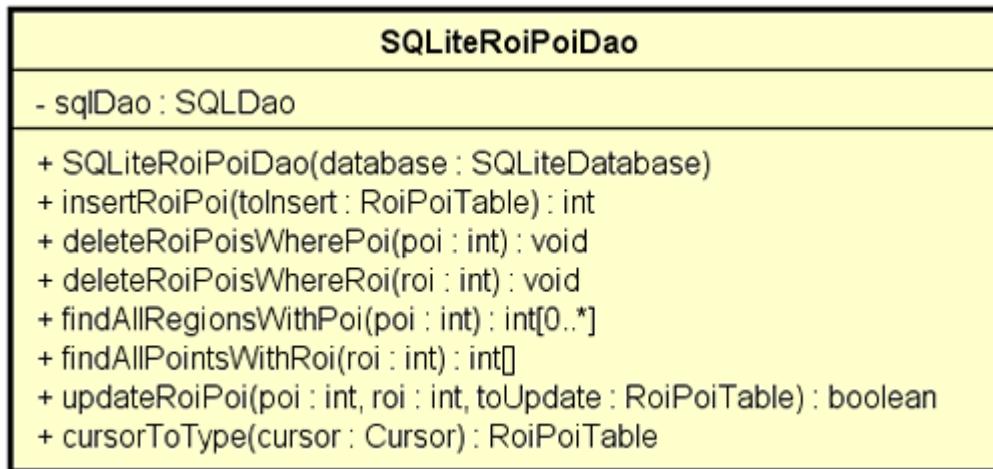
– `id : int`

Identificativo della RegionOfInterest di cui aggiornare le informazioni

– `toUpdate : RegionOfInterestTable`

Oggetto che contiene le informazioni aggiornate della RegionOfInterest

#### 4.4.64 model::dataaccess::dao::SQLiteRoiPoiDao



**Figura 82:** Classe SQLiteRoiPoiDao

**Nome:** SQLiteRoiPoiDao;

**Tipo:** Classe;

**Implementa:**

- RoiPoiDao.

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per effettuare le operazioni CRUD sulla tabella "ROIPOI" del database locale;

**Descrizione:** Classe che rappresenta un DAO per la tabella "ROIPOI" del database locale;

**Metodi:**

- + cursorToType(cursor : Cursor) : RoiPoiTable

**Override** Metodo che viene utilizzato per convertire il risultato della query sulla tabella "ROIPOI" del database locale in un oggetto RoiPoiTable

**Argomenti:**

– `cursor : Cursor`

Risultato della query sulla tabella "ROIPOI" del database locale

- + `deleteRoiPoisWherePoi(poi : int) : void`

**Override** Metodo che permette la rimozione delle associazioni tra un ROI e i POI ad esso associato dalla tabella "ROIPOI" del database locale

**Argomenti:**

– `poi : int`

Identificativo del POI di cui rimuovere le associazioni con i ROI dal database locale

- + `deleteRoiPoisWhereRoi(roi : int) : void`

**Override** Metodo che permette la rimozione delle associazioni tra un POI e i ROI ad esso associato dalla tabella "ROIPOI" del database locale

**Argomenti:**

– `roi : int`

Identificativo del ROI di cui rimuovere le associazioni con i POI dal database locale

- + `findAllPointsWithRoi(roi : int) : int[]`

**Override** Metodo per recuperare tutti gli identificativi dei POI associati ad un ROI

**Argomenti:**

– `roi : int`

Identificativo del ROI di cui recuperare gli identificativi di tutti i POI associati

- + `findAllRegionsWithPoi(poi : int) : int[]`

**Override** Metodo per recuperare tutti gli identificativi dei ROI associati ad un POI

**Argomenti:**

– `poi : int`

Identificativo del POI di cui recuperare gli identificativi di tutti i ROI associati

- + `insertRoiPoi(toInsert : RoiPoiTable) : int`

**Override** Metodo che permette l'inserimento tra ROI ed POI nel database locale utilizzando un oggetto RoiPoiTable

**Argomenti:**

- **toInsert** : RoiPoiTable  
Oggetto di tipo RoiPoiTable che contiene le associazioni tra ROI e POI
- + **SQLiteRoiPoiDao(database : SQLiteDatabase)**  
Costruttore della classe SQLiteRoiPoiDao

**Argomenti:**

- **database** : SQLiteDatabase  
Il database locale
- + **updateRoiPoi(poi : int, roi : int, toUpdate : RoiPoiTable) : boolean**  
**Override** Metodo per aggiornare le associazioni tra POI e ROI

**Argomenti:**

- **poi** : int  
Identificativo del POI di cui aggiungere una associazione con un ROI
- **roi** : int  
Identificativo del ROI di cui aggiungere una associazione con un POI
- **toUpdate** : RoiPoiTable  
Oggetto che contiene le associazioni tra ROI e POI

**4.4.65 model::dataaccess::service::BuildingService****Figura 83:** Classe BuildingService**Nome:** BuildingService;**Tipo:** Classe;**Implementa:**

- DatabaseService.

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** Viene utilizzata come layer Service tra gli oggetti BuildingMap e gli oggetti DAO corrispettivi;

**Descrizione:** Classe che rappresenta il layer Service tra gli oggetti Building-Map e gli oggetti DAO corrispettivi;

**Attributi:**

- - databaseURL : String  
URL del database remoto
- - edgeService : EdgeService  
Oggetto che si pone come layer Service tra gli oggetti EnrichedEdge e gli oggetti DAO corrispettivi
- - poiService : PointOfInterestService  
Oggetto che si pone come layer Service tra gli oggetti PointOfInterest e gli oggetti DAO corrispettivi
- - remoteBuildingDao : RemoteBuildingDao  
Oggetto di utility per la conversione da JSON a BuildingTable
- - roiService : RegionOfInterestService  
Oggetto che si pone come layer Service tra gli oggetti RegionOfInterest e gli oggetti DAO corrispettivi
- - sqliteBuildingDao : SQLiteBuildingDao  
Oggetto che rappresenta un DAO per la tabella "Building" del database locale

**Metodi:**

- + BuildingService(dbURL : String, sqliteBuilding : SQLiteDao, remoteBuilding : RemoteBuildingDao, roiService : RegionOfInterestService, poiService : PointOfInterestService,

`edgeService : EdgeService)`  
Costruttore della classe BuildingService

**Argomenti:**

- `dbURL : String`  
URL del database remoto
- `sqliteBuilding : SQLiteDatabase`  
Oggetto che rappresenta un DAO per la tabella "Building" del database locale
- `remoteBuilding : RemoteBuildingDao`  
Oggetto di utility per la conversione da JSON a Building-Table
- `roiService : RegionOfInterestService`  
Oggetto che si pone come layer Service tra gli oggetti RegionOfInterest e gli oggetti DAO corrispettivi
- `poiService : PointOfInterestService`  
Oggetto che si pone come layer Service tra gli oggetti PointOfInterest e gli oggetti DAO corrispettivi
- `edgeService : EdgeService`  
Oggetto che si pone come layer Service tra gli oggetti EnrichedEdge e gli oggetti DAO corrispettivi

• + `convertAndInsert(object : JsonObject) : void`

Metodo per la conversione di un JsonObject in un oggetto BuildingTable, che verrà inserito nel database locale

**Argomenti:**

- `object : JsonObject`  
Oggetto JsonObject che contiene le informazioni di un edificio

• + `deleteBuilding(id : int) : void`

**Override** Metodo per rimuovere la mappa di un edificio dal database locale

**Argomenti:**

- `id : int`  
Identificativo dell'edificio da rimuovere

• + `findAllBuildings() : Collection<BuildingTable>`

**Override** Metodo per recuperare le informazioni di tutte le mappe degli edifici presenti nel database locale

- + `findAllRemoteBuildings()` : `Collection<BuildingTable>`  
**Override** Metodo per recuperare le informazioni di tutte le mappe degli edifici presenti nel database remoto
- + `findBuildingByMajor(major : int)` : `BuildingMap`  
**Override** Metodo per recuperare la mappa di un edificio ricercandola nel database locale

**Argomenti:**

- `major : int`  
Major dell'edificio

- + `findRemoteBuildingByMajor(major : int)` : `BuildingMap`  
**Override** Metodo per recuperare la mappa di un edificio ricercandola nel database remoto

**Argomenti:**

- `major : int`  
Major dell'edificio

- - `fromTableToBo(buildingTable : BuildingTable)` : `BuildingMap`  
Metodo per la costruzione di oggetto BuildingMap a partire da un BuildingTable

**Argomenti:**

- `buildingTable : BuildingTable`  
Oggetto contenente le informazioni dell'edificio

- + `isBuildingMapPresent(major : int)` : `boolean`  
**Override** Metodo per verificare la presenza di una mappa di un edificio nel database locale

**Argomenti:**

- `major : int`  
Major dell'edificio

- + `isBuildingMapUpdated(major : int)` : `boolean`  
**Override** Metodo per verificare se la mappa di un edificio è aggiornata all'ultima versione

**Argomenti:**

- `major : int`  
Major dell'edificio

- - `retrieveAndInsertMap(major : int)` : `void`  
Metodo per scaricare la mappa di un edificio dal database remoto ed inserirla nel database locale

**Argomenti:**

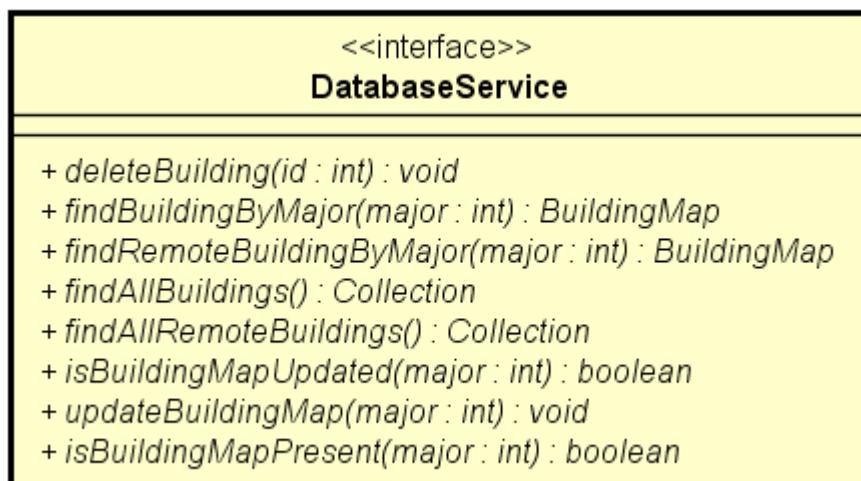
– major : int  
Major dell'edificio

- + updateBuildingMap(major : int) : void

**Override** Metodo per aggiornare la mappa di un edificio all'ultima versione disponibile

**Argomenti:**

– major : int  
Major dell'edificio

**4.4.66 model::dataaccess::service::DatabaseService**

**Figura 84:** Interfaccia DatabaseService

**Nome:** DatabaseService;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'accesso alle mappe dal modo di recuperarle;

**Descrizione:** Interfaccia che espone tutti i metodi per l'accesso alle mappe contenute nel database locale o remoto;

**Metodi:**

- + *deleteBuilding(id : int) : void*

Metodo per cancellare una mappa a partire dall'identificativo di un edificio

**Argomenti:**

- *id : int*

Identificativo numerico di un oggetto BuildingMap

- + *findAllBuildings() : Collection<BuildingTable>*

Metodo che ritorna la lista di tutti gli oggetti BuildingTable presenti nel database locale

- + *findAllRemoteBuildings() : Collection<BuildingTable>*

Metodo che ritorna la lista di tutti gli oggetti BuildingTable presenti nel database remoto

- + *findBuildingByMajor(major : int) : BuildingMap*

Metodo per il recupero di un oggetto BuildingMap da un database locale o remoto tramite l'identificativo Major uguale in tutti i beacon presenti in uno stesso edificio

**Argomenti:**

- *major : int*

Identificativo major uguale per tutti i beacon presenti in uno stesso edificio

- + *findRemoteBuildingByMajor(major : int) : BuildingMap*

Metodo per effettuare il download di una mappa dal database remoto a partire dall'identificativo major uguale per tutti i beacon presenti in un certo edificio

**Argomenti:**

- *major : int*

Identificativo major uguale per tutti i beacon presenti in uno stesso edificio

- + *isBuildingMapPresent(major : int) : boolean*

Metodo per verificare la presenza di una mappa di un edificio nel database locale

**Argomenti:**

- *major : int*

Major dell'edificio

- + *isBuildingMapUpdated(major : int) : boolean*

Metodo per verificare se la mappa di un edificio è aggiornata all'ultima versione

**Argomenti:**

- major : int  
Major dell'edificio

- + *updateBuildingMap(major : int) : void*

Metodo per aggiornare la mappa di un edificio all'ultima versione disponibile

**Argomenti:**

- major : int  
Major dell'edificio

**4.4.67 model::dataaccess::service::EdgeService**

**Figura 85:** Classe EdgeService

**Nome:** EdgeService;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;

**Utilizzo:** Viene utilizzata come layer Service tra gli oggetti EnrichedEdge e gli oggetti DAO corrispondenti;

**Descrizione:** Classe che rappresenta il layer Service tra gli oggetti EnrichedEdge e gli oggetti DAO corrispondenti;

**Attributi:**

- - **photoService** : PhotoService  
Oggetto che si pone come layer Service tra gli oggetti PhotoRef e gli oggetti DAO corrispettivi
- - **remoteEdgeDao** : RemoteEdgeDao  
Oggetto di utility per la conversione da JSON a EdgeTable
- - **remoteEdgeTypeDao** : RemoteEdgeTypeDao  
Oggetto di utility per la conversione da JSON a EdgeTypeTable
- - **roiService** : RegionOfInterestService  
Oggetto che si pone come layer Service tra gli oggetti RegionOfInterest e gli oggetti DAO corrispettivi
- - **sqliteEdgeDao** : SQLiteEdgeDao  
Oggetto che rappresenta un DAO per la tabella "Edge" del database locale
- - **sqliteEdgeTypeDao** : SQLiteEdgeTypeDao  
Oggetto che rappresenta un DAO per la tabella "EdgeType" del database locale

**Metodi:**

- + **convertAndInsert(object : JsonObject) : void**  
Metodo per la conversione di un JsonObject in un oggetto EdgeTable, che verrà inserito nel database locale

**Argomenti:**

- object : JsonObject  
Oggetto JsonObject che contiene le informazioni di un Edge

- + **convertAndInsertEdgeType(object : JsonObject) : void**  
Metodo per la conversione di un JsonObject in un oggetto EdgeTypeTable, che verrà inserito nel database locale

**Argomenti:**

- object : JsonObject  
Oggetto JsonObject che contiene le informazioni di un tipo di Edge

- + **deleteEdge(id : int) : void**  
Metodo per rimuovere un Edge dal database locale

**Argomenti:**

- id : int  
Identificativo numerico dell'Edge da rimuovere

- + **EdgeService**(sqliteEdge : SQLiteEdgeDao, remoteEdge : RemoteEdgeDao, sqliteEdgeType : SQLiteEdgeTypeDao, remoteEdgeType : RemoteEdgeTypeDao, photoService : PhotoService, roiService : RegionOfInterestService)  
Costruttore della classe EdgeService

**Argomenti:**

- **sqliteEdge** : SQLiteEdgeDao  
Oggetto che rappresenta un DAO per la tabella "Edge" del database locale
- **remoteEdge** : RemoteEdgeDao  
Oggetto di utility per la conversione da JSON a EdgeTable
- **sqliteEdgeType** : SQLiteEdgeTypeDao  
Oggetto che rappresenta un DAO per la tabella "Edge-Type" del database locale
- **remoteEdgeType** : RemoteEdgeTypeDao  
Oggetto di utility per la conversione da JSON a EdgeTypeTable
- **photoService** : PhotoService  
Oggetto che si pone come layer Service tra gli oggetti PhotoRef e gli oggetti DAO corrispettivi
- **roiService** : RegionOfInterestService  
Oggetto che si pone come layer Service tra gli oggetti RegionOfInterest e gli oggetti DAO corrispettivi

- + **findAllEdgesOfBuilding**(major : int) : Collection<EnrichedEdge>  
Metodo per recuperare le informazioni di tutti gli Edge di un edificio, dato il major dell'edificio

**Argomenti:**

- **major** : int  
Major dell'edificio

- + **findEdge**(id : int) : EnrichedEdge  
Metodo per recuperare un Edge ricercandolo nel database locale

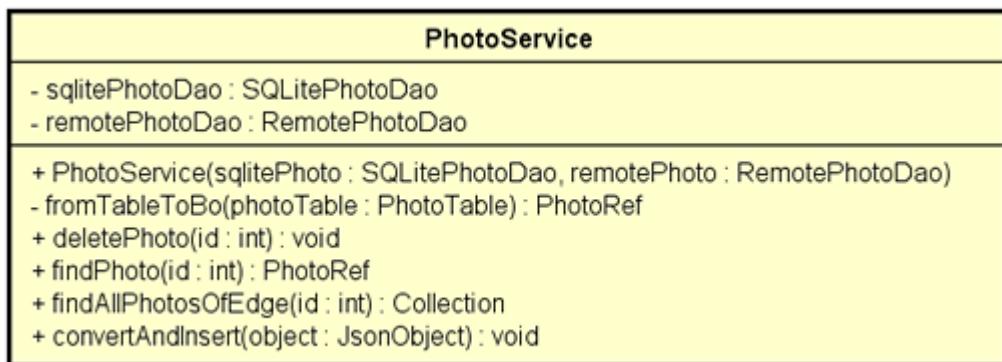
**Argomenti:**

- **id** : int  
Identificativo numerico dell'Edge da ricercare

- - **fromTableToBo**(edgeTable : EdgeTable) : EnrichedEdge  
Metodo per la costruzione di oggetto EnrichedEdge a partire da un EdgeTable

**Argomenti:**

- edgeTable : EdgeTable  
Oggetto contenente le informazioni di un Edge

**4.4.68 model::dataaccess::service::PhotoService****Figura 86:** Classe PhotoService**Nome:** PhotoService;**Tipo:** Classe;**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;**Utilizzo:** Viene utilizzata come layer Service tra gli oggetti PhotoRef e gli oggetti DAO corrispettivi;**Descrizione:** Classe che rappresenta il layer Service tra gli oggetti PhotoRef e gli oggetti DAO corrispettivi;**Attributi:**

- - remotePhotoDao : RemotePhotoDao  
Oggetto di utility per la conversione da JSON a PhotoTable

- - **sqlitePhotoDao** : `SQLitePhotoDao`  
Oggetto che rappresenta un DAO per la tabella "Photo" del database locale

**Metodi:**

- + **convertAndInsert(object : JsonObject) : void**  
Metodo per la conversione di un JsonObject in un oggetto PhotoTable, che verrà inserito nel database locale

**Argomenti:**

- **object : JsonObject**  
Oggetto JsonObject che contiene le informazioni di una foto

- + **deletePhoto(id : int) : void**  
Metodo per rimuovere una foto di un Edge dal database locale

**Argomenti:**

- **id : int**  
Identificativo numerico della foto da rimuovere

- + **findAllPhotosOfEdge(id : int) : Collection<PhotoRef>**  
Metodo per recuperare tutte le foto di un Edge dal database locale

**Argomenti:**

- **id : int**  
Identificativo dell'Edge di cui si vuole recuperare tutte le foto

- + **findPhoto(id : int) : PhotoRef**  
Metodo per recuperare una foto ricercandola nel database locale

**Argomenti:**

- **id : int**  
Identificativo numerico della foto da recuperare

- - **fromTableToBo(photoTable : PhotoTable) : PhotoRef**  
Metodo per la costruzione di oggetto PhotoRef a partire da un PhotoTable

**Argomenti:**

- **photoTable : PhotoTable**  
Oggetto contenente le informazioni della foto

- + **PhotoService(sqlitePhoto : SQLitePhotoDao, remotePhoto : RemotePhotoDao)**  
Costruttore della classe PhotoService

**Argomenti:**

- **sqlitePhoto** : **SQLitePhotoDao**  
Oggetto che rappresenta un DAO per la tabella "Photo" del database locale
- **remotePhoto** : **RemotePhotoDao**  
Oggetto di utility per la conversione da JSON a PhotoTable

**4.4.69 model::dataaccess::service::PointOfInterestService****Figura 87:** Classe PointOfInterestService**Nome:** PointOfInterestService;**Tipo:** Classe;**Componenti delle librerie utilizzate:**

- com.google.api.client.json.JsonObjectParser ;
- com.google.gson.JsonArray;
- com.google.gson.JsonElement;
- com.google.gson.JsonObject.

**Visibilità:** public;**Utilizzo:** Viene utilizzata come layer Service tra gli oggetti PointOfInterest e gli oggetti DAO corrispondenti;**Descrizione:** Classe che rappresenta il layer Service tra gli oggetti PointOfInterest e gli oggetti DAO corrispondenti;**Attributi:**

- – **remoteCategoryDao** : **RemoteCategoryDao**  
Oggetto di utility per la conversione da JSON a CategoryTable

- - `remotePointOfInterestDao` : `RemotePointOfInterestDao`  
Oggetto di utility per la conversione da JSON a PointOfInterestTable
- - `remoteRoiPoiDao` : `RemoteRoiPoiDao`  
Oggetto di utility per la conversione da JSON a RoiPoiTable
- - `sqliteCategoryDao` : `SQLiteCategoryDao`  
Oggetto che rappresenta un DAO per la tabella "Category" del database locale
- - `sqlitePointOfInterestDao` : `SQLitePointOfInterestDao`  
Oggetto che rappresenta un DAO per la tabella "POI" del database locale
- - `sqliteRoiPoiDao` : `SQLiteRoiPoiDao`  
Oggetto che rappresenta un DAO per la tabella "ROIPOI" del database locale

**Metodi:**

- + `convertAndInsert(object : JsonObject) : void`  
Metodo per la conversione di un JsonObject in un oggetto PointOfInterestTable, che verrà inserito nel database locale

**Argomenti:**

- `object : JsonObject`  
Oggetto JsonObject che contiene le informazioni di un PointOfInterest

- + `convertAndInsertCategory(object : JsonObject) : void`  
Metodo per la conversione di un JsonObject in un oggetto CategoryTable, che verrà inserito nel database locale

**Argomenti:**

- `object : JsonObject`  
Oggetto JsonObject che contiene le informazioni di una categoria

- + `convertAndInsertRoiPoi(object : JsonObject) : void`  
Metodo per la conversione di un JsonObject in un oggetto RoiPoiTable, che verrà inserito nel database locale

**Argomenti:**

- `object : JsonObject`  
Oggetto JsonObject che contiene le stesse informazioni di un oggetto EdgeType

- + **deletePointOfInterest(id : int) : void**  
Metodo per rimuovere un PointOfInterest dal database locale

**Argomenti:**

- **id : int**  
Identificativo numerico del PointOfInterest da rimuovere

- + **findAllPointsWithMajor(major : int) : Collection<PointOfInterest>**  
Metodo per recuperare le informazioni di tutti i PointOfInterest di un edificio, dato il major dell'edificio

**Argomenti:**

- **major : int**  
Major dell'edificio

- + **findPointOfInterest(id : int) : PointOfInterest**  
Metodo per recuperare un PointOfInterest ricercandolo nel database locale

**Argomenti:**

- **id : int**  
Identificativo numerico del PointOfInterest da recuperare

- - **fromTableToBo(pointOfInterestTable : PointOfInterestTable) : PointOfInterest**  
Metodo per la costruzione di oggetto PointOfInterest a partire da un PointOfInterestTable

**Argomenti:**

- **pointOfInterestTable : PointOfInterestTable**  
Oggetto contenente le informazioni del PointOfInterest

- + **PointOfInterestService(sqlitePOI : SQLitePointOfInterestDao, remotePOI : RemotePointOfInterestDao, sqliteRoiPoi : SQLiteRoiPoiDao, remoteRoiPoi : RemoteRoiPoiDao, sqliteCategory : SQLiteCategoryDao, remoteCategory : RemoteCategoryDao)**  
Costruttore della classe PointOfInterestService

**Argomenti:**

- **sqlitePOI : SQLitePointOfInterestDao**  
Oggetto che rappresenta un DAO per la tabella "POI" del database locale
- **remotePOI : RemotePointOfInterestDao**  
Oggetto di utility per la conversione da JSON a PointOfInterestTable

- **sqliteRoiPoi** : `SQLiteRoiPoiDao`  
Oggetto che rappresenta un DAO per la tabella "ROI-POI" del database locale
- **remoteRoiPoi** : `RemoteRoiPoiDao`  
Oggetto di utility per la conversione da JSON a RoiPoi-Table
- **sqliteCategory** : `SQLiteCategoryDao`  
Oggetto che rappresenta un DAO per la tabella "Category" del database locale
- **remoteCategory** : `RemoteCategoryDao`  
Oggetto di utility per la conversione da JSON a CategoryTable

#### 4.4.70 model::dataaccess::service::RegionOfInterestService



**Figura 88:** Classe RegionOfInterestService

**Nome:** `RegionOfInterestService`;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `com.google.api.client.json.JsonObjectParser` ;
- `com.google.gson.JsonArray`;
- `com.google.gson.JsonElement`;
- `com.google.gson.JsonObject`.

**Visibilità:** `public`;

**Utilizzo:** Viene utilizzata come layer Service tra gli oggetti `RegionOfInterest` e gli oggetti DAO corrispondenti;

**Descrizione:** Classe che rappresenta il layer Service tra gli oggetti RegionOfInterest e gli oggetti DAO corrispettivi;

**Attributi:**

- - `remoteRegionOfInterestDao` : `RemoteRegionOfInterestDao`  
Oggetto di utility per la conversione da JSON a RegionOfInterestTable
- - `remoteRoiPoiDao` : `RemoteRoiPoiDao`  
Oggetto di utility per la conversione da JSON a RoiPoiTable
- - `sqliteRegionOfInterestDao` : `SQLiteRegionOfInterestDao`  
Oggetto che rappresenta un DAO per la tabella "ROI" del database locale
- - `sqliteRoiPoiDao` : `SQLiteRoiPoiDao`  
Oggetto che rappresenta un DAO per la tabella "ROIPOI" del database locale

**Metodi:**

- + `convertAndInsert(object : JsonObject) : void`  
Metodo per la conversione di un JsonObject in un oggetto RegionOfInterestTable, che verrà inserito nel database locale

**Argomenti:**

- `object` : `JsonObject`  
Oggetto JsonObject che contiene le informazioni di una RegionOfInterest

- + `deleteRegionOfInterest(id : int) : void`

Metodo per rimuovere una RegionOfInterest dal database locale

**Argomenti:**

- `id` : `int`  
Identificativo numerico della RegionOfInterest da rimuovere

- + `findAllRegionsWithMajor(major : int) : Collection<RegionOfInterest>`

Metodo per recuperare le informazioni di tutte le RegionOfInterest di un edificio, dato il major dell'edificio

**Argomenti:**

- `major` : `int`  
Major dell'edificio

- + **findRegionOfInterest(id : int) : RegionOfInterest**  
Metodo per recuperare una RegionOfInterest ricercandola nel database locale

**Argomenti:**

- **id : int**

Identificativo numerico della RegionOfInterest da recuperare

- - **fromTableToBo(regionOfInterestTable : RegionOfInterestTable) : RegionOfInterest**

Metodo per la costruzione di oggetto RegionOfInterest a partire da un RegionOfInterestTable

**Argomenti:**

- **regionOfInterestTable : RegionOfInterestTable**  
Oggetto contenente le informazioni della RegionOfInterest

- + **RegionOfInterestService(sqliteROI : SQLiteRegionOfInterestDao, remoteROI : RemoteRegionOfInterestDao, sqliteRoiPoi : SQLiteRoiPoiDao, remoteRoiPoi : RemoteRoiPoiDao)**

Costruttore della classe RegionOfInterestService

**Argomenti:**

- **sqliteROI : SQLiteRegionOfInterestDao**

Oggetto che rappresenta un DAO per la tabella "ROI" del database locale

- **remoteROI : RemoteRegionOfInterestDao**

Oggetto di utility per la conversione da JSON a RegionOfInterestTable

- **sqliteRoiPoi : SQLiteRoiPoiDao**

Oggetto che rappresenta un DAO per la tabella "ROI-POI" del database locale

- **remoteRoiPoi : RemoteRoiPoiDao**

Oggetto di utility per la conversione da JSON a RoiPoiTable

#### 4.4.71 model::dataaccess::service::ServiceHelper



**Figura 89:** Classe ServiceHelper

**Nome:** ServiceHelper;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** Viene utilizzata per ottenere un'istanza di DatabaseService;

**Descrizione:** Classe che rappresenta un aiutante per la costruzione di un DatabaseService;

**Metodi:**

- + getService(sqliteDaoFactory : SQLiteDaoFactory, remoteDaoFactory : RemoteDaoFactory, dbURL : String) : DatabaseService  
Metodo che viene utilizzato per ottenere un'istanza di DatabaseService

**Argomenti:**

- sqliteDaoFactory : SQLiteDaoFactory  
Un oggetto SQLiteDaoFactory necessaria per la creazione degli oggetti DAO locali
- remoteDaoFactory : RemoteDaoFactory  
Un oggetto RemoteDaoFactory necessario per la creazione degli oggetti DAO remoti
- dbURL : String  
L'URL del database remoto

#### 4.4.72 model::navigator::BuildingInformation

BuildingInformation
- name : String {readOnly} - description : String {readOnly} - openingHours : String {readOnly} - address : String {readOnly}
+ BuildingInformation(name : String, description : String, timetable : String, address : String) + getName() : String + getDescription() : String + getOpeningHours() : String + getAddress() : String + toString() : String

**Figura 90:** Classe BuildingInformation

**Nome:** BuildingInformation;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v4.content.LocalBroadcastManager (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per accedere alle informazioni associate ad un edificio come il nome, l'orario di apertura, una descrizione e l'indirizzo;

**Descrizione:** Classe che rappresenta le informazioni di un edificio;

**Attributi:**

- - address : String {readOnly}  
Indirizzo dell'edificio
- - description : String {readOnly}  
Descrizione dell'edificio
- - name : String {readOnly}  
Nome dell'edificio
- - openingHours : String {readOnly}  
Orari di apertura dell'edificio

**Metodi:**

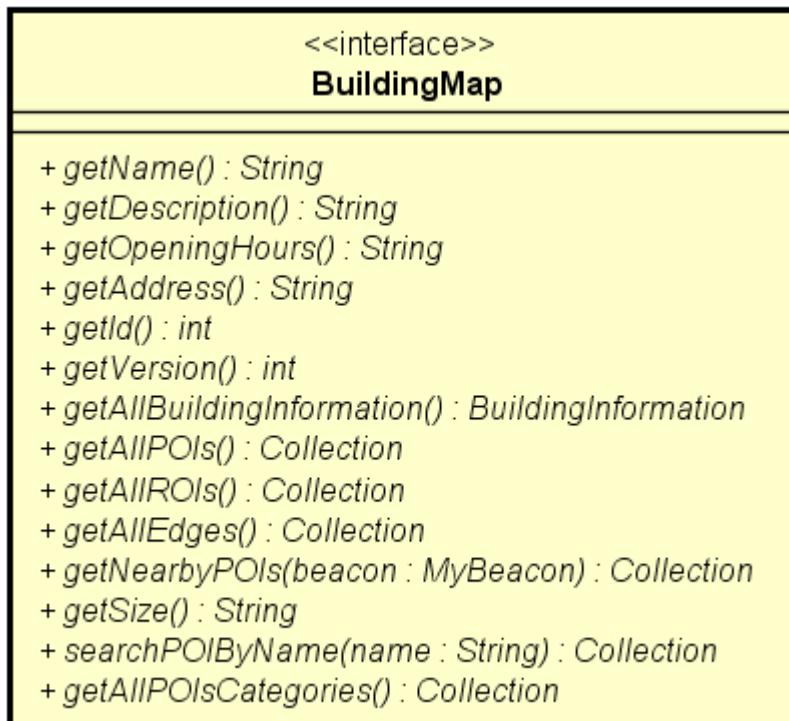
- + BuildingInformation(name : String, description : String, openingHours : String, address : String)  
Costruttore della classe BuildingInformation

**Argomenti:**

- name : String  
Nome dell'edificio
- description : String  
Descrizione dell'edificio
- openingHours : String  
Orari di apertura dell'edificio
- address : String  
Indirizzo dell'edificio

- + `getAddress() : String`  
Metodo che ritorna l'indirizzo dell'edificio al quale tale oggetto è associato
- + `getDescription() : String`  
Metodo che ritorna la descrizione dell'edificio al quale tale oggetto è associato
- + `getName() : String`  
Metodo che ritorna il nome dell'edificio al quale tale oggetto è associato
- + `getOpeningHours() : String`  
Metodo che ritorna gli orari dell'edificio al quale tale oggetto è associato
- + `toString() : String`  
Metodo che ritorna tutte le informazioni dell'edificio al quale tale oggetto è associato sottoforma di stringa

#### 4.4.73 model::navigator::BuildingMap



**Figura 91:** Interfaccia BuildingMap

**Nome:** BuildingMap;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'accesso alle informazioni di un edificio dal modo in cui vengono implementate e gestite;

**Descrizione:** Interfaccia che espone i metodi per l'accesso alle informazioni di un edificio;

**Metodi:**

- + *getAddress() : String*  
Metodo che ritorna l'indirizzo dell'edificio a cui l'oggetto è associato.
- + *getAllBuildingInformation() : BuildingInformation*  
Metodo che ritorna un oggetto BuildingInformation contenente tutte le informazioni dell'edificio a cui è associato.
- + *getAllEdges() : Collection<EnrichedEdge>*  
Metodo che ritorna la collezione di tutti gli archi previsti nella rappresentazione a grafo di un edificio.
- + *getAllPOIs() : Collection<PointOfInterest>*  
Metodo che ritorna la collezione di tutti i POI presenti in un edificio.
- + *getAllPOIsCategories() : Collection<String>*  
Metodo che ritorna una collezione di stringhe, eventualmente vuota, che rappresentano le categorie di appartenenza dei POI
- + *getAllROIs() : Collection<RegionOfInterest>*  
Metodo che ritorna la collezione di tutti i ROI presenti in un edificio.
- + *getDescription() : String*  
Metodo che ritorna una descrizione dell'edificio a cui l'oggetto è associato.
- + *getId() : int*  
Metodo che l'identificativo numerico della mappa all'interno di un database.
- + *getName() : String*  
Metodo che restituisce il nome dell'edificio a cui è associato tale oggetto.

- + *getNearbyPOIs(beacon : MyBeacon) : Collection<PointOfInterest>*  
Metodo che ritorna la collezione di POI associati alla ROI che contiene il beacon passato come argomento.

**Argomenti:**

- *beacon : MyBeacon*

Beacon associato alla RegionOfInterest di cui si vogliono conoscere l'insieme di POI che contiene.

- + *getOpeningHours() : String*

Metodo che ritornagli orari di apertura dell'edificio a cui l'oggetto è associato.

- + *getSize() : String*

Metodo che ritorna la dimensione della mappa dell'edificio (espressa in MB)

- + *getVersion() : int*

Metodo che ritorna l'identificativo numerico della mappa.

- + *searchPOIByName(name : String) : Collection<PointOfInterest>*

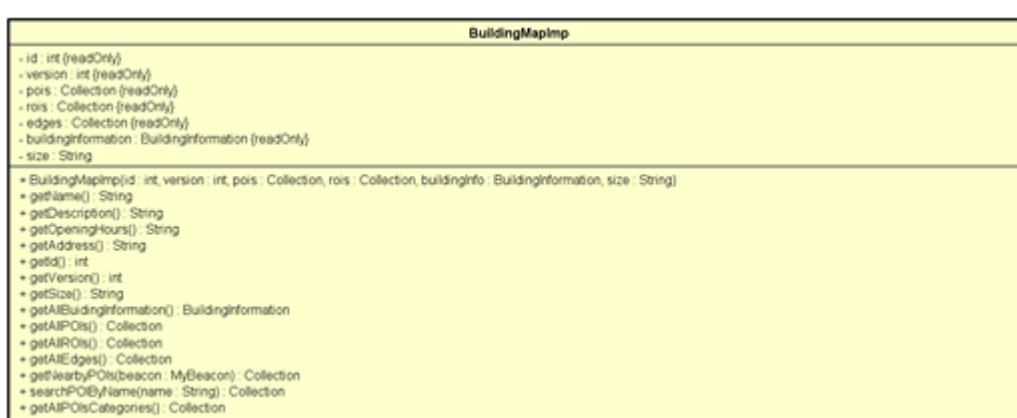
Metodo che permette di cercare i POI di un edificio in cui nome contiene la stringa passata come parametro. Ritorna una collezione, eventualmente vuota, di oggetti PointOfInterest nel cui nome contengono la stringa passata come parametro

**Argomenti:**

- *name : String*

Stringa da cercare nei POI dell'edificio

#### 4.4.74 model::navigator::BuildingMapImp



**Figura 92:** Classe BuildingMapImp

**Nome:** BuildingMapImp;

**Tipo:** Classe;

**Implementa:**

- BuildingMap.

**Componenti delle librerie utilizzate:**

- android.content.Intent (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per accedere alle informazioni riguardanti la mappa di un edificio, alle informazioni generali dell'edificio stesso e per accedere alle collezioni di archi, POI e ROI in cui è stato decomposto un edificio;

**Descrizione:** Classe che rappresenta la mappa di un edificio con tutte le informazioni ad esso associate;

**Attributi:**

- - buildingInformation : BuildingInformation  
Informazioni riguardanti l'edificio rappresentato dalla mappa
- - edges : Collection<EnrichedEdge> {readOnly}  
Insieme di archi che indicano i possibili percorsi tra due ROI
- - id : int {readOnly}  
Identificativo univoco dell'edificio
- - pois : Collection<PointOfInterest> {readOnly}  
Insieme di POI appartenenti all'edificio rappresentato dalla mappa
- - rois : Collection<RegionOfInterest> {readOnly}  
Insieme di ROI appartenenti all'edificio rappresentato dalla mappa
- - size : String  
Dimensione della mappa dell'edificio (in MB)
- - version : int {readOnly}  
Versione corrente della mappa

**Metodi:**

- + BuildingMapImp(id : int, version : int, pois : Collection<PointOfInterest>, rois : Collection<RegionOfInterest>, buildingInfo : BuildingInformation, size : String)  
Costruttore della classe BuildingMapImp

**Argomenti:**

- **id** : int  
Identificativo dell'edificio
  - **version** : int  
Versione della mappa
  - **pois** : Collection<PointOfInterest>  
Tutti i POI appartenenti all'edificio
  - **rois** : Collection<RegionOfInterest>  
Tutte le ROI appartenente all'edificio
  - **buildingInfo** : BuildingInformation  
Informazioni dell'edificio
  - **size** : String  
Dimensione della mappa dell'edificio (espressa in MB)
- + **getAddress()** : String  
**Override** Metodo che ritorna l'indirizzo dell'edificio a cui l'oggetto è associato
  - + **getAllBuildingInformation()** : BuildingInformation  
**Override** Metodo che ritorna un oggetto BuildingInformation contenente tutte le informazioni dell'edificio a cui è associato.
  - + **getAllEdges()** : Collection<EnrichedEdge>  
**Override** Metodo che ritorna la collezione di tutti gli archi previsti nella rappresentazione a grafo di un edificio
  - + **getAllNearbyPOIs(beacon : MyBeacon)** : Collection<PointOfInterest>  
**Override** Metodo che ritorna la collezione di POI associati alla ROI che contiene il beacon passato come argomento

**Argomenti:**

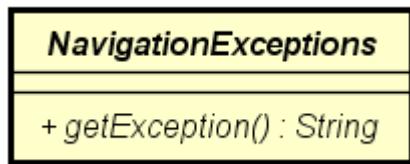
- **beacon** : MyBeacon  
Beacon associato alla RegionOfInterest di cui si vogliono conoscere l'insieme di POI che contiene
- + **getAllPOIs()** : Collection<PointOfInterest>  
**Override** Metodo che ritorna la collezione di tutti i POI presenti in un edificio
  - + **getAllPOIsCategories()** : Collection<String>  
**Override** Metodo che ritorna una collezione di stringhe, eventualmente vuota, che rappresentano le categorie di appartenenza dei POI

- + `getAllROIs() : Collection<RegionOfInterest>`  
**Override** Metodo che ritorna la collezione di tutti i ROI presenti in un edificio
- + `getDescription() : String`  
**Override** Metodo che ritorna una descrizione dell'edificio a cui l'oggetto è associato
- + `getId() : int`  
**Override** Metodo che l'identificativo numerico della mappa all'interno di un database
- + `getName() : String`  
**Override** Metodo che restituisce il nome dell'edificio a cui è associato tale oggetto
- + `getOpeningHours() : String`  
**Override** Metodo che ritornagli orari di apertura dell'edificio a cui l'oggetto è associato
- + `getSize() : String`  
**Override** Metodo che ritorna la dimensione della mappa dell'edificio (espressa in MB)
- + `getVersion() : int`  
**Override** Metodo che ritorna l'identificativo numerico della mappa
- + `searchPOIByName(name : String) : Collection<PointOfInterest>`  
**Override** Metodo che permette di cercare i POI di un edificio in cui nome contiene la stringa passata come parametro. Ritorna una collezione, eventualmente vuota, di oggetti PointOfInterest nel cui nome contengono la stringa passata come parametro

**Argomenti:**

- `name : String`  
Stringa da cercare nei POI dell'edificio

#### 4.4.75 model::navigator::NavigationExceptions



**Figura 93:** Classe astratta NavigationExceptions

**Nome:** *NavigationExceptions*;

**Tipo:** Classe astratta;

**Componenti delle librerie utilizzate:**

- org.altbeacon.beacon.BeaconParser (AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per fornire una implementazione di base dei metodi derivati da java.lang.Exception;

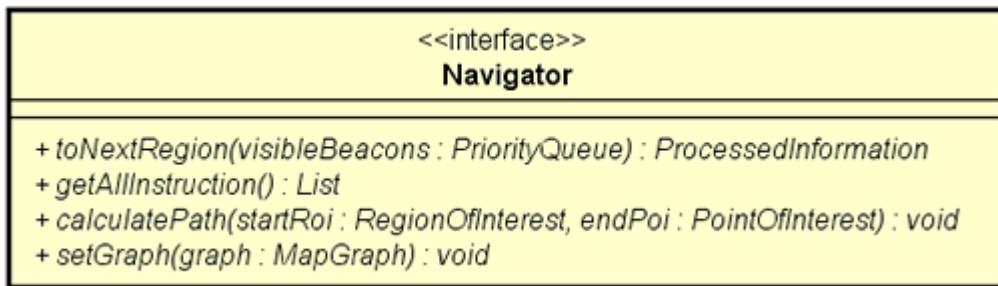
**Descrizione:** Classe base per le eccezioni che possono essere lanciate durante la navigazione. Estende java.lang.Exception;

**Metodi:**

- **+ getException() : String**

Metodo che ritorna una stringa che rappresenta il motivo per cui è stata lanciata l'eccezione

#### 4.4.76 model::navigator::Navigator



**Figura 94:** Interfaccia Navigator

**Nome:** Navigator;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente il modo in cui viene implementata la navigazione dal modo in cui è possibile usufruirne;

**Descrizione:** Interfaccia che espone i metodi per accedere alle funzionalità di navigazione;

**Metodi:**

- + `calculatePath(startRoi : RegionOfInterest, endRoi : RegionOfInterest) : void`

Metodo che calcola un percorso tra due RegionOfInterest viste come vertici di un grafo MapGraph utilizzando un oggetto PathFinder. Il punto di partenza e il punto di arrivo sono i parametri richiesti in input dal metodo mentre il grafo è un campo dati. Viene lanciata un'eccezione di tipo NoGraphSetException nel caso in cui non sia settato alcun grafo.

**Argomenti:**

- `startRoi : RegionOfInterest`  
Punto di partenza del percorso.
- `endRoi : RegionOfInterest`  
Punto di arrivo del percorso.

- + `getAllInstructions()` : `List<ProcessedInformation>`  
Metodo che ritorna la lista completa delle ProcessedInstruction da seguire per percorrere un percorso calcolato.
- + `setGraph(graph : MapGraph)` : `void`  
Metodo per settare il grafo sul quale calcolare il percorso.

**Argomenti:**

- `graph` : `MapGraph`  
Grafo sul quale si vogliono calcolare dei percorsi.

- + `toNextRegion(visibleBeacons : PriorityQueue<MyBeacon>)` : `ProcessedInformation`  
Metodo che ritorna le informazioni da seguire per raggiungere la prossima RegionOfInterest. Le informazioni fornite dipendono dalla lista di beacon passata come parametro in ingrasso e dal beacon più potente tra quelli in essa contenuti. Viene lanciata una eccezione di tipo NoNavigationInformationException nel caso in cui si cerchi di accedere a tale metodo senza prima aver calcolato un percorso di navigazione. Viene lanciata una eccezione di tipo PathException nel caso in cui il beacon più potente nella lista di beacon in ingrasso sia associato ad una RegionOfInterest non appartenente ad una di quelle presenti nel percorso calcolato.

**Argomenti:**

- `visibleBeacons` : `PriorityQueue<MyBeacon>`  
Insieme di beacon visibili al momento della chiamata al metodo.

#### 4.4.77 model::navigator::NavigatorImp

NavigatorImp	
- path : List	
- progress : Iterator	
- buildingGraph : MapGraph	
- pathFinder : PathFinder	
- compass : Compass	
+ NavigatorImp(compass : Compass)	
+ toNextRegion(visibleBeacons : PriorityQueue) : ProcessedInformation	
+ getAllInstruction() : List	
+ calculatePath(startRoi : RegionOfInterest, endPoi : PointOfInterest) : void	
+ setGraph(graph : MapGraph) : void	
+ getPath() : List	
- isShorter(firstPath : List, secondPath : List) : boolean	
- getStarterInformation() : String	
- getMostPowerfulBeacon(beacons : PriorityQueue) : MyBeacon	
- createInformation(edge : EnrichedEdge) : ProcessedInformation	

**Figura 95:** Classe NavigatorImp

**Nome:** NavigatorImp;

**Tipo:** Classe;

**Implementa:**

- Navigator.

**Componenti delle librerie utilizzate:**

- org.altbeacon.beacon.BeaconManager (AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per calcolare un percorso da seguire durante la navigazione e per recuperare da quest'ultimo tutte le informazioni necessarie da fornire all'utente per seguire tale percorso;

**Descrizione:** Classe che si occupa della navigazione;

**Attributi:**

- - **buildingGraph** : `MapGraph`  
Grafo dell'edificio in cui si desidera navigare
- - **compass** : `Compass`  
Sensore di tipo Compass utilizzato all'avvio della navigazione
- - **path** : `List<EnrichedEdge>`  
Lista di EnrichedEdge rappresentanti le indicazioni da seguire per raggiungere la destinazione
- - **pathFinder** : `PathFinder`  
Campo dati che si occupa del calcolo del percorso
- - **progress** : `Iterator`  
Iteratore rappresentante il punto che è stato raggiunto durante la navigazione

#### Metodi:

- + **calculatePath(startRoi : RegionOfInterest, endRoi : RegionOfInterest) : void**  
**Override** Metodo che calcola un percorso tra due RegionOfInterest viste come vertici di un grafo MapGraph utilizzando un oggetto PathFinder. Il punto di partenza e il punto di arrivo sono i parametri richiesti in input dal metodo mentre il grafo è un campo dati. Viene lanciata un eccezione di tipo NoGraphSetException nel caso in cui non sia settato alcun grafo
- **Argomenti:**
  - **startRoi** : `RegionOfInterest`  
Punto di partenza del percorso
  - **endRoi** : `RegionOfInterest`  
Punto di arrivo del percorso
- - **createInformation(edge : EnrichedEdge) : ProcessedInformation**  
Metodo che crea le ProcessedInformation in base al tipo di arco e in base alle informazioni provenienti dal beacon e da eventuali sensori utilizzati
- **Argomenti:**
  - **edge** : `EnrichedEdge`  
Edge di cui devono essere recuperate le informazioni
- + **getAllInstructions() : List<ProcessedInformation>**  
**Override** Metodo che ritorna la lista completa delle ProcessedInstruction da seguire per percorrere un percorso calcolato

- - `getMostPowerfullBeacon(beacons : PriorityQueue<MyBeacon> : MyBeacon)`  
Metodo che ritorna il beacon con potenza maggiore tra quelli rilevati

**Argomenti:**

- `beacons : PriorityQueue<MyBeacon>`  
Queue dei beacon rilevati

- + `getPath() : List<EnrichedEdge>`  
Metodo per ottenere la lista di EnrichedEdge rappresentanti un percorso
- - `getStarterInformation() : String`  
Metodo che ritorna le prime informazioni utili alla navigazione
- - `isShorter(firstPath : List<EnrichedEdge>, secondPath : List<EnrichedEdge>) : boolean`  
Metodo per determinare se un percorso è più lungo o più breve di un altro percorso

**Argomenti:**

- `firstPath : List<EnrichedEdge>`  
Lista di EnrichedEdge rappresentante un percorso
  - `secondPath : List<EnrichedEdge>`  
Lista di EnrichedEdge rappresentante un percorso

- + `NavigatorImp(compass : Compass)`  
Costruttore della classe NavigatorImp

**Argomenti:**

- `compass : Compass`  
Sensore di tipo Compass utilizzato durante la navigazione

- + `setGraph(graph : MapGraph) : void`  
**Override** Metodo per settare il grafo sul quale calcolare il percorso

**Argomenti:**

- `graph : MapGraph`  
Grafo sul quale si vogliono calcolare dei percorsi

- + `toNextRegion(visibleBeacons : PriorityQueue<MyBeacon> : ProcessedInformation)`  
**Override** Metodo che ritorna le informazioni da seguire per raggiungere la prossima RegionOfInterest. Le informazioni fornite

dipendono dalla lista di beacon passata come parametro in ingresso e dal beacon più potente tra quelli in essa contenuti. Viene lanciata una eccezione di tipo NoNavigationInformationException nel caso in cui si cerchi di accedere a tale metodo senza prima aver calcolato un percorso di navigazione. Viene lanciata una eccezione di tipo PathException nel caso in cui il beacon più potente nella lista di beacon in ingresso sia associato ad una RegionOfInterest non appartenente ad una di quelle presenti nel percorso calcolato

**Argomenti:**

- visibleBeacons : PriorityQueue<MyBeacon>  
Insieme di beacon visibili al momento della chiamata al metodo

**4.4.78 model::navigator::NoGraphSetException**

<b>NoGraphSetException</b>	
- DEFAULT_MESSAGE : String	
- message : String {readOnly}	
+ getException() : String	
+ NoGraphSetException()	
+ NoGraphSetException(message : String)	

**Figura 96:** Classe NoGraphSetException

**Nome:** NoGraphSetException;

**Tipo:** Classe;

**Estende:**

- NavigationExceptions.

**Componenti delle librerie utilizzate:**

- org.altbeacon.beacon.RangeNotifier (AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per lanciare una eccezione nel caso in cui si cerchi di accedere alle funzioni di navigazione offerte da Navigator senza aver settato un grafo;

**Descrizione:** Classe che rappresenta una eccezione che può essere lanciata durante l'utilizzo della classe Navigator;

**Attributi:**

- - DEFAULT\_MESSAGE : String {readOnly}  
Messaggio di default associato all'eccezione nel caso in cui si avvi la navigazione senza aver settato il grafo
- - message : String {readOnly}  
Stringa che rappresenta l'eccezione

**Metodi:**

- + getException() : String  
**Override** Metodo che ritorna il messaggio che rappresenta l'eccezione
- + NoGraphSetException()  
Costruttore di default della classe NoGraphSetException
- + NoGraphSetException(message : String)  
Costruttore della classe NoGraphSetException

**Argomenti:**

- message : String  
Messaggio che rappresenta l'eccezione

#### 4.4.79 model::navigator::NoNavigationInformationException

NoNavigationInformationException
<u>- DEFAULT_MESSAGE : String</u> <u>- message : String {readOnly}</u>
+ <u>getException() : String</u> + <u>NoNavigationInformationException()</u> + <u>NoNavigationInformationException(message : String)</u>

**Figura 97:** Classe NoNavigationInformationException

**Nome:** NoNavigationInformationException;

**Tipo:** Classe;

**Estende:**

- NavigationExceptions.

**Componenti delle librerie utilizzate:**

- org.altbeacon.beacon.BeaconConsumer(AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per lanciare un'eccezione nel caso in cui si voglia accedere alle informazioni di navigazione ma non è ancora stato calcolato un percorso;

**Descrizione:** Classe che rappresenta una eccezione che può essere lanciata durante l'utilizzo della classe Navigator;

**Attributi:**

- - DEFAULT\_MESSAGE : String {readOnly}  
Messaggio di default associato all'eccezione nel caso in cui si avvi la navigazione senza aver settato il grafo
- - message : String {readOnly}  
Stringa che rappresenta l'eccezione

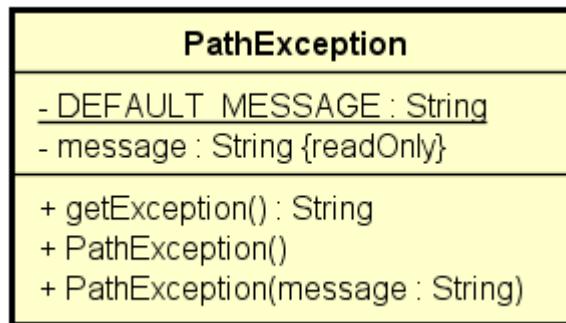
**Metodi:**

- + getException() : String  
**Override** Metodo che ritorna il messaggio che rappresenta l'eccezione
- + NoNavigationInformatioonException()  
Costruttore di default della classe NoNavigationInformationException
- + NoNavigationInformatioonException(message : String)  
Costruttore della classe NoNavigationInformationException

**Argomenti:**

- message : String  
Messaggio che rappresenta l'eccezione

#### 4.4.80 model::navigator::PathException



**Figura 98:** Classe PathException

**Nome:** PathException;

**Tipo:** Classe;

**Estende:**

- NavigationExceptions.

**Componenti delle librerie utilizzate:**

- org.altbeacon.beacon.Ragion (AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per lanciare una eccezione nel caso in cui si rilevino dei beacon non previsti dal percorso calcolato;

**Descrizione:** Classe che rappresenta una eccezione che può essere lanciata durante l'utilizzo della classe Navigator;

**Attributi:**

- - `DEFAULT_MESSAGE : String {readOnly}`  
Messaggio di default associato all'eccezione nel caso in cui si avvia la navigazione senza aver settato il grafo
- - `message : String {readOnly}`  
Stringa che rappresenta l'eccezione

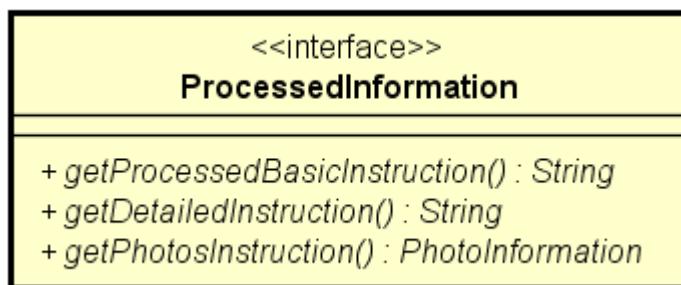
**Metodi:**

- + `getException() : String`  
Override Metodo che ritorna il messaggio che rappresenta l'eccezione
- + `PathException()`  
Costruttore di default della classe PathException
- + `PathException(message : String)`  
Costruttore della classe PathException

**Argomenti:**

- `message : String`  
Messaggio che rappresenta l'eccezione

#### 4.4.81 model::navigator::ProcessedInformation



**Figura 99:** Interfaccia ProcessedInformation

**Nome:** ProcessedInformation;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione delle informazioni di navigazione e la loro costruzione dall'utilizzo di quest'ultime;

**Descrizione:** Interfaccia che espone i metodi per l'accesso alle informazioni di navigazione, pronte per essere restituite ad un utilizzatore di tali informazioni;

**Metodi:**

- + *getDetailedInstruction()* : *String*  
Metodo che ritorna le istruzioni dettagliate per superare un certo arco nel percorso calcolato.
- + *getPhotoInstruction()* : *PhotoInformation*  
Metodo che ritorna un oggetto PhotoInformation con il quale è possibile accedere alle fotografie che ritraggono l'arco da superare nel percorso calcolato
- + *getProcessedBasicInstruction()* : *String*  
Metodo che ritorna le istruzioni basilari per superare un certo arco nel percorso calcolato.

#### 4.4.82 model::navigator::ProcessedInformationImp

ProcessedInformationImp
- basic : String - detailed : String - photos : List
+ ProcessedInformationImp(edge : EnrichedEdge) + ProcessedInformation(edge : EnrichedEdge, starterInformation : String) + getProcessedBasicInstruction() : String + getDetailedInstruction() : String + getPhotosInstruction() : PhotoInformation

**Figura 100:** Classe ProcessedInformationImp

**Nome:** ProcessedInformationImp;

**Tipo:** Classe;

**Implementa:**

- ProcessedInformation.

**Componenti delle librerie utilizzate:**

- org.altbeacon.beacon.startup.BootstrapNotifier (AltBeacon).

**Visibilità:** public;

**Utilizzo:** È utilizzata per accedere ai vari tipi di informazioni forniti quali informazioni di base, informazioni dettagliate e le foto dei luoghi da raggiungere;

**Descrizione:** Classe che rappresenta le informazioni di navigazione pronte per essere restituite ad un eventuale utilizzatore;

**Attributi:**

- - `basic : String`  
Informazioni di base di un Edge
- - `detailed : String`  
Indicazioni dettagliate di un Edge
- - `photos : List<String>`  
Lista di foto di un Edge

**Metodi:**

- + `getDetailedInstruction() : String`  
**Override** Metodo che ritorna le istruzioni dettagliate per superare un certo arco nel percorso calcolato
- + `getPhotoInstruction() : PhotoInformation`  
**Override** Metodo che ritorna un oggetto PhotoInformation con il quale è possibile accedere alle fotografie che ritraggono l'arco da superare nel percorso calcolato
- + `getProcessedBasicInstruction() : String`  
**Override** Metodo che ritorna le istruzioni basilari per superare un certo arco nel percorso calcolato
- + `ProcessedInformationImp(edge : EnrichedEdge)`  
Costruttore della classe ProcessedInformationImp

**Argomenti:**

- `edge : EnrichedEdge`  
Edge da cui devono essere estratte le informazioni
- + `ProcessedInformationImp(edge : EnrichedEdge, starterInformation : String)`  
Costruttore della classe ProcessedInformationImp

**Argomenti:**

- `edge : EnrichedEdge`  
Edge da cui devono essere estratte le informazioni

- **starterInformation : String**  
Informazioni aggiuntive per costruire le informazioni associate ad un arco del percorso per superarlo

#### 4.4.83 model::navigator::algorithm::DijkstraPathFinder



**Figura 101:** Classe DijkstraPathFinder

**Nome:** DijkstraPathFinder;

**Tipo:** Classe;

**Estende:**

- PathFinder.

**Visibilità:** public;

**Utilizzo:** È utilizzata per calcolare il percorso di navigazione sfruttando l'algoritmo di Dijkstra. Per fare questo utilizza la classe org.jgrapht.alg.DijkstraShortestPath<V,E>;

**Descrizione:** Classe che rappresenta un algoritmo per il calcolo del percorso di navigazione;

**Metodi:**

- + calculatePath(MapGraph, RegionOfInterest, RegionOfInterest) : List<EnrichedEdge>  
**Override** Metodo che calcola un percorso tra due RegionOfInterest viste come vertici di un grafo MapGraph. Il grafo, il punto di partenza e il punto di arrivo sono i parametri richiesti in input dal metodo

**Argomenti:**

- graph : MapGraph  
Grafo sul quale calcolare il percorso

- **startROI** : `RegionOfInterest`  
RegionOfInterest di partenza del percorso. Deve appartenere al grafo passato come parametro.
  - **endROI** : `RegionOfInterest`  
RegionOfInterest di arrivo del percorso. Deve appartenere al grafo passato come parametro.
- + **DijkstraPathFinder()**  
Costruttore di default della classe DijkstraPathFinder

#### 4.4.84 model::navigator::algorithm::PathFinder



**Figura 102:** Interfaccia PathFinder

**Nome:** PathFinder;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'utilizzo dei risultati del calcolo di un percorso dal modo in cui questo viene effettivamente calcolato. È particolarmente utile nel caso in cui si voglia cambiare algoritmo di calcolo del percorso sul grafo;

**Descrizione:** Interfaccia che espone i metodi per calcolare un percorso di navigazione;

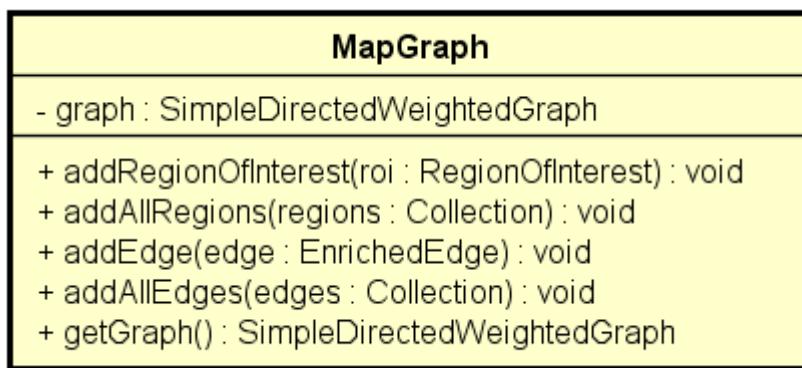
**Metodi:**

- + `calculatePath(graph : MapGraph, startRoi : RegionOfInterest, endRoi : RegionOfInterest) : List<EnrichedEdge>`  
Metodo che calcola un percorso tra due RegionOfInterest viste come vertici di un grafo MapGraph. Il grafo, il punto di partenza e il punto di arrivo sono i parametri richiesti in input dal metodo.

**Argomenti:**

- **graph** : MapGraph  
Grafo sul quale calcolare il percorso.
- **startRoi** : RegionOfInterest  
Punto di partenza del percorso.
- **endRoi** : RegionOfInterest  
Punto di arrivo del percorso.

#### 4.4.85 model::navigator::graph::MapGraph



**Figura 103:** Classe MapGraph

**Nome:** MapGraph;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rappresentare un grafo sul quale applicare un algoritmo per il calcolo di un percorso nel package algorithm. Al suo interno per la rappresentazione del grafo utilizza la classe org.jgrapht.graph.SimpleDirectedWeightedGraph;

**Descrizione:** Classe che rappresenta un grafo da utilizzare per il calcolo del percorso di navigazione;

**Attributi:**

- - **graph** : SimpleDirectedWeightedGraph<RegionOfInterest,EnrichedEdge>  
Rappresentazione a grafo dell'edificio

**Metodi:**

- + `addAllEdges(edges : Collection<EnrichedEdge>) : void`  
Metodo che permette di aggiungere più archi al grafo che rappresenta l'edificio

**Argomenti:**

- `edges : Collection<EnrichedEdge>`  
Archi da aggiungere al grafo che rappresenta l'edificio

- + `addAllRegions(regions : Collection<RegionOfInterest>) : void`

Metodo che permette di aggiungere più RegionOfInterest al grafo che rappresenta l'edificio

**Argomenti:**

- `regions : Collection<RegionOfInterest>`  
Collezione di RegionOfInterest da aggiungere al grafo che rappresenta l'edificio

- + `addEdge(edge : EnrichedEdge) : void`

Metodo che permette di aggiungere un arco al grafo che rappresenta l'edificio

**Argomenti:**

- `edge : EnrichedEdge`  
Arco da aggiungere al grafo che rappresenta l'edificio

- + `addRegionOfInterest(roi : RegionOfInterest) : void`

Metodo che permette di aggiungere una RegionOfInterest al grafo che rappresenta l'edificio

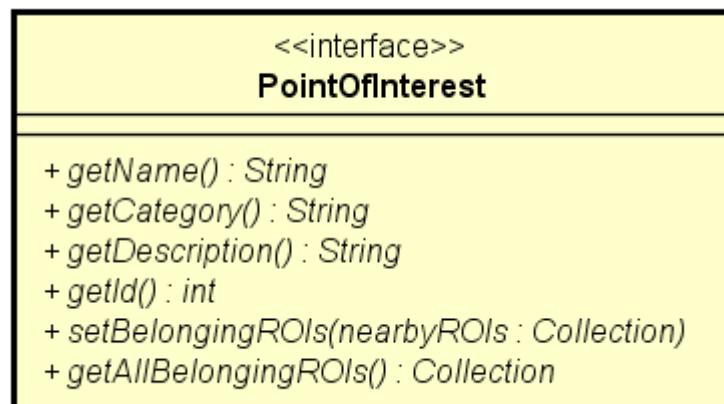
**Argomenti:**

- `roi : RegionOfInterest`  
RegionOfInterest da aggiungere al grafo che rappresenta l'edificio

- + `getGraph() : SimpleDirectedWeightedGraph<RegionOfInterest, EnrichedEdge>`

Metodo che

#### 4.4.86 model::navigator::graph::area::PointOfInterest



**Figura 104:** Interfaccia PointOfInterest

**Nome:** PointOfInterest;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione delle informazioni di un POI dal loro utilizzo;

**Descrizione:** Interfaccia che rappresenta un'area di interesse all'interno di un edificio. Espone i metodi per accedere alle informazioni di quest'area e per settare a quali aree coperte dal segnale di beacon appartiene;

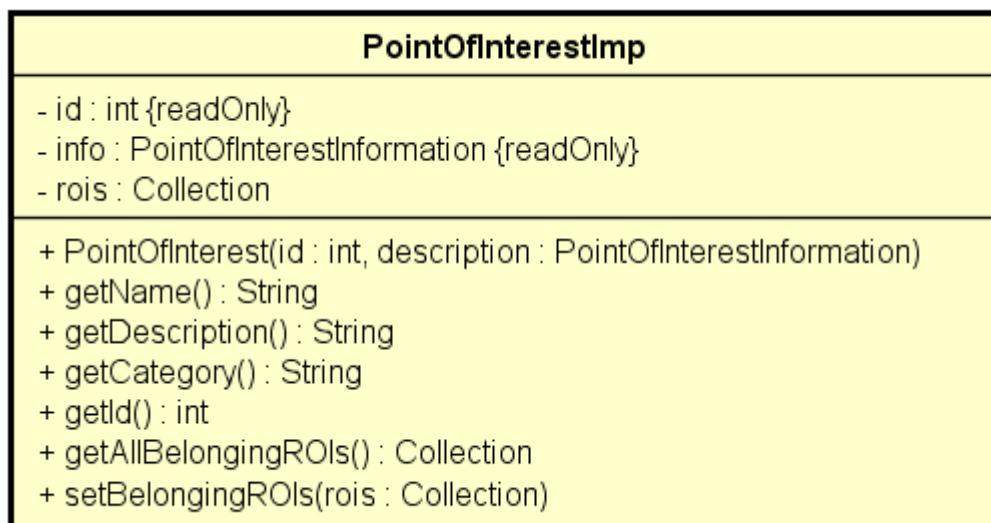
**Metodi:**

- + *getBelongingROIs() : Collection<RegionOfInterest>*  
Metodo che ritorna la collezione di RegionOfInterest alle quali tale oggetto appartiene
- + *getCategory() : String*  
Metodo che ritorna il nome della categoria di appartenenza del PointOfInterest
- + *getDescription() : String*  
Metodo che ritorna una descrizione del PointOfInterest
- + *getId() : int*  
Metodo che ritorna l'identificativo numerico associato al PointOfInterest

- + *getName()* : *String*  
Metodo che ritorna il nome associato al PointOfInterest
- + *setAllBelongingROIs(rois : Collection<RegionOfInterest>)* : *void*  
Metodo che permette di settare l'insieme di RegionOfInterest nelle quali tale PointOfInterest è contenuto

**Argomenti:**

- *rois* : *Collection<RegionOfInterest>*  
Insieme di RegionOfInterest alle quali appartiene il PointOfInterest

**4.4.87 model::navigator::graph::area::PointOfInterestImp****Figura 105:** Classe PointOfInterestImp**Nome:** PointOfInterestImp;**Tipo:** Classe;**Implementa:**

- PointOfInterest.

**Visibilità:** public;

**Utilizzo:** È utilizzata per definire dei punti di partenza e destinazione all'interno di un edificio, permettendo la navigazione tra di essi. Inoltre, viene utilizzata per indicare un punto di interesse nell'edificio e accedere alle sue informazioni;

**Descrizione:** Classe che rappresenta un POI, ossia un punto all'interno di un edificio ritenuto di possibile interesse;

#### Attributi:

- - id : int {readOnly}  
Identificativo numerico di un PointOfInterestImp
- - info : PointOfInterestInformation {readOnly}  
Informazioni relative ad un PointOfInterestImp
- - rois : Collection<RegionOfInterest>  
Collezione degli oggetti RegionOfInterest alle quali appartiene l'oggetto

#### Metodi:

- + getAllBelongingROIs() : Collection<RegionOfInterest>  
**Override** Metodo che ritorna la collezione di RegionOfInterest alle quali tale oggetto appartiene
- + getCategory() : String  
**Override** Metodo che ritorna il nome della categoria di appartenenza del PointOfInterestImp.
- + getDescription() : String  
**Override** Metodo che ritorna una descrizione del PointOfInterestImp.
- + getId() : int  
**Override** Metodo che ritorna l'identificativo numerico associato al PointOfInterestImp.
- + getName() : String  
**Override** Metodo che ritorna il nome associato al PointOfInterestImp
- + PointOfInterestImp(id : int, info : PointOfInterestInformation)  
Costruttore della classe PointOfInterestImp

#### Argomenti:

- id : int  
Identificativo numerico della classe PointOfInterestImp.

- info : PointOfInterestInformation  
Informazioni relative ad un POI
  - + setBelongingROIs(rois : Collection<RegionOfInterest>) : void  
**Override** Metodo che permette di settare l'insieme di RegionOfInterest nelle quali tale PointOfInterestImp è contenuto
- Argomenti:**
- rois : Collection<RegionOfInterest>  
Insieme di RegionOfInterest alle quali appartiene il PointOfInterestImp.

#### 4.4.88 model::navigator::graph::area::PointOfInterestInformation

PointOfInterestInformation
<ul style="list-style-type: none"> <li>- description : String {readOnly}</li> <li>- name : String {readOnly}</li> <li>- category : String {readOnly}</li> </ul>
<ul style="list-style-type: none"> <li>+ PointOfInterestInformation(name : String, description : String, category : String)</li> <li>+ getName() : String</li> <li>+ getDescription() : String</li> <li>+ getCategory() : String</li> </ul>

**Figura 106:** Classe PointOfInterestInformation

**Nome:** PointOfInterestInformation;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare le informazioni associate ad un POI, utili per fornire informazioni ad un utente durante la modalità esplorazione;

**Descrizione:** Classe che rappresenta le informazioni associate ad un POI;

**Attributi:**

- - category : String {readOnly}  
Categoria di appartenenza del POI

- - `description` : `String {readOnly}`  
Descrizione del POI
- - `name` : `String {readOnly}`  
Nome del POI

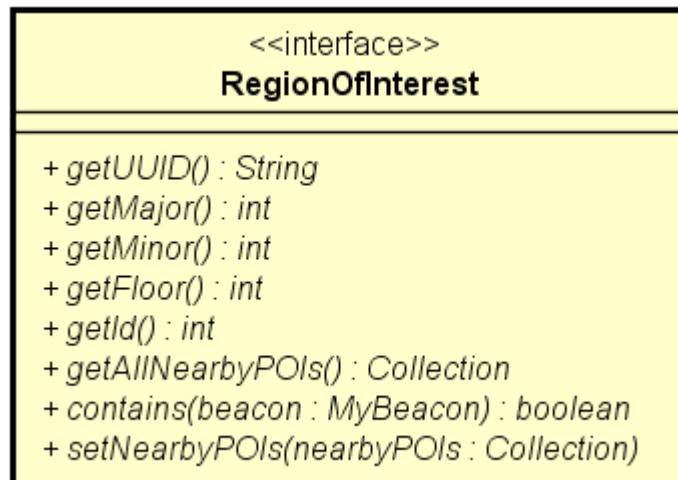
**Metodi:**

- + `getCategory()` : `String`  
Metodo che ritorna la categoria di appartenenza del PointOfInterest a cui l'oggetto è associato
- + `getDescription()` : `String`  
Metodo che ritorna la descrizione del PointOfInterest a cui l'oggetto è associato
- + `getName()` : `String`  
Metodo che ritorna il nome del PointOfInterest a cui l'oggetto è associato
- + `PointOfInterestInformation(name : String, description : String, category : String)`  
Costruttore della classe PointOfInterestInformation

**Argomenti:**

- `name` : `String`  
Nome del POI
- `description` : `String`  
Descrizione del POI
- `category` : `String`  
Categoria di appartenenza del POI

#### 4.4.89 model::navigator::graph::area::RegionOfInterest



**Figura 107:** Interfaccia RegionOfInterest

**Nome:** RegionOfInterest;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione delle informazioni di un ROI dall'utilizzo di quest'ultime;

**Descrizione:** Interfaccia che serve per descrivere un'area coperta dal segnale di un beacon. Espone i metodi per accedere alle informazioni riguardanti a quale beacon è associata tale area, che POI appartengono a tale area e per impostare tali POI;

**Metodi:**

- + *contains(beacon : MyBeacon) : boolean*

Metodo utilizzato per verificare se il beacon passato come parametro è il beacon associato alla RegionOfInterest

**Argomenti:**

- *beacon : MyBeacon*

Beacon di cui si vuole verificare l'associazione con la RegionOfInterest

- + *getAllNearbyPOIs()* : *Collection<PointOfInterest>*  
Metodo che ritorna la collezione di PointOfInterest appartenenti a tale RegionOfInterest
- + *getFloor()* : *int*  
Metodo che ritorna il piano dell'edificio nel quale la ROI rappresentata da tale oggetto si trova
- + *getId()* : *int*  
Metodo che ritorna l'identificativo numerico associato all'oggetto
- + *getMajor()* : *int*  
Metodo che ritorna l'identificativo Major del beacon associato al ROI rappresentato da tale RegionOfInterest
- + *getMinor()* : *int*  
Metodo che ritorna l'identificativo Minor del beacon associato al ROI rappresentato da tale RegionOfInterest
- + *getUUID()* : *String*  
Metodo che ritorna l'identificativo UUID del beacon associato al ROI rappresentato da tale RegionOfInterest
- + *setNearbyPOIs(pois : Collection<PointOfInterest>)* : *void*  
Metodo utilizzato per settare l'insieme di PointOfInterest associato a tale RegionOfInterest

**Argomenti:**

- *pois* : *Collection<PointOfInterest>*  
Insieme di PointOfInterest appartenenti alla RegionOfInterest

#### 4.4.90 model::navigator::graph::area::RegionOfInterestImp

RegionOfInterestImp	
- uuid : String {readOnly}	
- major : int {readOnly}	
- minor : int {readOnly}	
- id : int	
- pois : Collection	
+ RegionOfInterestImp(id : int, UUID : String, major : int, minor : int)	
+ getUUID() : String	
+ getMajor() : int	
+ getMinor() : int	
+ getFloor() : int	
+ getId() : int	
+ contains(beacon : MyBeacon) : boolean	
+ getAllNearbyPOIs() : Collection	
+ setNearbyPOIs(nearbyPOIs : Collection)	

**Figura 108:** Classe RegionOfInterestImp

**Nome:** RegionOfInterestImp;

**Tipo:** Classe;

**Estende:**

- VertexImp.

**Implementa:**

- RegionOfInterest.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la navigazione all'interno di un edificio, permettendo di ottenere indicazioni o informazioni a seconda dell'appartenenza o meno ad una determinata ROI;

**Descrizione:** Classe che rappresenta una ROI, area coperta da un beacon che può contenere uno o più POI. Implementa la classe VertexImp;

**Attributi:**

- - `id` : `int {readOnly}`  
Identificativo numerico di un oggetto RegionOfInterestImp
- - `major` : `int {readOnly}`  
Identificativo Major del beacon associato alla ROI rappresentata dall'oggetto
- - `minor` : `int {readOnly}`  
Identificativo Minor del beacon associato alla ROI rappresentata dall'oggetto
- - `pois` : `Collection<PointOfInterest>`  
Collezione degli oggetti PointOfInterest appartenenti all'oggetto
- - `uuid` : `String {readOnly}`  
Identificativo UUID del beacon associato alla ROI rappresentata dall'oggetto

**Metodi:**

- + `contains(beacon : MyBeacon) : boolean`  
**Override** Metodo utilizzato per verificare se il beacon passato come paramentro è il beacon associato alla RegionOfInterest

**Argomenti:**

- `beacon : MyBeacon`

Beacon di cui si vuole verificare l'associazione con la RegionOfInterest

- + `getAllNearbyPOIs() : Collection<PointOfInterest>`  
**Override** Metodo che ritorna la collezione di PointOfInterest appartenenti a tale RegionOfInterestImp

- + `getFloor() : int`

**Override** Metodo che ritorna il piano dell'edificio nel quale la ROI rappresentata da tale oggetto si trova

- + `getMajor() : int`

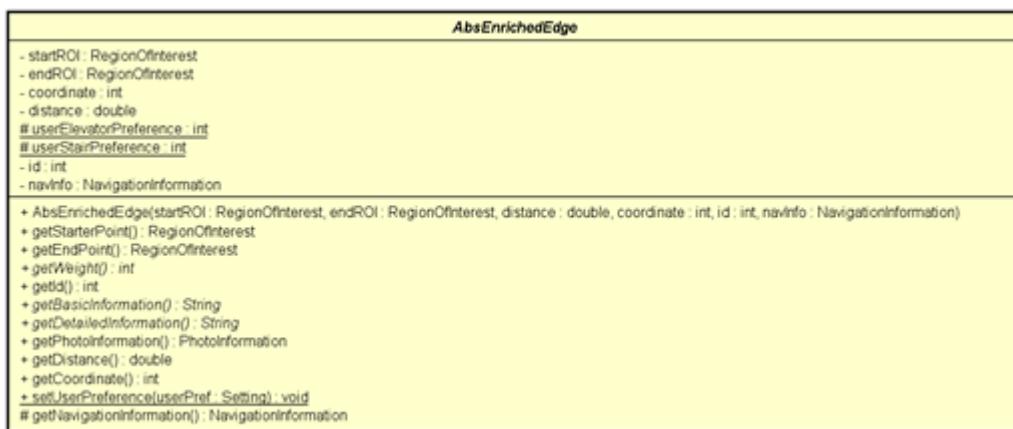
**Override** Metodo che ritorna l'identificativo Major del beacon associato al ROI rappresentato da tale RegionOfInterestImp

- + `getMinor() : int`

**Override** Metodo che ritorna l'identificativo Minor del beacon associato al ROI rappresentato da tale RegionOfInterestImp

- + `getUUID() : String`  
**Override** Metodo che ritorna l'identificativo UUID del beacon associato al ROI rappresentato da tale RegionOfInterestImp
- + `RegionOfInterestImp(id : int, uuid : String, major : int, minor : int)`  
 Costruttore della classe RegionOfInterestImp  
**Argomenti:**
  - `id : int`  
 Identificativo numerico di un RegionOfInterestImp.
  - `uuid : String`  
 Identificativo UUID del beacon associato alla ROI rappresentata dall'oggetto
  - `major : int`  
 Identificativo Major del beacon associato alla ROI rappresentata dall'oggetto
  - `minor : int`  
 Identificativo Minor del beacon associato alla ROI rappresentata dall'oggetto
- + `setNearbyPOIs() : void`  
**Override** Metodo utilizzato per settare l'insieme di PointOfInterest associato a tale RegionOfInterestImp

#### 4.4.91 model::navigator::graph::edge::AbsEnrichedEdge



**Figura 109:** Classe astratta AbsEnrichedEdge

**Nome:** *AbsEnrichedEdge*;

**Tipo:** Classe astratta;

**Implementa:**

- `EnrichedEdge`.

**Visibilità:** `public`;

**Utilizzo:** È utilizzata per costruire il grafo di un edificio insieme a `Vertex`, necessario per la navigazione;

**Descrizione:** Classe astratta che rappresenta l'implementazione di base utilizzata per gli archi del grafo;

**Attributi:**

- - `coordinate` : `int {readOnly}`  
Angolo rispetto al Nord polare tra il punto di inizio dell'arco e il punto finale dell'arco
- - `distance` : `double {readOnly}`  
Lunghezza dell'arco
- - `endROI` : `RegionOfInterest {readOnly}`  
Punto di arrivo dell'arco
- - `id` : `int {readOnly}`  
Identificativo numerico di un arco
- - `navInfo` : `NavigationInformation {readOnly}`  
Informazioni associate ad un arco per il superamento dello stesso
- - `startROI` : `RegionOfInterest {readOnly}`  
Punto di partenza dell'arco
- - `userElevatorPreference` : `int {readOnly}`  
Preferenze dell'utente rispetto agli archi che prevedono un ascensore
- # `userStairPreference` : `int {readOnly}`  
Preferenze dell'utente rispetto agli archi che prevedono delle scale

**Metodi:**

- + `AbsEnrichedEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)`  
Costruttore della classe `AbsEnrichedEdge`

**Argomenti:**

- `startROI` : `RegionOfInterest`  
Punto di partenza dell'arco
  - `endROI` : `RegionOfInterest`  
Punto di arrivo dell'arco
  - `distance` : `double`  
Lunghezza dell'arco
  - `coordinate` : `int`  
Angolo rispetto al Nord polare tra il punto di partenza dell'arco e il punto di arrivo dell'arco
  - `id` : `int`  
Identificativo numerico dell'arco
  - `navInfo` : `NavigationInformation`  
Informazioni associate ad un arco per il superamento dello stesso
- + `getBasicInformation()` : `String`  
**Override** Metodo astratto che ritorna le informazioni di base associate all'arco
  - + `getCoordinate()` : `int`  
**Override** Metodo che ritorna le coordinate di un arco
  - + `getDetailedInformation()` : `String`  
**Override** Metodo astratto che ritorna le informazioni di base associate all'arco
  - + `getDistance()` : `double`  
**Override** Metodo che ritorna la lunghezza di un arco
  - + `getEndPoint()` : `RegionOfInterest`  
**Override** Metodo che ritorna la RegionOfInterest di arrivo dell'arco
  - + `getId()` : `int`  
**Override** Metodo che ritorna l'identificativo numerico associato all'oggetto AbsEnrichedEdge
  - # `getNavigationInformation()` : `NavigationInformation`  
Metodo che ritorna le informazioni di navigazione associate ad un arco
  - + `getPhotoInformation()` : `PhotoInformation`  
**Override** Metodo che ritorna la un oggetto PhotoInformation contente un riferimento alle fotografie associate all'arco

- + `getStarterPoint() : RegionOfInterest`  
**Override** Metodo che ritorna la RegionOfInterest di partenza dell'arco
- + `getWeight() : int`  
**Override** Metodo astratto che ritorna il peso dell'arco
- + `setUserPreference(setting : Setting) : void`  
Metodo che permette di impostare le preferenze di un utente per il calcolo del peso dell'arco

**Argomenti:**

- `setting : Setting`  
Preferenze da impostare

**4.4.92 model::navigator::graph::edge::DefaultEdge**

DefaultEdge
<pre>+ DefaultEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation) + getWeight() : int + getBasicInformation() : String + getDetailedInformation() : String</pre>

**Figura 110:** Classe DefaultEdge**Nome:** DefaultEdge;**Tipo:** Classe;**Estende:**

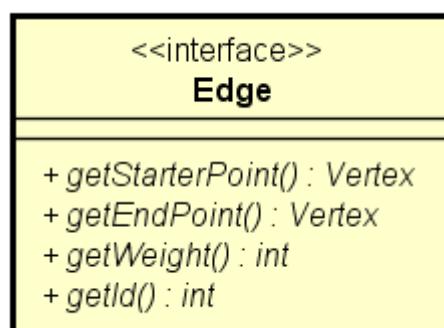
- AbsEnrichedEdge.

**Visibilità:** public;**Utilizzo:** È utilizzata per costruire il grafo di un edificio insieme a Vertex, necessario per la navigazione;**Descrizione:** Classe che implementa AbsEnrichedEdge e rappresenta un arco del grafo senza caratteristiche particolari;**Metodi:**

- + `DefaultEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)`  
Costruttore della classe DefaultEdge

**Argomenti:**

- `startROI` : `RegionOfInterest`  
RegionOfInterest di partenza dell'arco
- `endROI` : `RegionOfInterest`  
RegionOfInterest di arrivo dell'arco
- `distance` : `double`  
Lunghezza dell'arco
- `coordinate` : `int`  
Angolo rispetto al Nord polare presente tra il punto iniziale e il punto finale dell'arco
- `id` : `int`  
Identificativo numerico dell'arco
- `navInfo` : `NavigationInformation`  
Informazioni di navigazione associate all'arco
- + `getBasicInformation()` : `String`  
**Override** Metodo che ritorna le informazioni di base per attraversare l'arco
- + `getDetailedInformation()` : `String`  
**Override** Metodo che ritorna le informazioni di base per attraversare l'arco
- + `getWeight()` : `int`  
**Override** Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

**4.4.93 model::navigator::graph::edge::Edge****Figura 111:** Interfaccia Edge**Nome:** Edge;

**Tipo:** Interfaccia;

**Visibilità:** public;

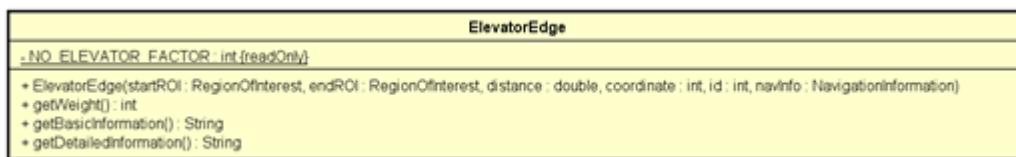
**Utilizzo:** È utilizzata per rendere indipendente l'implementazione di un arco dal suo utilizzo;

**Descrizione:** Interfaccia che serve per descrivere le funzionalità di un arco di un grafo;

**Metodi:**

- + *getEndPoint()* : *Vertex*  
Metodo che restituisce il Vertex di arrivo dell'arco
- + *getId()* : *int*  
Metodo che ritorna l'identificativo numerico dell'arco
- + *getStarterPoint()* : *Vertex*  
Metodo che restituisce il Vertex di partenza dell'arco
- + *getWeight()* : *int*  
Metodo che ritorna il peso dell'arco

#### 4.4.94 model::navigator::graph::edge::ElevatorEdge



**Figura 112:** Classe ElevatorEdge

**Nome:** ElevatorEdge;

**Tipo:** Classe;

**Estende:**

- AbsEnrichedEdge.

**Visibilità:** public;

**Utilizzo:** È utilizzata per costruire il grafo di un edificio insieme a Vertex, necessario per la navigazione;

**Descrizione:** Classe che implementa AbsEnrichedEdge e rappresenta un arco del grafo che corrisponde ad un ascensore;

**Attributi:**

- - NO\_ELEVATOR\_FACTOR : int {readOnly}  
Fattore da aggiungere al peso dell'arco in base alle preferenze di navigazione

**Metodi:**

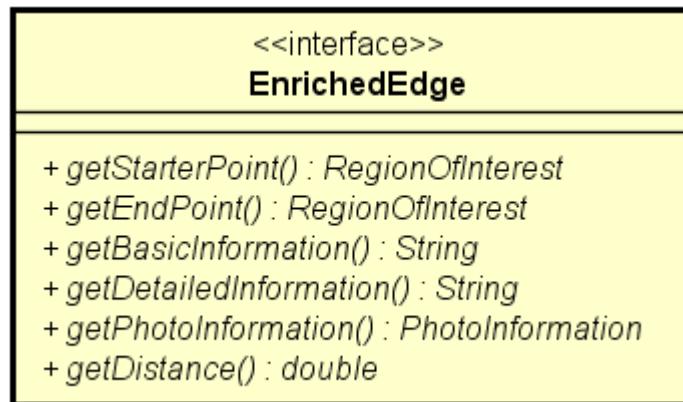
- + **ElevatorEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)**  
Costruttore della classe ElevatorEdge

**Argomenti:**

- **startROI : RegionOfInterest**  
RegionOfInterest di partenza dell'arco
- **endROI : RegionOfInterest**  
RegionOfInterest di arrivo dell'arco
- **distance : double**  
Lunghezza dell'arco
- **coordinate : int**  
Angolo rispetto al Nord polare presente tra il punto iniziale e il punto finale dell'arco
- **id : int**  
Identificativo numerico dell'arco
- **navInfo : NavigationInformation**  
Informazioni di navigazione associate all'arco

- + **getBasicInformation() : String**  
**Override** Metodo che ritorna le informazioni di base per attraversare l'arco
- + **getDetailedInformation() : String**  
**Override** Metodo che ritorna le informazioni di base per attraversare l'arco
- + **getWeight() : int**  
**Override** Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

#### 4.4.95 model::navigator::graph::edge::EnrichedEdge



**Figura 113:** Interfaccia EnrichedEdge

**Nome:** EnrichedEdge;

**Tipo:** Interfaccia;

**Estende:**

- Edge.

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'utilizzo di tale tipo di arco, utile per costruire un insieme di informazioni per far seguire all'utente un determinato percorso, dal modo in cui queste informazioni sono gestite;

**Descrizione:** Interfaccia che serve per rappresentare un arco completo delle informazioni che possono aiutare a superarlo. Questo può rappresentare per esempio un corridoio di un edificio. Quest'interfaccia quindi espone i metodi per accedere a tali informazioni;

**Metodi:**

- + *getBasicInformation()* : String

Metodo astratto che ritorna le informazioni di base associate all'arco

- + *getDetailedInformation()* : *String*  
Metodo astratto che ritorna le informazioni di base associate all'arco
- + *getDistance()* : *double*  
È stata restituita la lunghezza dell'arco
- + *getEndPoint()* : *RegionOfInterest*  
Metodo che ritorna la RegionOfInterest di arrivo dell'arco
- + *getPhotoInformation()* : *PhotoInformation*  
Metodo che ritorna un oggetto PhotoInformation contenente un riferimento alle fotografie associate all'arco
- + *getStarterPoint()* : *RegionOfInterest*  
Metodo che ritorna la RegionOfInterest di partenza dell'arco

#### 4.4.96 model::navigator::graph::edge::StairEdge

StairEdge
- NO_STAIR_FACTOR int {readOnly}
+ StairEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)
+ getWeight() : int
+ getBasicInformation() : String
+ getDetailedInformation() : String

**Figura 114:** Classe StairEdge

**Nome:** StairEdge;

**Tipo:** Classe;

**Estende:**

- AbsEnrichedEdge.

**Visibilità:** public;

**Utilizzo:** È utilizzata per costruire il grafo di un edificio insieme a Vertex, necessario per la navigazione;

**Descrizione:** Classe che implementa AbsEnrichedEdge e rappresenta un arco del grafo corrispondente ad una rampa di scale;

**Attributi:**

- - NO\_STAIR\_FACTOR : int {readOnly}

Fattore da aggiungere al peso dell'arco in base alle preferenze di navigazione

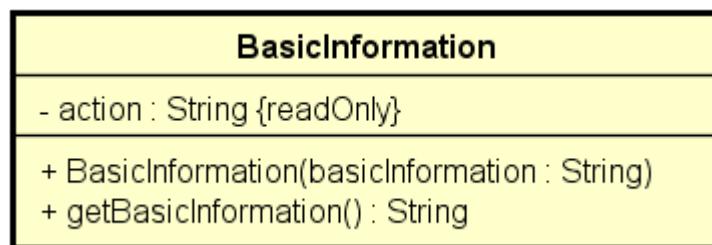
#### Metodi:

- + **getBasicInformation()** : String  
**Override** Metodo che ritorna le informazioni dettagliate per attraversare l'arco
- + **getDetailedInformation()** : String  
**Override** Metodo che ritorna le informazioni dettagliate per attraversare l'arco
- + **getWeight()** : int  
**Override** Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente
- + **StairEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)**  
Costruttore della classe StairEdge

#### Argomenti:

- **startROI : RegionOfInterest**  
RegionOfInterest di partenza dell'arco
- **endROI : RegionOfInterest**  
RegionOfInterest di arrivo dell'arco
- **distance : double**  
Lunghezza dell'arco
- **coordinate : int**  
Angolo rispetto al Nord polare presente tra il punto iniziale e il punto finale dell'arco
- **id : int**  
Identificativo numerico dell'arco
- **navInfo : NavigationInformation**  
Informazioni di navigazione associate all'arco

#### 4.4.97 model::navigator::graph::navigationinformation::BasicInformation



**Figura 115:** Classe BasicInformation

**Nome:** BasicInformation;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per restituire le informazioni di base necessarie per la navigazione sotto forma di testo;

**Descrizione:** Classe contenente le informazioni di base per la navigazione;

**Attributi:**

- - action : String {readOnly}  
Azione da compiere per superare un determinato arco

**Metodi:**

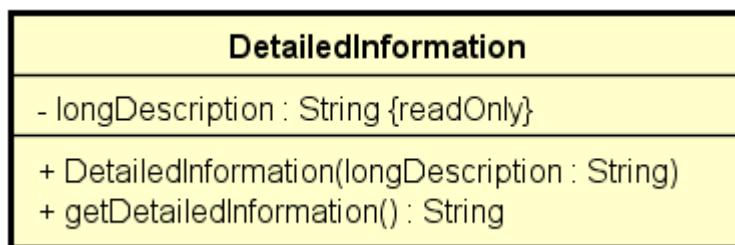
- + BasicInformation(basicInformation : String)  
Costruttore della classe BasicInformation

**Argomenti:**

- basicInformation : String  
Informazioni di base per il superamento di un arco

- + getBasicInformation() : String  
Metodo che ritorna le informazioni contenute in un oggetto BasicInformation per il superamento di un determinato arco

#### 4.4.98 model::navigator::graph::navigationinformation::DetailedInformation



**Figura 116:** Classe DetailedInformation

**Nome:** DetailedInformation;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per restituire le informazioni dettagliate sotto forma di testo, utilizzate durante la navigazione;

**Descrizione:** Classe contenente le informazioni dettagliate utilizzate per la navigazione;

**Attributi:**

- - longDescription : String {readOnly}  
Descrizione dettagliata delle azioni da compiere per il superamento di un certo arco

**Metodi:**

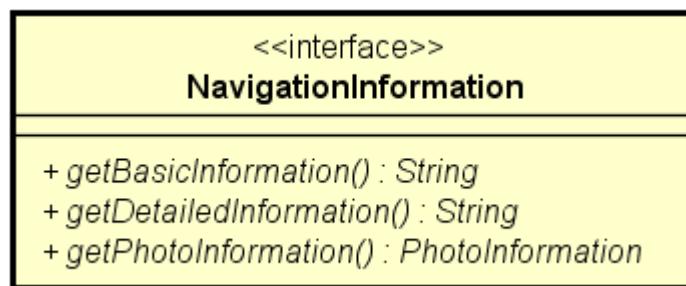
- + DetailedInformation(longDescription : String)  
Costruttore della classe DetailedInformation

**Argomenti:**

- longDescription : String  
Descrizione dettagliata delle azioni da compiere per il superamento di un certo arco

- + getDetailedInformation()  
Metodo che ritorna le informazioni contenute in un oggetto DetailedInformation per il superamento di un determinato arco

#### 4.4.99 model::navigator::graph::navigationinformation::NavigationInformation



**Figura 117:** Interfaccia NavigationInformation

**Nome:** NavigationInformation;

**Tipo:** Interfaccia;

**Visibilità:** public;

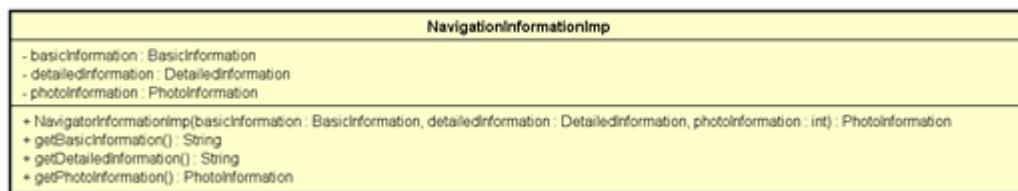
**Utilizzo:** È utilizzata per rendere indipendente l'implementazione delle informazioni di navigazione dall'utilizzo di quest'ultime;

**Descrizione:** Interfaccia che espone i metodi per accedere alle informazioni di navigazione;

**Metodi:**

- **+ *getBasicInformation()* : String**  
Metodo che ritorna le informazioni di base per il superamento dell'arco al quale tale oggetto è associato
- **+ *getDetailedInformation()* : String**  
Metodo che ritorna delle informazioni dettagliate per il superamento dell'arco al quale tale oggetto è associato
- **+ *getPhotoInformation()* : PhotoInformation**  
Metodo che ritorna un oggetto PhotoInformation contenente i riferimenti alle fotografie riguardanti l'arco al quale tale oggetto è associato

#### 4.4.100 model::navigator::graph::navigationinformation:: NavigationInformationImp



**Figura 118:** Classe NavigationInformationImp

**Nome:** NavigationInformationImp;

**Tipo:** Classe;

**Implementa:**

- NavigationInformation.

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare le informazioni testuali, testuali estese e visive per permettere all'utente di navigare tramite dei campi dati di tipo BasicInformation, DetailedInformation e PhotoInformation;

**Descrizione:** Classe utilizzata per recuperare le informazioni da fornire all'utente per la navigazione;

**Attributi:**

- - basicInformation : BasicInformation {readOnly}  
Informazioni di base associate ad un EnrichedEdge
- - detailedInformation : DetailedInformation {readOnly}  
Informazioni di dettagliate associate ad un EnrichedEdge
- - photoInformation : PhotoInformation {readOnly}  
Fotografie associate ad un EnrichedEdge

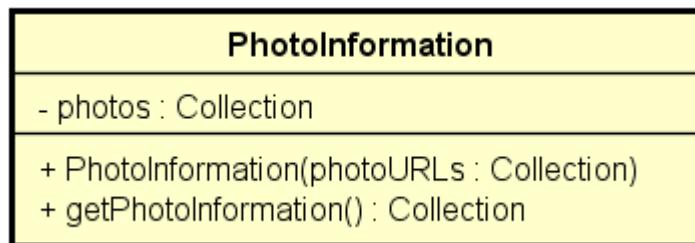
**Metodi:**

- + getBasicInformation() : String  
**Override** Metodo che ritorna le informazioni di base per il superamento dell'arco al quale tale oggetto è associato

- + `getDetailedInformation() : String`  
**Override** Metodo che ritorna delle informazioni dettagliate per il superamento dell'arco al quale tale oggetto è associato
- + `getPhotoInformation() : PhotoInformation`  
**Override** Metodo che ritorna un oggetto PhotoInformation contenente i riferimenti alle fotografie riguardanti l'arco al quale tale oggetto è associato
- + `NavigationInformationImp(basicInformation : BasicInformation, detailedInformation : DetailedInformation, photoInformation : PhotoInformation)`  
Costruttore della classe NavigationInformationImp

**Argomenti:**

- `basicInformation : BasicInformation`  
Informazioni di base associate ad un EnrichedEdge
- `detailedInformation : DetailedInformation`  
Informazioni di dettagliate associate ad un EnrichedEdge
- `photoInformation : PhotoInformation`  
Fotografie associate ad un EnrichedEdge

**4.4.101 model::navigator::graph::navigationinformation::PhotoInformation****Figura 119:** Classe PhotoInformation

**Nome:** PhotoInformation;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per restituire le informazioni visive utilizzate durante la navigazione;

**Descrizione:** Classe che contiene le informazioni visive (sotto forma di URI a foto) utilizzate per la navigazione;

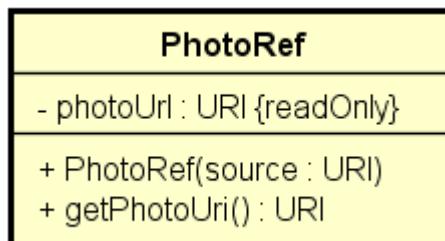
**Attributi:**

- - photos : Collection<PhotoRef> {readOnly}  
Collezione di oggetti PhotoRef rappresentanti le fotografie dell'arco a cui l'oggetto è associato

**Metodi:**

- + getPhotoInformation() : Collection<PhotoRef>  
Metodo che restituisce una collezione di oggetti PhotoRef rappresentanti le fotografie dell'arco a cui l'oggetto è associato
- + PhotoInformation()  
Costruttore della classe PhotoInformation

#### 4.4.102 model::navigator::graph::navigationinformation::PhotoRef



**Figura 120:** Classe PhotoRef

**Nome:** PhotoRef;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per fornire l'URI di una foto, necessario per il recupero della foto durante la navigazione;

**Descrizione:** Classe che rappresenta l'URI di una foto;

**Attributi:**

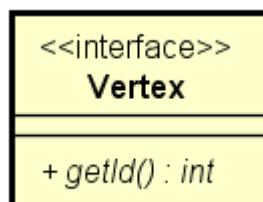
- - `photoUrl : URI {readOnly}`  
URL di una fotografia

**Metodi:**

- + `getPhotoUri() : URI`  
Metodo che restituisce l'URL per accedere alla fotografia che l'oggetto rappresenta
- + `PhotoRef(source : URI)`  
Costruttore della classe PhotoRef

**Argomenti:**

- `source : URI`  
URL di una fotografia

**4.4.103 model::navigator::graph::vertex::Vertex****Figura 121:** Interfaccia Vertex**Nome:** Vertex;**Tipo:** Interfaccia;**Visibilità:** public;**Utilizzo:** È utilizzata per rendere indipendente l'implementazione di un vertice dal suo utilizzo;**Descrizione:** Interfaccia che serve per descrivere le funzionalità di un vertice di un grafo;**Metodi:**

- + `getId() : int`  
Metodo che ritorna l'identificativo numerico associato al vertice

#### 4.4.104 model::navigator::graph::vertex::VertexImp



**Figura 122:** Classe VertexImp

**Nome:** VertexImp;

**Tipo:** Classe;

**Implementa:**

- Vertex.

**Visibilità:** public;

**Utilizzo:** È utilizzata per costruire il grafo di un edificio, necessario per ricavare il percorso da seguire durante la navigazione;

**Descrizione:** Classe rappresentante il vertice (o nodo) di un grafo;

**Attributi:**

- - id : int {readOnly}
- Identificativo numerico di un VertexImp

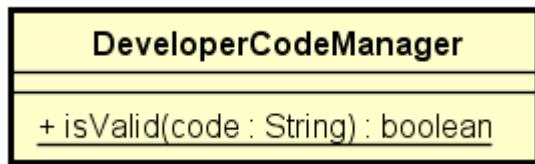
**Metodi:**

- + getId() : int  
**Override** Metodo che ritorna l'identificativo numerico associato all'oggetto VertexImp
- + VertexImp(id : int )  
Costruttore della classe VertexImp

**Argomenti:**

- id : int  
Identificativo numerico di un oggetto VertexImp

#### 4.4.105 model::usersetting::DeveloperCodeManager



**Figura 123:** Classe DeveloperCodeManager

**Nome:** DeveloperCodeManager;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per verificare che un codice sviluppatore sia valido o meno;

**Descrizione:** Classe che per la verifica dei codici sviluppatore;

**Metodi:**

- + isValid(code : String) : boolean

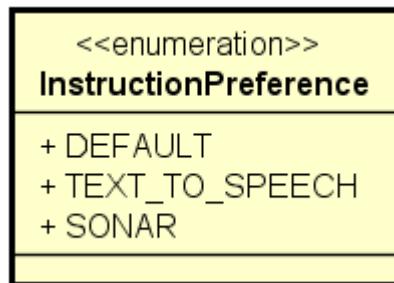
Questo metodo permette di verificare se il codice inserito è valido per attivare la modalità sviluppatore

**Argomenti:**

- code : String

Questo parametro richiede il codice per attivare la modalità sviluppatore

#### 4.4.106 model::usersetting::InstructionPreference



**Figura 124:** Classe InstructionPreference

**Nome:** InstructionPreference;

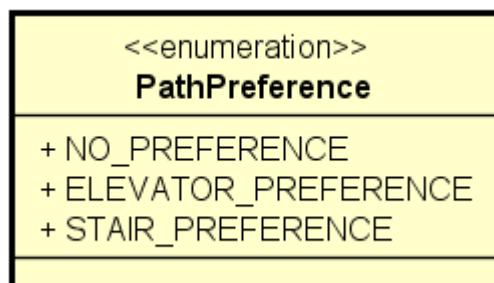
**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per elencare tutte le possibili scelte che possono essere fatte riguardanti la fruizione delle informazioni in fase di navigazione e i metodi per la conversione di tali valori da e verso int;

**Descrizione:** Classe enumeratore che espone le possibili preferenze riguardanti la fruizione delle informazioni;

#### 4.4.107 model::usersetting::PathPreference



**Figura 125:** Classe PathPreference

**Nome:** PathPreference;

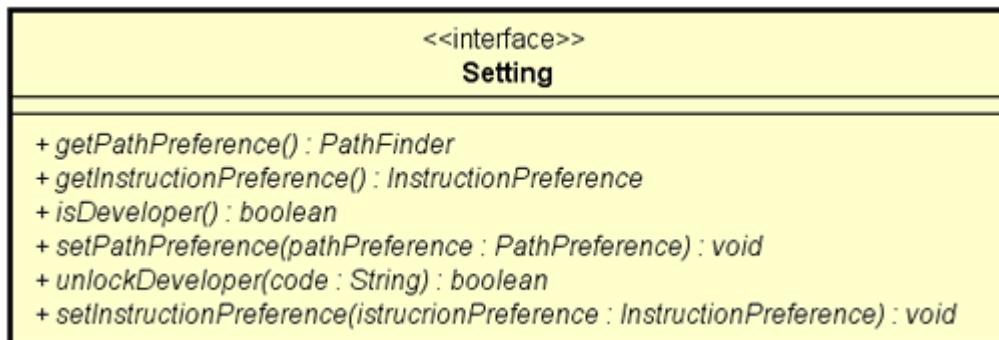
**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È utilizzata per elencare tutte le possibili scelte che possono essere fatte riguardanti il percorso preferito in fase di navigazione e i metodi per la conversione di tali valori da e verso int;

**Descrizione:** Classe enumeratore che espone le possibili preferenze riguardanti il percorso di navigazione;

#### 4.4.108 model::usersetting::Setting



**Figura 126:** Interfaccia Setting

**Nome:** Setting;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'accesso e la gestione delle impostazioni di un utente dall'implementazione dei metodi e delle classi che si occupano di tale gestione;

**Descrizione:** Interfaccia che espone i metodi per accedere alle preferenze di un utente riguardo il percorso di navigazione e le istruzioni di navigazione. Espone inoltre i metodi per verificare se è stato inserito un codice sviluppatore valido e per verificare i codici inseriti;

**Metodi:**

- + *getInstructionPreference()* : *InstructionPreference*  
Metodo che ritorna le preferenze riguardanti la modalità di fruizione delle informazioni
- + *getPathPreference()* : *PathPreference*  
Metodo che ritorna le preferenze di percorso
- + *isDeveloper()* : *boolean*  
Metodo che verifica se è stato inserito in precedenza un codice sviluppatore valido
- + *setInstructionPreference(instructionPreference : InstructionPreference) : void*  
Metodo che permette di modificare le impostazioni riguardanti le preferenze di fruizione delle istruzioni di navigazione

**Argomenti:**

- *instructionPreference* : *InstructionPreference*  
Preferenza riguardante la fruizione delle istruzioni di navigazione.
- + *setPathPreference(pathPreference : PathPreference) : void*  
Metodo che permette di modificare le impostazioni riguardanti le preferenze sul percorso di navigazione

**Argomenti:**

- *pathPreference* : *PathPreference*  
Preferenza riguardante il percorso di navigazione.
- + *unlockDeveloper() : boolean*  
Metodo che passato una stringa in ingresso controlla se tale stringa rappresenta un codice sviluppatore valido

#### 4.4.109 model::usersetting::SettingImp

SettingImp
- pathPreference : PathPreference - instructionPreference : InstructionPreference
+ getPathPreference() : PathPreference + getInstructionPreference() : InstructionPreference + isDeveloper() : boolean + unlockDeveloper(code : String) : boolean + setPathPreference(pathPreference : PathPreference) : void + setInstructionPreference(instructionPreference : InstructionPreference) : void

**Figura 127:** Classe SettingImp

**Nome:** SettingImp;

**Tipo:** Classe;

**Implementa:**

- Setting.

**Componenti delle librerie utilizzate:**

- android.content.SharedPreferences (Android);
- android.content.SharedPreferences.Editor (Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per accedere alle impostazioni relative a preferenze e alle opzioni di sviluppatore salvate sul dispositivo utilizzando la classe android.content.SharedPreferences. Ha inoltre il compito di modificare tali preferenze e impostazioni utilizzando le classi android.content.SharedPreferences e android.content.SharedPreferences.Editor;

**Descrizione:** Classe che implementa l'interfaccia Setting. Implementa tutti i metodi per la gestione delle impostazioni dell'utente;

**Attributi:**

- - `instructionPreference : InstructionPreference`  
Preferenze dell'utente riguardanti le modalità di fruizione delle informazioni per la navigazione
- - `pathPreference : PathPreference`  
Preferenze di percorso dell'utente

**Metodi:**

- + `getInstructionPreference() : InstructionPreference`  
**Override** Metodo che ritorna le preferenze riguardanti la modalità di fruizione delle informazioni
- + `getPathPreference() : PathPreference`  
**Override** Metodo che ritorna le preferenze di percorso
- + `isDeveloper() : boolean`  
**Override** Metodo che verifica se è stato inserito in precedenza un codice sviluppatore valido
- + `setInstructionPreference(instructionPreference : InstructionPreference) : void`  
**Override** Metodo che permette di modificare le impostazioni riguardanti le preferenze di fruizione delle istruzioni di navigazione

**Argomenti:**

- `instructionPreference : InstructionPreference`  
Questo parametro richiede il tipo di istruzioni che si vuole ricevere durante la navigazione
- + `setPathPreference(pathPreference : PathPreference) : void`  
**Override** Metodo che permette di modificare le impostazioni riguardanti le preferenze sul percorso di navigazione

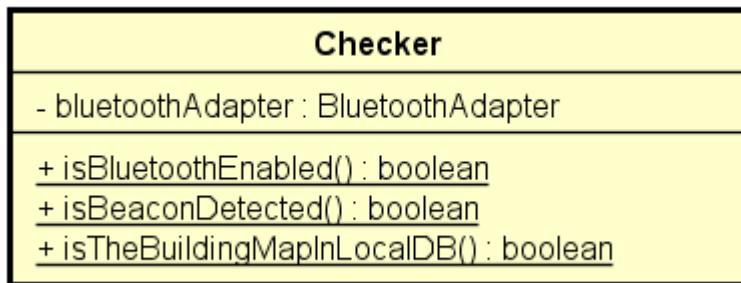
**Argomenti:**

- `pathPreference : PathPreference`  
Questo parametro richiede le preferenze di percorso che un utente vuole impostare
- + `unlockDeveloper(code : String) : boolean`  
**Override** Metodo che passato una stringa in ingresso controlla se tale stringa rappresenta un codice sviluppatore valido

**Argomenti:**

- `code : String`  
Questo parametro richiede il codice per attivare la modalità sviluppatore

#### 4.4.110 presenter::Checker



**Figura 128:** Classe Checker

**Nome:** Checker;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.bluetooth.BluetoothAdapter(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per effettuare controlli sullo stato del dispositivo e dell'applicativo;

**Descrizione:** Classe che utilizza BluetoothAdapter e permette di effettuare controlli sullo stato del dispositivo e dell'applicativo;

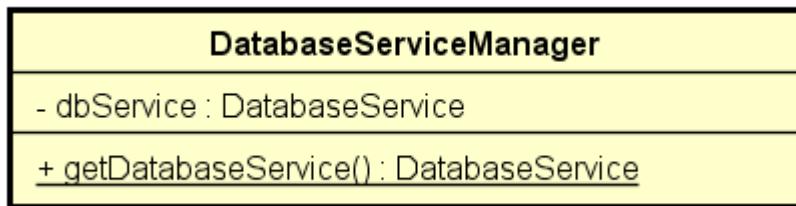
**Attributi:**

- - bluetoothAdapter : BluetoothAdapter  
Utilizzato per accedere al Bluetooth del dispositivo

**Metodi:**

- + isBeaconDetected() : boolean  
Metodo che controlla se sono stati rilevati dei beacon
- + isBluetoothEnabled() : boolean  
Metodo che controlla se è stato attivato il Bluetooth del dispositivo
- + isTheBuildingMapInLocalDB() : boolean  
Metodo che controlla se una mappa è presente sul database locale

#### 4.4.111 presenter::DatabaseServiceManager



**Figura 129:** Classe DatabaseServiceManager

**Nome:** DatabaseServiceManager;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.app.Application(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per fornire un riferimento al DatabaseService del model;

**Descrizione:** Classe che estende Application e recupera il riferimento al DatabaseService;

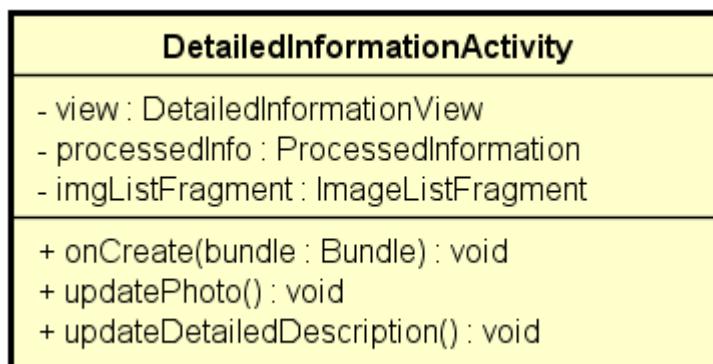
**Attributi:**

- - dbService : DatabaseService  
Riferimento per accedere al database locale

**Metodi:**

- + getDatabaseService() : DatabaseService  
Metodo che restituisce un riferimento al database locale

#### 4.4.112 presenter::DetailedInformationActivity



**Figura 130:** Classe DetailedInformationActivity

**Nome:** DetailedInformationActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare le informazioni dettagliate riguardo ad una certa istruzione di navigazione. Gestisce anche tutte le richieste che vengono fatte dalla DetailedInformationView;

**Descrizione:** Classe che estende AppCompatActivity per la gestione delle informazioni dettagliate riguardo alla navigazione;

**Attributi:**

- - imgListFragment : ImageListFragment  
Contiene la lista di tutte le anteprime delle immagini associate ad un certo Edge
- - processedInfo : ProcessedInformation  
ProcessedInformation associata all'Edge corrente
- - view : DetailedInformationView  
View associata a tale Activity

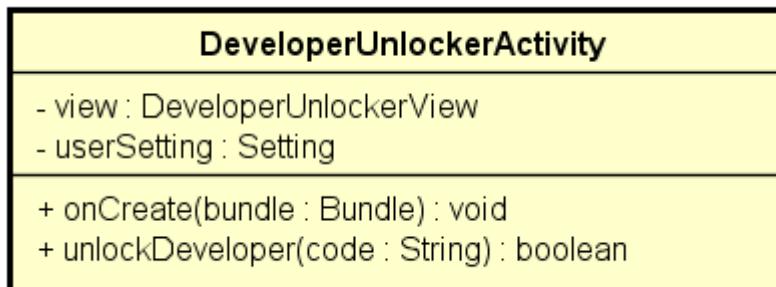
**Metodi:**

- + `onCreate(bundle : Bundle) : void`  
Override Metodo che inizializza la DetailedInformationView

**Argomenti:**

- `bundle : Bundle`  
Componente per salvare lo stato dell'applicazione

- + `updateDetailedDescription() : void`  
Metodo che aggiorna le informazioni testuali estese visualizzate sulla View
- + `updatePhoto() : void`  
Metodo che aggiorna la foto visualizzata sulla View

**4.4.113 presenter::DeveloperUnlockerActivity**

**Figura 131:** Classe DeveloperUnlockerActivity

**Nome:** DeveloperUnlockerActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.support.v7.app.AppCompatActivity(Android)`.

**Visibilità:** public;

**Utilizzo:** È usata per verificare se il codice inserito dall'utente è corretto.  
Gestisce tutte le possibili richieste di DeveloperUnlockerView ;

**Descrizione:** È una classe che estende AppCompatActivity che consente di gestire l'interazione tra DeveloperUnlockerView ed il model;

**Attributi:**

- - userSetting : Setting  
Impostazioni dell'utente
- - view : DeveloperUnlockerView  
View associata a tale Activity

**Metodi:**

- + onCreate(bundle : Bundle) : void  
**Override** Metodo che inizializza la View associata e recupera un riferimento alle impostazioni dell'utente

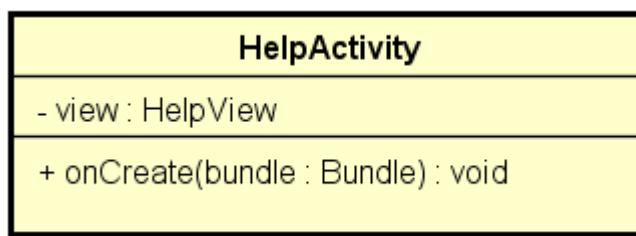
**Argomenti:**

- bundle : Bundle  
Componente per salvare lo stato dell'applicazione

- + unlockDeveloper(code : String) : boolean  
Metodo che permette di attivare le funzionalità sviluppatore

**Argomenti:**

- code : String  
Codice di sblocco delle attività sviluppatore

**4.4.114 presenter::HelpActivity**

**Figura 132:** Classe HelpActivity

**Nome:** HelpActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È usata per recuperare la guida dell'applicativo dal model e metterla a disposizione di HelpView. Gestisce tutte le possibili richieste effettuare da HelpView;

**Descrizione:** È una classe che estende AppCompatActivity che gestisce consente di gestire l'interazione tra HelpView ed il model;

**Attributi:**

- - view : HelpView  
View associata a tale Activity

**Metodi:**

- + onCreate(bundle : Bundle) : void  
**Override** Metodo che inizializza la HelpView

**Argomenti:**

- bundle : Bundle  
Componente per salvare lo stato dell'applicazione

#### 4.4.115 presenter::HomeActivity

HomeActivity	
- view : HomeView	
- informationManager : InformationManager	
+ onCreate(bundle : Bundle) : void	
+ updateBuildingName() : void	
+ updateBuildingDescription() : void	
+ updateBuildingOpeningHours() : void	
+ updatePoiCategoryList() : void	
+ updateBuildingAddress() : void	
+ enableSuggestions() : void	
+ showPoisCategory(categoryName : String) : void	
+ showExplorer() : void	
+ showLocalMaps() : void	
+ showPreferences() : void	
+ showHelp() : void	
+ startNavigation(poiPosition : int) : void	

**Figura 133:** Classe HomeActivity

**Nome:** HomeActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare tutte le informazioni necessarie dal model al fine di popolare la HomeView. Gestisce anche tutte le richieste che vengono fatte dalla HomeView;

**Descrizione:** Classe che estende AppCompatActivity per la gestione dell’interazione tra HomeView ed il model;

**Attributi:**

- - **informationManager** : `InformationManager`  
Riferimento utilizzato per accedere alle informazioni trattate dal model
- - **view** : `HomeView`  
View associata a tale Activity

**Metodi:**

- + `enableSuggestions()` : `void`  
Metodo che permette di attivare la lista dei possibili POI raggiungibili a partire da una stringa
- + `onCreate(bundle : Bundle)` : `void`  
**Override** Metodo che inizializza la View associata e recupera un riferimento all'InformationManager

**Argomenti:**

- `bundle` : `Bundle`  
Componente per salvare lo stato dell'applicazione

- + `showExplorer()` : `void`  
Metodo che viene invocato a seguito della richiesta di visualizzazione della modalità esplora
- + `showHelp()` : `void`  
Metodo che viene invocato a seguito della richiesta di visualizzazione della guida
- + `showLocalMaps()` : `void`  
Metodo che viene invocato a seguito della richiesta di visualizzazione delle mappe salvate nel database locale
- + `showPoisCategory(categoryName : String)` : `void`  
Metodo che viene invocato a seguito della richiesta di visualizzazione di tutti i POI appartenenti ad un certa categoria

**Argomenti:**

- `categoryName` : `String`  
Nome della categoria di cui visualizzare l'insieme di POI appartenente

- + `showPreferences()` : `void`  
Metodo che viene invocato a seguito della richiesta di visualizzazione delle preferenze dell'utente
- + `startNavigation(poiPosition : int)` : `void`  
Metodo che viene invocato a seguito della richiesta di inizio della navigazione

**Argomenti:**

- poiPosition : int  
Identificativo del POI verso il quale si vuole effettuare una navigazione
- + updateBuildingAddress() : void  
Metodo che recupera l'indirizzo dell'edificio e lo passa alla View corrispondente
- + updateBuildingDescription() : void  
Metodo che recupera la descrizione dell'edificio e lo passa alla View corrispondente
- + updateBuildingName() : void  
Metodo che recupera il nome dell'indirizzo dell'edificio e lo passa alla View corrispondente
- + updateBuildingOpeningHours() : void  
Metodo che recupera l'orario di apertura dell'edificio e lo passa alla View corrispondente
- + updatePoiCategoryList() : void  
Metodo che recupera la lista di categorie di POI nell'edificio e lo passa alla View corrispondente

**4.4.116 presenter::ImageAdapter**

ImageAdapter	
- context : Context	
+ ImageAdapter()	
+ getCount() : int	
+ getItem(position : int) : Object	
+ getItemId(position : int) : long	
+ getView(position : int, convertView : View, parent : ViewGroup) : View	

**Figura 134:** Classe ImageAdapter**Nome:** ImageAdapter;**Tipo:** Classe;**Componenti delle librerie utilizzate:**

- android.widget.BaseAdapter(Android).

**Visibilità:** public;

**Utilizzo:** È usata per gestire la lista di immagini associate ad una certa istruzione di navigazione;

**Descrizione:** Classe che estende FragmentStatePagerAdapter per gestire la lista di immagini associate ad una certa istruzione di navigazione;

**Attributi:**

- - context : Context  
Contesto dell'applicativo

**Metodi:**

- + getCount() : int  
**Override** Metodo che viene utilizzato per ottenere il numero di immagini relative a quel POI
- + getItem(position : int) : Object  
**Override** Metodo che ritorna l'URL di una foto in una certa posizione

**Argomenti:**

- position : int  
Posizione della foto di cui si vuole recuperare l'URL

- + getItemId(position : int) : long  
**Override** Metodo che ritorna l'identificativo numerico di una foto in una certa posizione

**Argomenti:**

- position : int  
Posizione della foto di cui si vuole recuperare l'identificativo numerico

- + getView(position : int, convertView : View, parent : ViewGroup) : View  
**Override** Metodo che restituisce la foto in una certa posizione

**Argomenti:**

- position : int  
Posizione della foto che si vuole recuperare
- convertView : View  
Il layout dell'anteprima della foto restituita

- parent : ViewGroup  
Il layout della lista di anteprime
- + ImageAdapter()  
Costruttore della classe ImageAdapter

#### 4.4.117 presenter::ImageDetailActivity

ImageDetailActivity	
- view : ImageDetailView	
- listPhotos : List	
- startItem : int	
- imgPgAdpt : ImagePageAdapter	
+ onCreate(bundle : Bundle) : void	

**Figura 135:** Classe ImageDetailActivity

**Nome:** ImageDetailActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare l'insieme di immagini associate ad una certa istruzione di navigazione ed esporle alla view che le utilizza. Gestisce anche tutte le richieste che vengono fatte dalla ImageDetailView;

**Descrizione:** Classe che implementa AppCompatActivity per la gestione dell'interazione tra ImageDetailView ed il model;

**Attributi:**

- - imgPgAdpt : ImagePageAdapter  
ImagePageAdapter per la gestione dello slideshow delle immagini relative ad una certa istruzione di navigazione

- - `listPhotos : List<String>`  
Lista contenenti gli URI delle foto
- - `startItem : int`  
Elemento di partenza sul quale l'utente ha cliccato
- - `view : ImageDetailView`  
View associata a tale Activity

**Metodi:**

- + `onCreate(bundle : Bundle) : void`  
**Override** Metodo che inizializza ImageDetailView

**Argomenti:**

- `bundle : Bundle`  
Componente per salvare lo stato dell'applicazione

#### 4.4.118 presenter::ImageListFragment

<b>ImageListFragment</b>	
-	<code>imgAdapter : ImageAdapter</code>
-	<code>photosUrls : List</code>
+	<code>ImageListFragment()</code>
+	<code>newInstance(photosUrls : List) : ImageListFragment</code>
+	<code>onCreate(bundle : Bundle) : void</code>
+	<code>onCreateView(inflater : LayoutInflater, viewGroup : ViewGroup, bundle : Bundle) : View</code>
+	<code>onItemClick() : void</code>

**Figura 136:** Classe ImageListFragment

**Nome:** ImageListFragment;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.app.Fragment(Android).`

**Visibilità:** public;

**Utilizzo:** È utilizzata per la gestione dell'interazione tra lista di immagini associate ad una istruzione e le possibili azioni che è possibile effettuare su di esse;

**Descrizione:** Classe che estende BaseAdapter per la gestione della lista di immagini relative ad una istruzione di navigazione;

**Attributi:**

- - `imgAdapter : ImageAdapter`  
Collegamento tra la lista di URL relative ad un certo POI e la View che mostra quelle anteprime
- - `photosUrls : List<String>`  
Gli URL delle foto

**Metodi:**

- + `ImageListFragment()`  
Costruttore della classe ImageListFragment
- + `newInstance(photosUrls : List<String>) : ImageListFragment`  
Metodo che ritorna una nuova istanza di un ImageListFragment a partire da una lista di foto

**Argomenti:**

- `photosUrls : List<String>`  
Lista di URL delle foto
- + `onCreate(bundle : Bundle) : void`  
**Override** Metodo che inizializza ImageView

**Argomenti:**

- `bundle : Bundle`  
Componente per salvare lo stato dell'applicazione
- + `onCreateView(lay : LayoutInflater, viewGr : ViewGroup, bundle : Bundle) : View`  
**Override** Metodo che crea il layout di un fragment

**Argomenti:**

- `lay : LayoutInflater`  
Oggetto rappresentante il file XML della View e farne linflate
- `viewGr : ViewGroup`  
Layout della lista di anteprime
- `bundle : Bundle`  
Componente per salvare lo stato dell'applicazione
- + `onItemClick() : void`  
**Override** Metodo che viene invocato alla pressione di un bottone

#### 4.4.119 presenter::ImagePagerAdapter

ImagePagerAdapter	
-	size : int
+	ImagePagerAdapter(fm : FragmentManager, size : int)
+	getCount() : int
+	getItem(position : int) : Fragment

**Figura 137:** Classe ImagePagerAdapter

**Nome:** ImagePagerAdapter;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v4.app.FragmentStatePagerAdapter(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per la gestione dello slideshow delle immagini relative ad una certa indicazione di navigazione;

**Descrizione:** Classe che estende FragmentStatePagerAdapter per la gestione dello slideshow delle immagini relative ad una certa istruzione di navigazione;

**Attributi:**

- - size : int  
Numero di elementi della slideshow

**Metodi:**

- + getCount() : int  
Metodo che fornisce il numero di elementi all'interno dello slideshow
- + getItem(position : int) : Fragment  
Metodo che permette di ottenere un'immagine all'intero dello slideshow

**Argomenti:**

– position : int

Intero che indica la posizione dell'immagine da recuperare all'interno dello slideshow

- + `ImagePagerAdapter(fm : FragmentManager, size : int)`  
Costruttore della classe ImagePagerAdapter

**Argomenti:**

– fm : FragmentManager

Interfaccia che serve per l'interazione con un Fragment

– size : int

Il numero di immagini passate

#### 4.4.120 presenter::InformationManagerPresenter

InformationManagerPresenter
- informationManager : InformationManager
+ getInformationManager() : InformationManager

**Figura 138:** Classe InformationManagerPresenter

**Nome:** InformationManagerPresenter;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.app.Application(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per restituire un riferimento al NavigationManager del model;

**Descrizione:** Classe che estende Application e recupera il riferimento a InformationManager del model;

**Attributi:**

- - `informationManager : InformationManager`  
Riferimento utilizzato per accedere alle informazioni trattate dal model

**Metodi:**

- + `getInformationManager() : InformationManager`  
**Override** Metodo che restituisce un InformationManger per accedere alle informazioni del presenter

#### 4.4.121 presenter::LocalMapActivity

<b>LocalMapActivity</b>	
-	<code>view : LocalMapManagerView</code>
+	<code>onCreate(bundle : Bundle) : void</code>
+	<code>updateMap(mapPosition : int) : void</code>
+	<code>deleteMap(mapPosition : int) : void</code>

**Figura 139:** Classe LocalMapActivity

**Nome:** LocalMapActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.support.v7.app.AppCompatActivity(Android)`.

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare le mappe salvate in locale dal model ed esporle a LocalMapView. Gestisce tutte le possibili richieste fatte da LocalMapView;

**Descrizione:** Classe che estende AppCompatActivity e per la gestione dell'interazione tra LocalMapView ed il model;

**Attributi:**

- - `view : LocalMapManagerView`  
View associata a tale Activity

**Metodi:**

- + deleteMap(mapPosition : int) : void

Metodo che permette di rimuovere una mappa del database locale

**Argomenti:**

- mapPosition : int

Posizione occupata dalla mappa da rimuovere

- + onCreate(bundle : Bundle) : void

**Override** Metodo che inizializza la View associata a tale Activity

**Argomenti:**

- bundle : Bundle

Componente per salvare lo stato dell'applicazione

- + updateMap(mapPosition : int) : void

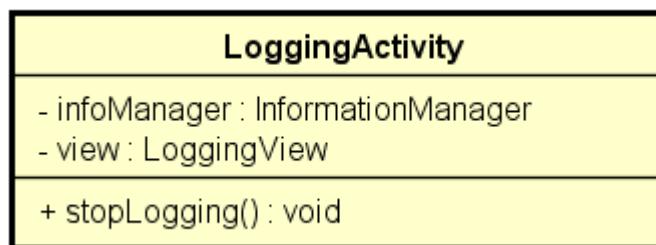
Metodo che permette di aggiornare una mappa del database locale

**Argomenti:**

- mapPosition : int

Posizione della mappa da aggiornare

#### 4.4.122 presenter::LoggingActivity



**Figura 140:** Classe LoggingActivity

**Nome:** LoggingActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È usata per arrestare un'attività di logging avviata e per recuperare i beacon circostanti. Gestisce tutte le possibili richieste effettuate da LoggingView;

**Descrizione:** Classe che estende AppCompatActivity per la gestione dell'interazione tra il model e LoggingView;

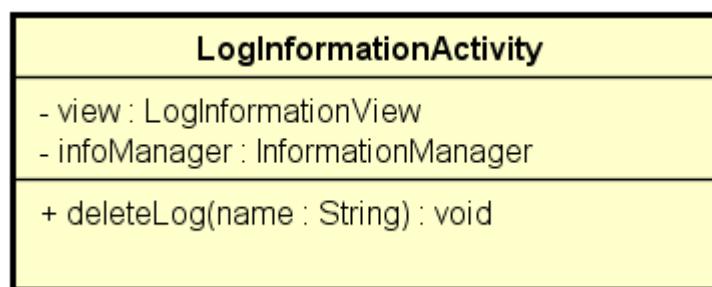
**Attributi:**

- - infoManager : InformationManager  
Riferimento utile per gestire i log
- - view : LoggingView  
View associata a tale Activity

**Metodi:**

- + stopLogging() : void  
Metodo che viene utilizzato per interrompere l'attività di log

#### 4.4.123 presenter::LogInformationActivity



**Figura 141:** Classe LogInformationActivity

**Nome:** LogInformationActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È usata per recuperare tutte le possibili informazioni di un log dal model e renderle disponibili a LogInformationView. Gestisce tutte le possibili richieste effettuate da LogInformationView;

**Descrizione:** Classe che estende AppCompatActivity e gestisce l'interazione tra LogInformationView ed il model;

**Attributi:**

- - infoManager : InformationManager  
Oggetto del Model per la gestione dei log
- - view : LogInformationView  
View associata a tale Activity

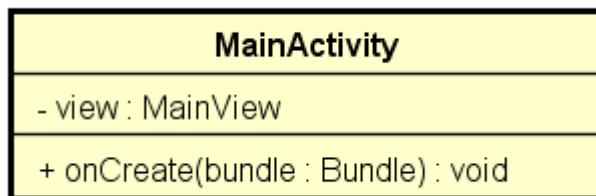
**Metodi:**

- + deleteLog(name : String) : void  
Metodo che viene utilizzato per rimuovere un log salvato

**Argomenti:**

- name : String  
Nome del log da eliminare

#### 4.4.124 presenter::MainActivity



**Figura 142:** Classe MainActivity

**Nome:** MainActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per la gestione delle MainView;

**Descrizione:** Classe che implementa AppCompatActivity per la gestione dell'interazione tra MainView ed il model;

**Attributi:**

- - view : MainView  
View associata a tale Activity

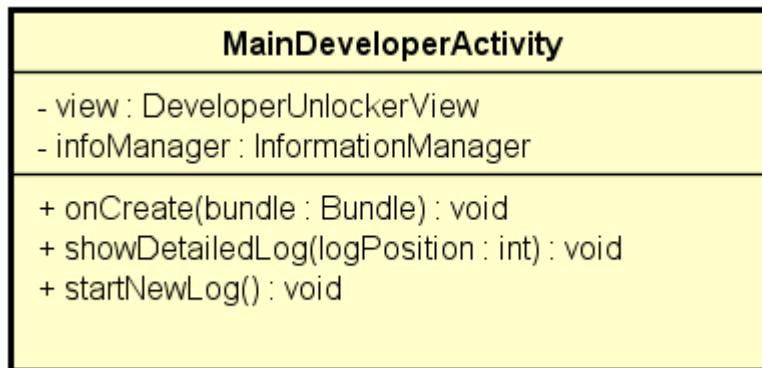
**Metodi:**

- + onCreate(bundle : Bundle) : void  
**Override** Metodo che inizializza la HomeView

**Argomenti:**

- bundle : Bundle  
Componente per salvare lo stato dell'applicazione

#### 4.4.125 presenter::MainDeveloperActivity



**Figura 143:** Classe MainDeveloperActivity

**Nome:** MainDeveloperActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È usata per recuperare i log dal model e avviare la registrazione di un nuovo log. Gestisce tutte le possibili richieste effettuate da MainDeveloperView;

**Descrizione:** È una classe che estende AppCompatActivity e consente di gestire l'interazione tra MainDeveloperView ed il model;

#### Attributi:

- - `infoManager` : `InformationManager`  
Oggetto del Model per la gestione delle informazioni
- - `view` : `DeveloperUnlockerView`  
View associata a tale Activity

#### Metodi:

- + `onCreate(bundle : Bundle) : void`  
**Override** Metodo che inizializza MainDeveloperView

#### Argomenti:

- `bundle : Bundle`  
Componente per salvare lo stato dell'applicazione

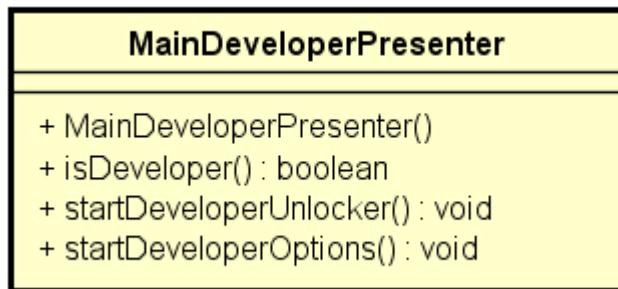
- + `showDetailedLog(logPosition : int) : void`  
Metodo che permette di visualizzare il contenuto di un log

#### Argomenti:

- `logPosition : int`  
Intero rappresentante la posizione del log selezionato all'interno della lista

- + `startNewLog() : void`  
Metodo che avvia un nuovo log

#### 4.4.126 presenter::MainDeveloperPresenter



**Figura 144:** Classe MainDeveloperPresenter

**Nome:** MainDeveloperPresenter;

**Tipo:** Classe;

**Visibilità:** public;

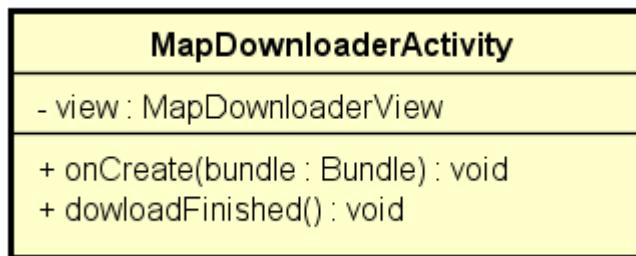
**Utilizzo:** È utilizzata per discriminare la visualizzazione delle funzionalità sviluppatore tra un utente sviluppatore ed un utente che non lo è;

**Descrizione:** Classe che estende AppCompatActivity e controlla utilizzando il model, se l'utente è sviluppatore o meno;

**Metodi:**

- • + isDeveloper() : boolean  
Metodo che permette di verificare se un utente è sviluppatore
- • + MainDeveloperPresenter()  
Costruttore della classe MainDeveloperPresenter
- • + startDeveloperOptions() : void  
Metodo che consente di avviare la gestione delle funzionalità sviluppatore
- • + startDeveloperUnlocker() : void  
Metodo che consente di avviare la gestione dello sblocco delle funzionalità sviluppatore

#### 4.4.127 presenter::MapDownloaderActivity



**Figura 145:** Classe MapDownloaderActivity

**Nome:** MapDownloaderActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per gestire il download o l'aggiornamento di una mappa. Gestisce anche tutte le possibili richieste effettuate da MapDownloaderView;

**Descrizione:** È una classe che estende AppCompatActivity che consente di gestire il download o l'aggiornamento delle mappe ;

**Attributi:**

- - view : MapDownloaderView  
View associata a tale Activity

**Metodi:**

- + downloadFinished() : void  
Metodo che gestisce la View per visualizzare il completamento del download di una mappa
- + onCreate(bundle : Bundle) : void  
Override Metodo che inizializza la View associata

**Argomenti:**

- bundle : Bundle  
Componente per salvare lo stato dell'applicazione

#### 4.4.128 presenter::NavigationActivity

<b>NavigationActivity</b>	
- view : NavigationView	
- navigationAdapter : NavigationAdapter	
+ onCreate(bundle : Bundle) : void	
+ pathError() : void	
+ informationUpdate(info : ProcessedInformation) : void	
+ showDetailedInformation(instructionPosition : int) : void	
+ stopNavigation() : void	

**Figura 146:** Classe NavigationActivity

**Nome:** NavigationActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare tutte le informazioni necessarie alla navigazione e renderle disponibili alla NavigationView. Gestisce anche tutte le richieste che vengono fatte dalla NavigationView;

**Descrizione:** Classe che estende AppCompatActivity per la gestione dell’interazione tra NavigationView ed il model;

**Attributi:**

- - navigationAdapter : NavigationAdapter  
Consente di gestire l’insieme di funzioni per la navigazione
- - view : NavigationView  
View associata a tale Activity

**Metodi:**

- + `informationUpdate(info : ProcessedInformation) : void`

Metodo che permette di aggiornare le informazioni

**Argomenti:**

- `info : ProcessedInformation`

Informazioni utili alla navigazione

- + `onCreate(bundle : Bundle) : void`

**Override** Metodo che inizializza NavigationView

**Argomenti:**

- `bundle : Bundle`

Componente per salvare lo stato dell'applicazione

- + `pathError() : void`

Metodo che viene invocato dal Model per segnalare un errore durante la navigazione

- + `showDetailedInformation(instructionPosition : int) : void`

Metodo che permette la visualizzazione delle informazioni dettagliate

**Argomenti:**

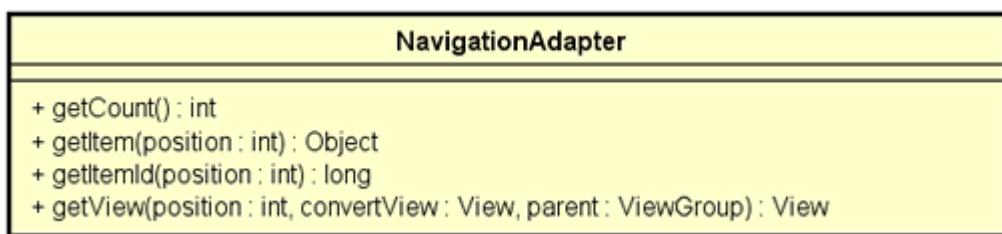
- `instructionPosition : int`

Intero rappresentante la posizione dell'informazione a cui accedere

- + `stopNavigation() : void`

Metodo che permette di interrompere la navigazione

#### 4.4.129 presenter::NavigationAdapter



**Figura 147:** Classe NavigationAdapter

**Nome:** `NavigationAdapter;`

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.widget.BaseAdapter(Android)`.

**Visibilità:** public;

**Utilizzo:** È utilizzata per la gestione dell'interazione tra lista di indicazioni e le possibili azioni che è possibile effettuare su di esse;

**Descrizione:** Classe che estende BaseAdapter per la gestione della lista di istruzioni di navigazione;

**Metodi:**

- `+ getCount() : int`

**Override** Metodo che ritorna il numero di istruzioni relative alla tua destinazione

- `+ getItem(position : int) : Object`

**Override** Metodo che viene utilizzato per recuperare un'istruzione in una certa posizione della lista di istruzioni

**Argomenti:**

- `position : int`

Posizione dell'istruzione da recuperare

- `+ getItemId(position : int) : long`

**Override** Metodo che viene utilizzato per recuperare l'id di un'istruzione in una certa posizione nella lista di istruzioni

**Argomenti:**

- `position : int`

Posizione dell'istruzione di cui recuperare l'identificativo numerico

- `+ getView(position : int, convertView : View, parent : ViewGroup) : View`

**Override** Metodo che viene utilizzato per recuperare un'istruzione in una certa posizione

**Argomenti:**

- `position : int`

Posizione dell'istruzione da recuperare

- `convertView : View`

Layout dell'istruzione recuperata

- parent : ViewGroup  
Layout della lista di istruzioni
- + NavigationAdapter()  
Costruttore della classe NavigationAdapter

#### 4.4.130 presenter::NavigationManagerPresenter

NavigationManagerPresenter	
-	navigationManager : NavigationManager
+	<u>getNavigationManager() : NavigationManager</u>

**Figura 148:** Classe NavigationManagerPresenter

**Nome:** NavigationManagerPresenter;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.app.Application(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per fornire un riferimento al NavigationManager del model;

**Descrizione:** Classe che estende Application e recupera il riferimento al NavigationMager;

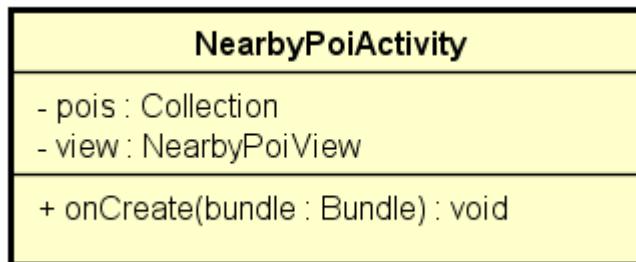
**Attributi:**

- - navigationManager : NavigationManager  
Oggetto del Model utilizzato per gestire la navigazione

**Metodi:**

- + getNavigationManager() : NavigationManager  
Metodo per ottenere l'oggetto NavigationManager

#### 4.4.131 presenter::NearbyPoiActivity



**Figura 149:** Classe NearbyPoiActivity

**Nome:** NearbyPoiActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare tutti i POI rilevati dal dispositivo e gestire tutte le richieste effettuate da NearbyPoiView;

**Descrizione:** Classe che estende AppCompatActivity è gestisce tutte le possibili interazioni tra NearbyPoiView ed il model;

**Attributi:**

- - pois : Collection<PointOfInterest>  
Insieme di POI rilevati nelle circostanze
- - view : NearbyPoiView  
View associata a tale Activity

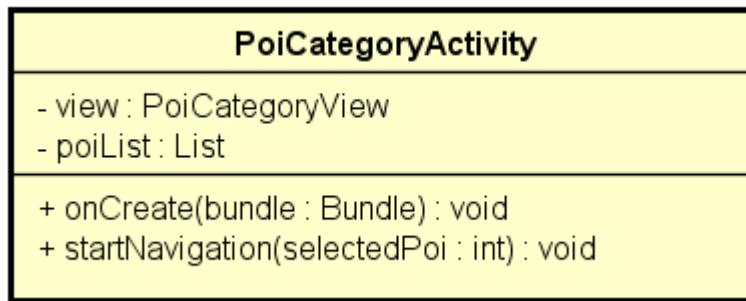
**Metodi:**

- + onCreate(bundle : Bundle) : void  
**Override** Metodo che istanzia NearbyPoiView

**Argomenti:**

- bundle : Bundle  
Componente per salvare lo stato dell'applicazione

#### 4.4.132 presenter::PoiCategoryActivity



**Figura 150:** Classe PoiCategoryActivity

**Nome:** PoiCategoryActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.support.v7.app.AppCompatActivity`(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per recuperare tutte le informazioni di tutti i POI associati ad una certa categoria dal model al fine di popolare la `PoiCategoryView`. Gestisce anche tutte le richieste che vengono fatte dalla `PoiCategoryView`;

**Descrizione:** Classe che implementa `AppCompatActivity` per la gestione dell'interazione tra `PoiCategoryView` ed il model;

**Attributi:**

- - `poiList : List<PointOfInterest>`  
Lista di POI associati ad una certa categoria
- - `view : PoiCategoryView`  
View associata a tale Activity

**Metodi:**

- + `onCreate(bundle : Bundle) : void`  
**Override** Metodo che implementa la `PoiCategoryView`

**Argomenti:**

– bundle : Bundle

Componente per salvare lo stato dell'applicazione

- + startNavigation(selectedPoi : int) : void

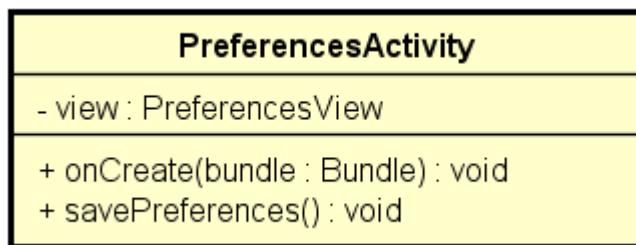
Metodo che permette di avviare la navigazione tramite l'oggetto navigator

**Argomenti:**

– selectedPoi : int

POI da raggiungere selezionato tramite la View

#### 4.4.133 presenter::PreferencesActivity



**Figura 151:** Classe PreferencesActivity

**Nome:** PreferencesActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- android.support.v7.app.AppCompatActivity(Android).

**Visibilità:** public;

**Utilizzo:** È utilizzata per gestire l'interazione tra PreferencesView ed il model. Gestisce tutte le possibili richieste di PreferencesView;

**Descrizione:** È una classe che estende AppCompatActivity che consente di gestire le preferenze utente recuperandole dal model;

**Attributi:**

- - view : PreferencesView  
View associata a tale Activity

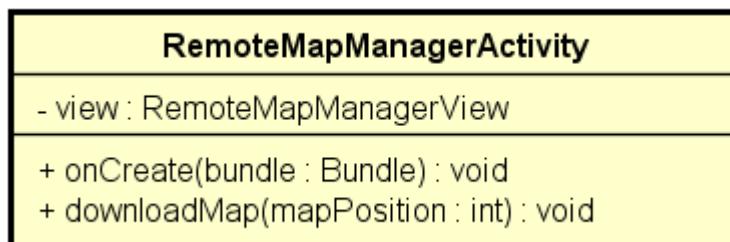
**Metodi:**

- + `onCreate(bundle : Bundle) : void`  
**Override** Metodo che inizializza la View associata

**Argomenti:**

- `bundle : Bundle`  
Componente per salvare lo stato dell'applicazione

- + `savePreferences() : void`  
Metodo che permette di salvare le preferenze utente

**4.4.134 presenter::RemoteMapManagerActivity**

**Figura 152:** Classe RemoteMapManagerActivity

**Nome:** RemoteMapManagerActivity;

**Tipo:** Classe;

**Componenti delle librerie utilizzate:**

- `android.support.v7.app.AppCompatActivity(Android)`.

**Visibilità:** public;

**Utilizzo:** È utilizzata per gestire le mappe in remoto dal model per poterle esporre in RemoteMapManagerView. Gestisce anche tutte le possibili richieste effettuate da RemoteMapManagerView;

**Descrizione:** È una classe che estende AppCompatActivity che recupera tutte le mappe in remoto dal model e gestisce RemoteMapManagerView;

**Attributi:**

- - view : RemoteMapManagerView  
View associata a tale Activity

**Metodi:**

- + downloadMap(mapPosition : int) : void  
Metodo che permette di eseguire il download di una mappa da un database remoto

**Argomenti:**

- mapPosition : int  
Posizione della mappa di cui fare il download

- + onCreate(bundle : Bundle) : void  
**Override** Metodo che inizializza la View associata

**Argomenti:**

- bundle : Bundle  
Componente per salvare lo stato dell'applicazione

**4.4.135 presenter::SearchSuggestionsProvider**

SearchSuggestionsProvider
<pre>-COLUMNS : String - homeActivity : HomeActivity</pre>
<pre>+ query(Uri : uri, projection : String[], selection : String, selectionArgs : String[], sortOrder : String) : Cursor + getType(Uri : uri) : String + insert(Uri : Uri, values : ContentValues) : Uri + update(Uri : Uri, values : ContentValues, selection : String, selectionArgs : String[]) : int + onCreate() : void</pre>

**Figura 153:** Classe SearchSuggestionsProvider**Nome:** SearchSuggestionsProvider;**Tipo:** Classe;**Componenti delle librerie utilizzate:**

- android.content.ContentProvider(Android).

**Visibilità:** public;**Utilizzo:** È usata per la gestione suggerimenti di ricerca per la navigazione;**Descrizione:** Classe che estende content Provider e si occupa della gestione suggerimenti di ricerca per la navigazione;

**Attributi:**

- - COLUMNS : String {readOnly}  
Identifica la struttura di un singolo suggerimento
- - homeActivity : HomeActivity  
Riferimento all'activity che gestisce la ricerca dei POI in base al nome

**Metodi:**

- + getType(Uri : Uri) : String  
**Override** Metodo che ritorna il MIME type associato al parametro passato

**Argomenti:**

- uri : Uri  
URI su cui fare la query

- + insert(Uri, ContentValues) : Uri  
**Override** Metodo che serve per inserire i suggerimenti nel content provider

**Argomenti:**

- uri : Uri  
URI in cui fare l'inserimento
- values : ContentValues  
Insieme di coppie nome-colonna/valore da inserire nel content provider

- + onCreate() : void  
**Override** Metodo che inizializza un oggetto di tipo SearchSuggestionProvider

- + query(Uri, String[], String, String[], String) : Cursor  
**Override** Metodo che serve per popolare i suggerimenti della SearchView in base al testo inserito

**Argomenti:**

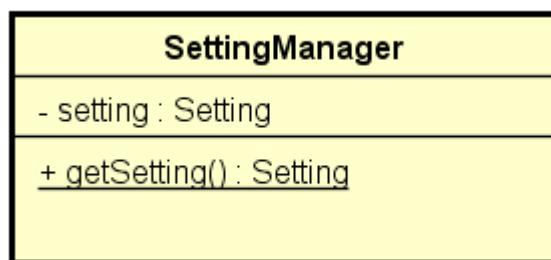
- uri : Uri  
URI su cui fare la query
- projection : String[]  
Lista delle colonne della tabella del content provider

- **selection : String**  
Criterio da applicare per filtrare le righe del content provider
  - **selectionArgs : String[]**  
Insieme di argomenti su cui fare la selezione
  - **sortOrder : String**  
Ordine dei risultati
- + **update(Uri uri, ContentValues values, String selection, String[] selectionArgs) : int**  
**Override** Metodo utilizzato per aggiornare il content provider

**Argomenti:**

- **uri : Uri**  
URI su cui fare la query
- **values : ContentValues**  
Valori da aggiungere
- **selection : String**  
Criterio da applicare per filtrare le righe del content provider
- **selectionArgs : String[]**  
Insieme di argomenti su cui fare la selezione

#### 4.4.136 presenter::SettingManager



**Figura 154:** Classe SettingManager

**Nome:** SettingManager;

**Tipo:** Classe;

**Visibilità:** public;

**Utilizzo:** È usata per fornire un punto di accesso alle impostazioni utente nel model;

**Descrizione:** È una classe che estende AppCompatActivity e consente di recuperare un riferimento verso le impostazioni utente nel model;

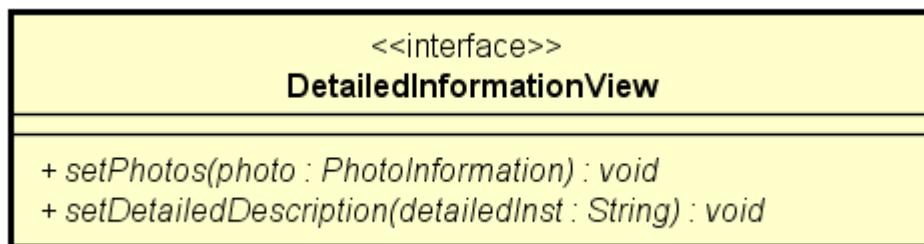
**Attributi:**

- - setting : Setting  
Le impostazioni e preferenze dell'utente

**Metodi:**

- + getSetting() : Setting  
Metodo che viene utilizzato per ottenere le impostazioni e le preferenze dell'utente

#### 4.4.137 view::DetailedInformationView



**Figura 155:** Interfaccia DetailedInformationView

**Nome:** DetailedInformationView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente la versione dettagliata di una certa istruzione e le foto relative al POI destinazione di tale istruzione;

**Metodi:**

- + *setDetailedDescription(detailedInst : String) : void*  
Metodo utilizzato per visualizzare la descrizione dettagliata relativa ad una certa istruzione

**Argomenti:**

- *detailedInst : String*  
Descrizione dettagliata relativa ad una certa istruzione di navigazione

- + *setPhotos(photo : PhotoInformation) : void*

Metodo utilizzato per visualizzare la lista delle anteprime delle foto relative ad un certo POI

**Argomenti:**

- *photo : PhotoInformation*  
Lista degli URI delle immagini relative ad un certo POI

#### 4.4.138 view::DetailedInformationViewImp

DetailedInformationViewImp
- presenter : DetailedInformationActivity
+ setPhotos(urls : List) : void
+ setDetailedDescription(detailedInst : String) : void
+ DetailedInformationViewImp(presenter : DetailedInformationActivity)

**Figura 156:** Classe DetailedInformationViewImp

**Nome:** DetailedInformationViewImp;

**Tipo:** Classe;

**Implementa:**

- DetailedInformationView.

**Visibilità:** public;

**Utilizzo:** La lista di anteprime deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista possa reagire alla pressione su di esso. Per recuperare un riferimento alla lista è sufficiente utilizzare la classe R.id di Android;

**Descrizione:** Classe che si occupa di mostrare: le anteprime delle foto rappresentanti il prossimo POI, la descrizione dettagliata di una certa informazione di navigazione. La UI legata a questa classe permette all'utente di accedere alle foto rappresentanti il prossimo POI;

**Attributi:**

- - presenter : DetailedInformationActivity  
Presenter della View

**Metodi:**

- + DetailedInformationViewImp(presenter : DetailedInformationActivity)  
Costruttore della classe DetailedInformationViewImp

**Argomenti:**

- presenter : DetailedInformationActivity  
Presenter della View che viene creata

- + setDetailedDescription(detailedInst : String) : void  
**Override** Metodo utilizzato per visualizzare la descrizione dettagliata relativa ad una certa istruzione

**Argomenti:**

- detailedInst : String  
Descrizione dettagliata relativa ad una certa istruzione di navigazione

- + setPhotos(urls : List<String>) : void  
**Override** Metodo utilizzato per visualizzare la lista delle anteprime delle foto relative ad un certo POI

**Argomenti:**

- urls : List<String>  
Lista degli URI delle immagini relative ad un certo POI

#### 4.4.139 view::DeveloperUnlockerView



**Figura 157:** Interfaccia DeveloperUnlockerView

**Nome:** DeveloperUnlockerView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per mostrare un errore all'utente, nel caso il codice sviluppatore inserito non sia corretto;

**Metodi:**

- + *showWrongCode()* : void

Metodo utilizzato per visualizzare un errore relativo all'errato inserimento del codice sviluppatore

#### 4.4.140 view::DeveloperUnlockerViewImp

DeveloperUnlockerViewImp	
- presenter : DeveloperUnlockerActivity	
- plainTxtCode : EditText	
- txtInsertCode : TextView	
- btnInsertCode : Button	
+ DeveloperUnlockerViewImp(presenter : DeveloperUnlockerActivity)	
+ showWrongCode() : void	

**Figura 158:** Classe DeveloperUnlockerViewImp

**Nome:** DeveloperUnlockerViewImp;

**Tipo:** Classe;

**Implementa:**

- DeveloperUnlockerView.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti agli elementi di layout che compongono la UI dell'inserimento. Tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso;

**Descrizione:** Classe che si occupa di mostrare i campi utili all'inserimento del codice sviluppatore ;

#### Attributi:

- - `btnInsertCode` : `Button`  
Bottone per confermare l'inserimento del codice sviluppatore
- - `plainTxtCode` : `EditText`  
`EditText` in cui è possibile inserire il codice sviluppatore
- - `presenter` : `DeveloperUnlockerActivity`  
Presenter della View
- - `txtInsertCode` : `TextView`  
`TextView` all'interno della quale viene visualizzato un suggerimento per l'inserimento del codice sviluppatore

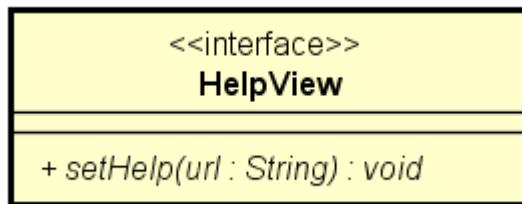
#### Metodi:

- + `DeveloperUnlockerViewImp(presenter : DeveloperUnlokerActivity)`  
Costruttore della classe `DeveloperUnlockerViewImp`

#### Argomenti:

- `presenter` : `DeveloperUnlokerActivity`  
Presenter della View che viene creata
- + `showWrongCode() : void`  
**Override** Metodo utilizzato per visualizzare un errore relativo all'errato inserimento del codice sviluppatore

#### 4.4.141 view::HelpView



**Figura 159:** Interfaccia HelpView

**Nome:** HelpView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente la guida dell'applicazione;

**Metodi:**

- `+ setHelp(url : String) : void`

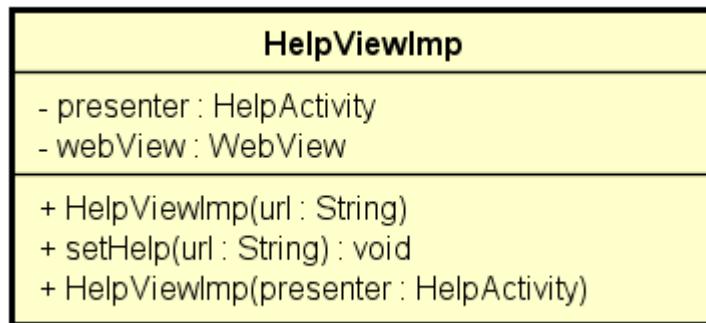
Metodo utilizzato per visualizzare la guida dell'applicazione

**Argomenti:**

- `url : String`

Stringa rappresentante l'URL a cui recuperare la guida dell'applicazione

#### 4.4.142 view::HelpViewImp



**Figura 160:** Classe HelpViewImp

**Nome:** HelpViewImp;

**Tipo:** Classe;

**Implementa:**

- HelpView.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti agli elementi del layout, tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout;

**Descrizione:** Classe che si occupa di mostrare la guida utente dell'applicazione;

**Attributi:**

- - `presenter : HelpActivity`  
Presenter della View
- - `webView : WebView`  
View che permette di visualizzare una pagina web

**Metodi:**

- + `HelpViewImp(url : String)`  
Costruttore della classe HelpViewImp che richiede l'url dove si trova la guida online

**Argomenti:**

– url : String

Stringa rappresentante l'URL a cui recuperare la guida dell'applicazione

- + HelpViewImp(presenter : HomeActivity)

Costruttore della classe HelpViewImp che richiede un'istanza di HomeActivity

**Argomenti:**

– presenter : HomeActivity

Presenter della View che viene creata

- + setHelp(url : String) : void

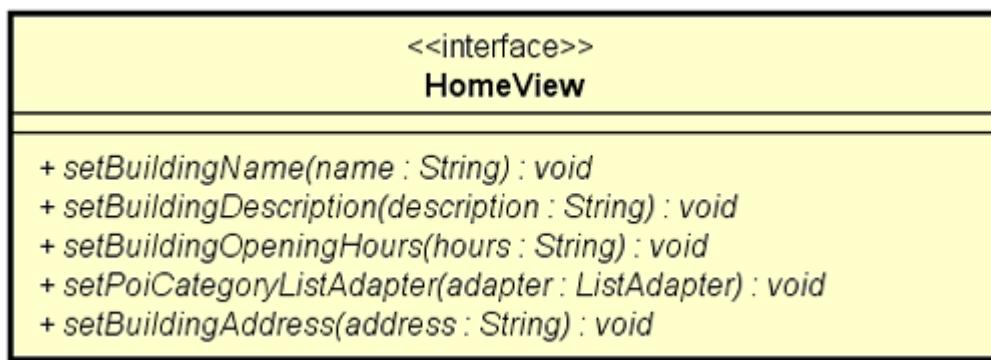
**Override** Metodo utilizzato per visualizzare la guida dell'applicazione

**Argomenti:**

– url : String

Stringa rappresentante l'URL a cui recuperare la guida dell'applicazione

#### 4.4.143 view::HomeView



**Figura 161:** Interfaccia HomeView

**Nome:** HomeView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI della Home;

**Metodi:**

- + *setBuildingAddress(address : String) : void*

Metodo utilizzato per visualizzare l'indirizzo di un edificio

**Argomenti:**

- address : String  
Indirizzo dell'edificio

- + *setBuildingDescription(description : String) : void*

Metodo utilizzato per visualizzare la descrizione di un edificio

**Argomenti:**

- description : String  
Descrizione dell'edificio

- + *setBuildingName(name : String) : void*

Metodo utilizzato per visualizzare il nome di un edificio

**Argomenti:**

- name : String  
Nome dell'edificio

- + *setBuildingOpeningHours(hours : String) : void*

Metodo utilizzato per visualizzare gli orari di apertura di un edificio

**Argomenti:**

- hours : String  
Orari di apertura

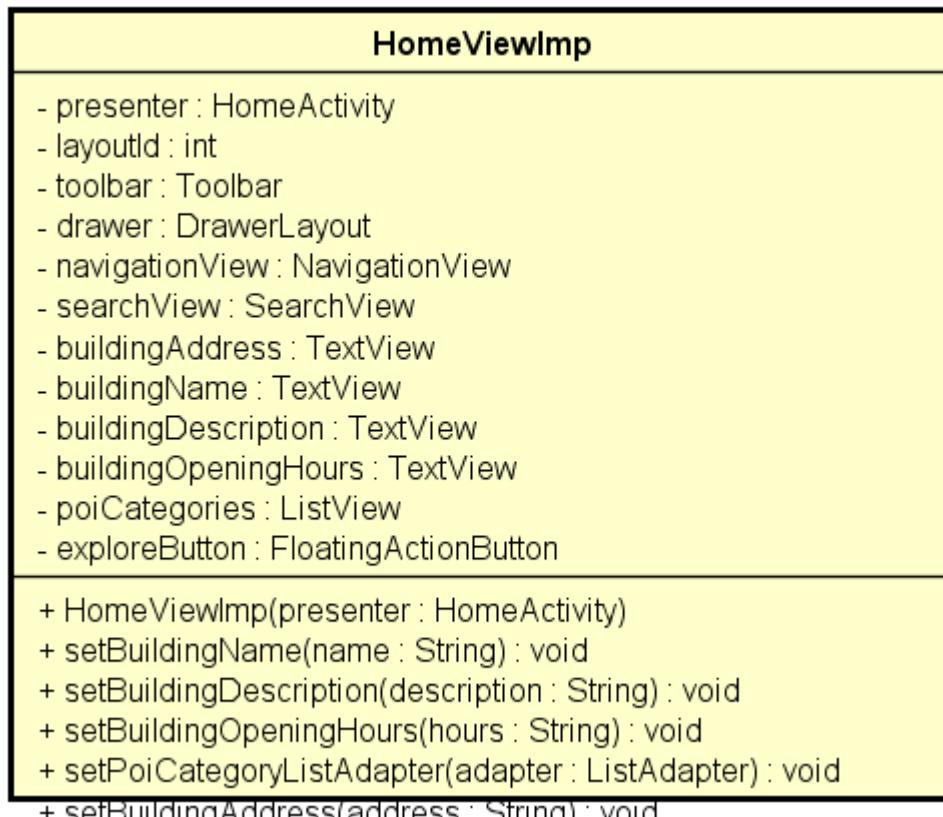
- + *setPoiCategoryListAdapter(adapter : ListAdapter) : void*

Metodo utilizzato per visualizzare la lista di categorie dei POI dell'edificio

**Argomenti:**

- adapter : ListAdapter  
Collegamento tra la lista delle categorie di POI dell'edificio e la view in cui esse devono essere mostrate

#### 4.4.144 view::HomeViewImp



**Figura 162:** Classe HomeViewImp

**Nome:** HomeViewImp;

**Tipo:** Classe;

**Implementa:**

- HomeView.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti agli elementi di layout che compongono la UI della Home. Tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso;

**Descrizione:** Classe che si occupa di mostrare: informazioni relative all'edificio, categorie di POI presenti nell'edificio. La UI legata a questa classe permette all'utente di accedere alle sezioni principali dell'applicazione e alla lista dei POI che si trovano nelle vicinanze dell'utente;

#### Attributi:

- - `buildingAddress` : `TextView`  
TextView all'interno della quale viene visualizzato l'indirizzo dell'edificio
- - `buildingDescription` : `TextView`  
TextView all'interno della quale viene visualizzata la descrizione dell'edificio
- - `buildingName` : `TextView`  
TextView all'interno della quale viene visualizzato il nome dell'edificio
- - `buildingOpeningHours` : `TextView`  
TextView all'interno della quale viene visualizzato l'orario di apertura dell'edificio
- - `drawer` : `DrawerLayout`  
Permette di estrarre delle view dagli angoli dello schermo
- - `exploreButton` : `FloatingActionButton`  
Bottone che permette di trovare i POI intorno all'utente
- - `navigationView` : `NavigationView`  
View che gestisce la navigazione
- - `poiCategories` : `ListView`  
View che si occupa di mostrare tutte le categorie dei POI
- - `presenter` : `HomeActivity`  
Presenter della View
- - `searchView` : `SearchView`  
View che permette la ricerca delle destinazioni di navigazione
- - `toolbar` : `Toolbar`  
Barra degli strumenti che permette la visualizzazione del menu

#### Metodi:

- + `HomeViewImp(presenter : HomeActivity)`  
Costruttore della classe HomeViewImp

#### Argomenti:

---

- **presenter** : HomeActivity  
Presenter della View che viene creata
- + **setBuildingAddress(address : String) : void**  
**Override** Metodo utilizzato per visualizzare l'indirizzo di un edificio

**Argomenti:**

- **address** : String  
Indirizzo dell'edificio
- + **setBuildingDescription(description : String) : void**  
**Override** Metodo utilizzato per visualizzare la descrizione di un edificio

**Argomenti:**

- **description** : String  
Descrizione dell'edificio
- + **setBuildingName(name : String) : void**  
**Override** Metodo utilizzato per visualizzare il nome di un edificio

**Argomenti:**

- **name** : String  
Nome dell'edificio
- + **setBuildingOpeningHours(hours : String) : void**  
**Override** Metodo utilizzato per visualizzare gli orari di apertura di un edificio

**Argomenti:**

- **hours** : String  
Orari di apertura dell'edificio
- + **setPoiCategoryListAdapter(adapter : ListAdapter) : void**  
**Override** Metodo utilizzato per visualizzare la lista di categorie dei POI dell'edificio

**Argomenti:**

- **adapter** : ListAdapter  
Collegamento tra la lista delle categorie di POI dell'edificio e la view in cui esse devono essere mostrate

#### 4.4.145 view::ImageDetailView



**Figura 163:** Interfaccia ImageDetailView

**Nome:** ImageDetailView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente lo slideshow delle immagini relative ad un certo POI;

**Metodi:**

- + *setAdapter(adp : Adapter) : void*

Metodo utilizzato per visualizzare una slideshow di immagini relative al POI da raggiungere

**Argomenti:**

- *adp : Adapter*

Collegamento tra la lista delle immagini e la view in cui esse devono essere mostrate

#### 4.4.146 view::ImageDetailViewImp

ImageDetailViewImp	
- presenter : ImageDetailActivity	
- pager : ViewPager	
+ setAdapter(adp : Adapter) : void	
+ ImageDetailViewImp(presenter : ImageDetailActivity) : void	

**Figura 164:** Classe ImageDetailViewImp

**Nome:** ImageDetailViewImp;

**Tipo:** Classe;

**Implementa:**

- ImageDetailView.

**Visibilità:** public;

**Utilizzo:** Mantiene un riferimento all'elemento di layout utile per passare da una foto alla successiva. alla pressione su di esso.;

**Descrizione:** Classe che si occupa di mostrare una foto relativa ad un certo POI. La UI legata a questa classe permette all'utente di accedere a tutte le foto relative allo stesso POI;

**Attributi:**

- - pager : ViewPager  
Manager del layout che permette di capovolgere la view
- - presenter : ImageDetailActivity  
Presenter della View

**Metodi:**

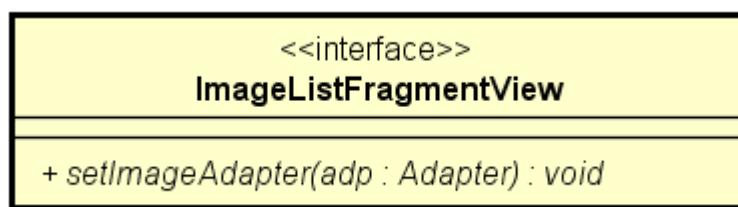
- + ImageDetailViewImp(presenter : ImageDetailActivity) : void  
Costruttore della classe ImageDetailViewImp

**Argomenti:**

- **presenter** : `ImageDetailActivity`  
Presenter della View che viene creata
- + `setAdapter(adp : Adapter) : void`  
**Override** Metodo utilizzato per visualizzare una slideshow di immagini relative al POI da raggiungere

**Argomenti:**

- `adp : Adapter`  
Collegamento tra la lista delle immagini e la view in cui esse devono essere mostrate

**4.4.147 view::ImageListView**

**Figura 165:** Interfaccia ImageListView

**Nome:** `ImageListView`;

**Tipo:** Interfaccia;

**Componenti delle librerie utilizzate:**

- `android.widget.AdapterView.OnItemClickListener`(Android).

**Visibilità:** `public`;

**Utilizzo:** Utilizzata per visualizzare le immagini di un arco necessarie per passare dal nodo di partenza a quello di arrivo;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente le anteprime delle immagini relative al prossimo POI;

**Metodi:**

- + `setImageAdapter(adp : Adapter) : void`  
Metodo utilizzato per visualizzare la lista delle anteprime delle immagine relative ad un certo POI

**Argomenti:**

- adp : Adapter  
Collegamento tra la lista delle immagini e la view in cui esse devono essere mostrate

**4.4.148 view::ImageListFragmentViewImp**

<b>ImageListFragmentViewImp</b>	
- presenter : ImageListFragment	
- listViewImages : ListView	

+ ImageListFragmentViewImp(presenter : ImageListFragment)
+ setImageAdapter(adp : Adapter) : void

**Figura 166:** Classe ImageListFragmentViewImp**Nome:** ImageListFragmentViewImp;**Tipo:** Classe;**Implementa:**

- ImageListFragmentView.

**Visibilità:** public;**Utilizzo:** Mantiene i riferimenti all'elemento di layout che rappresenta la lista di immagini. La lista di immagini deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso;**Descrizione:** Classe che si occupa di mostrare la lista delle anteprime delle immagini relative ad un certo POI. La UI legata a questa classe permette all'utente di accedere alle immagini relative ad un certo POI;**Attributi:**

- - listViewImages : ListView  
Lista delle anteprime delle foto
- - presenter : ImageListFragment  
Presenter della View

**Metodi:**

- + `ImageListFragmentViewImp(presenter : ImageListFragment)`  
Costruttore della classe `ImageListFragmentViewImp`

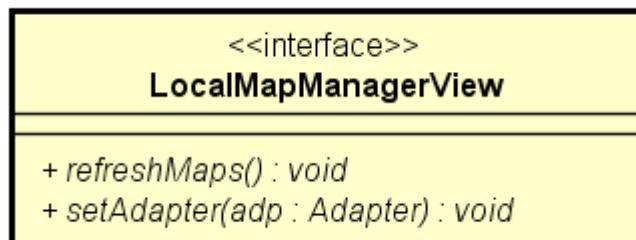
**Argomenti:**

- `presenter : ImageListFragment`  
Presenter della View che viene creata

- + `setImageAdapter(adp : Adapter) : void`  
**Override** Metodo utilizzato per visualizzare la lista delle anteprime delle immagine relative ad un certo POI

**Argomenti:**

- `adp : Adapter`  
Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate

**4.4.149 view::LocalMapManagerView**

**Figura 167:** Interfaccia LocalMapManagerView

**Nome:** `LocalMapManagerView`;

**Tipo:** Interfaccia;

**Visibilità:** `public`;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente le mappe salvate nel database locale. ;

**Metodi:**

- + *refreshMaps()* : void

Metodo che aggiorna la lista delle mappe salvate nel database locale

- + *setAdapter(adp : Adapter)* : void

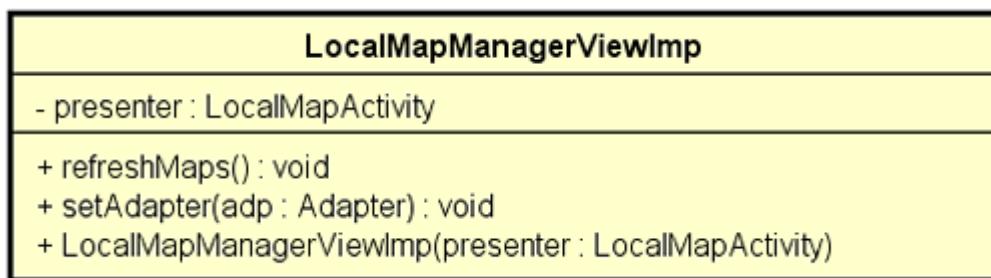
Metodo utilizzato per visualizzare la lista delle mappe salvate nel database locale

**Argomenti:**

- adp : Adapter

Collegamento tra la lista delle mappe salvate nel database locale e la view in cui esse devono essere mostrate

#### 4.4.150 view::LocalMapManagerViewImp



**Figura 168:** Classe LocalMapManagerViewImp

**Nome:** LocalMapManagerViewImp;

**Tipo:** Classe;

**Implementa:**

- LocalMapManagerView.

**Visibilità:** public;

**Utilizzo:** La lista delle mappe deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso.;

**Descrizione:** Classe che si occupa di mostrare le mappe degli edifici salvate nel database locale. La UI legata a questa classe permette all'utente di accedere alle funzionalità di aggiornamento e rimozione di una certa mappa;

**Attributi:**

- - presenter : LocalMapActivity  
Presenter della View

**Metodi:**

- + LocalMapManagerViewImp(presenter : LocalMapActivity)  
Costruttore della classe LocalMapManagerViewImp

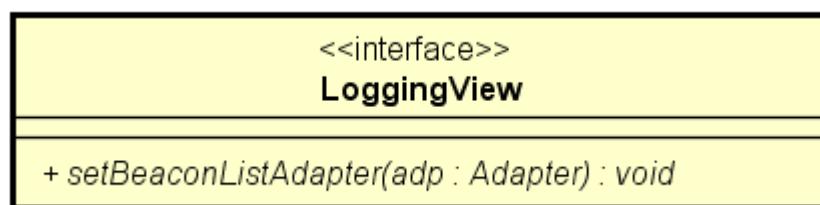
**Argomenti:**

- presenter : LocalMapActivity  
Presenter della View che viene creata

- + refreshMaps() : void  
**Override** Metodo che aggiorna la lista delle mappe salvate nel database locale
- + setAdapter(adp : Adapter) : void  
**Override** Metodo utilizzato per visualizzare la lista delle mappe salvate nel database locale

**Argomenti:**

- adp : Adapter  
Collegamento tra la lista delle mappe salvate nel database locale e la view in cui esse devono essere mostrate

**4.4.151 view::LoggingView**

**Figura 169:** Interfaccia LoggingView

**Nome:** LoggingView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente gli identificativi dei beacon rilevati;

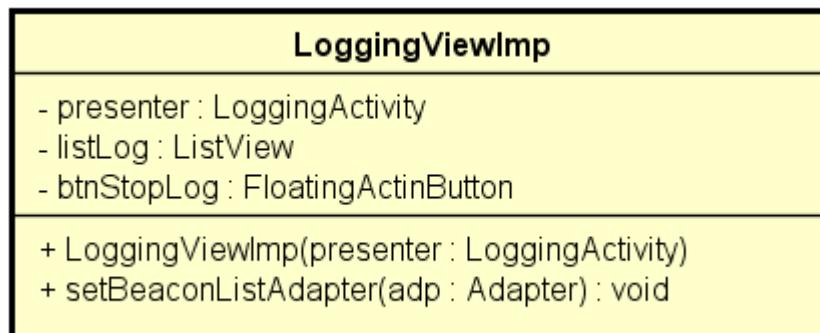
**Metodi:**

- + *setBeaconListAdapter(adp : Adapter) : void*  
Metodo utilizzato per visualizzare la lista dei beacon rilevati

**Argomenti:**

- adp : Adapter  
Collegamento tra la lista dei beacon rilevati e la view in cui essi devono essere mostrati

#### 4.4.152 view::LoggingViewImp



**Figura 170:** Classe LoggingViewImp

**Nome:** LoggingViewImp;

**Tipo:** Classe;

**Implementa:**

- LoggingView.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti agli elementi di layout che compongono la UI, tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad

un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso.;

**Descrizione:** Classe che permette di visualizzare il log in corso e di salvarlo.;

**Attributi:**

- - btnStopLog : FloatingActionButton  
Bottone per interrompere un log in corso
- - listLog : ListView  
Lista di log salvati sul dispositivo
- - presenter : LoggingActivity  
Presenter della View

**Metodi:**

- + LoggingViewImp(presenter : LoggingActivity)  
Costruttore della classe LoggingViewImp

**Argomenti:**

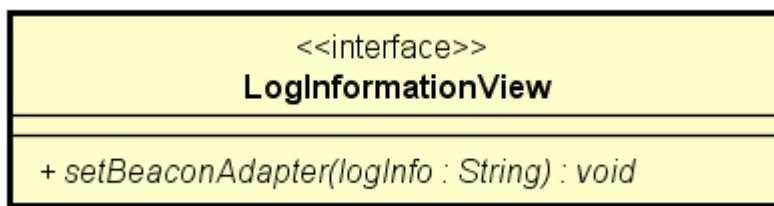
- presenter : LoggingActivity  
Presenter della View che viene creata

- + setBeaconListAdapter(adp : Adapter) : void  
**Override** Metodo utilizzato per visualizzare la lista dei beacon rilevati

**Argomenti:**

- adp : Adapter  
Collegamento tra la lista dei beacon rilevati e la view in cui essi devono essere mostrati

#### 4.4.153 view::LogInformationView



**Figura 171:** Interfaccia LogInformationView

**Nome:** LogInformationView;

---

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente le informazioni relative ad un singolo beacon;

**Metodi:**

- + *setBeaconAdapter(logInfo : String) : void*

Metodo utilizzato per visualizzare la lista delle informazioni di un certo beacon

**Argomenti:**

- *logInfo : String*

Collegamento tra la lista delle informazioni dei log e la view in cui esse devono essere mostrate

#### 4.4.154 view::LogInformationViewImp

LogInformationViewImp
- presenter : LogInformationActivity - txtLog : TextView - btnDeleteLog : FloatingActionButton
+ LogInformationViewImp(presenter : LogInformationActivity) + setBeaconAdapter(logInfo : String) : void

**Figura 172:** Classe LogInformationViewImp

**Nome:** LogInformationViewImp;

**Tipo:** Classe;

**Implementa:**

- LogInformationView.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti agli elementi di layout che compongono la UI, tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso;

**Descrizione:** Classe che si occupa di mostrare i dettagli relativi ad un singolo log. La UI legata a questa classe permette all'utente di eliminare il log ;

#### Attributi:

- - `btnDeleteLog` : `FloatingActionButton`  
Bottone per rimuovere un log salvato
- - `presenter` : `LogInformationActivity`  
Presenter della View
- - `txtLog` : `TextView`  
TextView all'interno della quale viene visualizzato il contenuto del log

#### Metodi:

- + `LogInformationViewImp(presenter : LogInformationActivity)`  
Costruttore della classe LogInformationViewImp

#### Argomenti:

- `presenter` : `LogInformationActivity`  
Presenter della View che viene creata

- + `setBeaconAdapter(logInfo : String) : void`  
**Override** Metodo utilizzato per visualizzare la lista delle informazioni di un certo beacon

#### Argomenti:

- `logInfo` : `String`  
Collegamento tra la lista delle informazioni dei log e la view in cui esse devono essere mostrate

#### 4.4.155 view::MainDeveloperView



**Figura 173:** Interfaccia MainDeveloperView

**Nome:** MainDeveloperView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente la lista dei log salvati e disponibili ad essere consultati;

**Metodi:**

- + *setLogsAdapter(adp : Adapter) : void*

Metodo utilizzato per visualizzare la lista di tutti i log salvati in locale

**Argomenti:**

- *adp : Adapter*

Collegamento tra la lista dei log e la view in cui essi devono essere mostrati

#### 4.4.156 view::MainDeveloperViewImp

MainDeveloperViewImp
- presenter : MainDeveloperActivity - logStartBtn : FloatingActionButton - logList : ListView
+ MainDeveloperViewImp(presenter : MainDeveloperActivity) + setLogsAdapter(adp : Adapter) : void

**Figura 174:** Classe MainDeveloperViewImp

**Nome:** MainDeveloperViewImp;

**Tipo:** Classe;

**Implementa:**

- MainDeveloperView.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti agli elementi del layout, tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso. Per lo stesso motivo, la lista dei log salvati deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener;

**Descrizione:** Classe che si occupa di mostrare la lista di log salvati. La UI legata a questa classe permette all'utente di: accedere alle informazioni di un certo log o avviare un nuovo log;

**Attributi:**

- - logList : ListView  
View che mostra la lista dei log
- - logStartBtn : FloatingActionButton  
Bottone che permette di attivare un log
- - presenter : MainDeveloperActivity  
Presenter della View

**Metodi:**

- + MainDeveloperViewImp(presenter : MainDeveloperActivity)  
Costruttore della classe MainDeveloperViewImp

**Argomenti:**

- presenter : MainDeveloperActivity  
Presenter della View che viene creata

- + setLogsAdapter(adp : Adapter) : void  
**Override** Metodo utilizzato per visualizzare la lista di tutti i log salvati in locale

**Argomenti:**

- adp : Adapter  
Collegamento tra la lista dei log e la view in cui essi devono essere mostrati

**4.4.157 view::MainView**

**Figura 175:** Interfaccia MainView

**Nome:** MainView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI relativa alla schermata di avvio dell'applicazione;

**Metodi:**

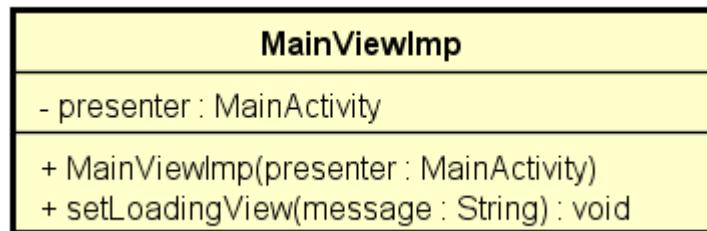
- + *setLoadingView(message : String) : void*

Metodo utilizzato per visualizzare la schermata di caricamento dell'applicazione

**Argomenti:**

- message : String  
Messaggio di caricamento

#### 4.4.158 view::MainViewImp



**Figura 176:** Classe MainViewImp

**Nome:** MainViewImp;

**Tipo:** Classe;

**Implementa:**

- MainView.

**Visibilità:** public;

**Utilizzo:** Questa classe viene utilizzata per eseguire tutte le operazioni CPU-intensive necessarie all'avvio dell'applicazione.;

**Descrizione:** Classe che si occupa di mostrare la schermata di caricamento dell'applicazione, eventuali operazioni CPU-intensive possono essere eseguite qui.;

**Attributi:**

- - presenter : MainActivity  
Presenter della View

**Metodi:**

- + MainViewImp(presenter : MainActivity)

Costruttore della classe MainViewImp

**Argomenti:**

- presenter : MainActivity

Presenter della View che viene creata

- + setLoadingView(message : String) : void

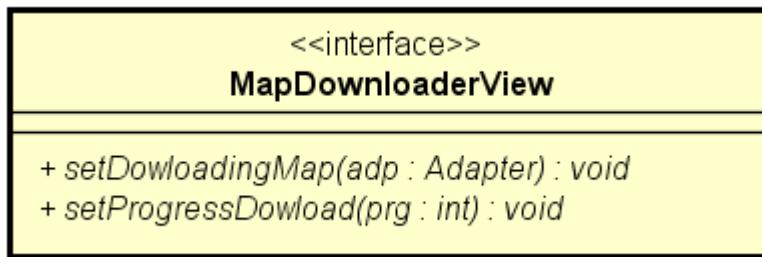
**Override** Metodo utilizzato per visualizzare la schermata di caricamento dell'applicazione

**Argomenti:**

- message : String

Messaggio di caricamento

#### 4.4.159 view::MapDownloaderView



**Figura 177:** Interfaccia MapDownloaderView

**Nome:** MapDownloaderView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI che mostra il progresso durante il download della mappa di un edificio dal server;

**Metodi:**

- + *setDowloadingMap(adp : Adapter) : void*

Metodo utilizzato per visualizzare la mappa che si sta scaricando

**Argomenti:**

– adp : Adapter

Collegamento tra la mappa che si sta scaricando e la view in cui essa deve essere mostrata

- + *setProgressDowload(prg : int) : void*

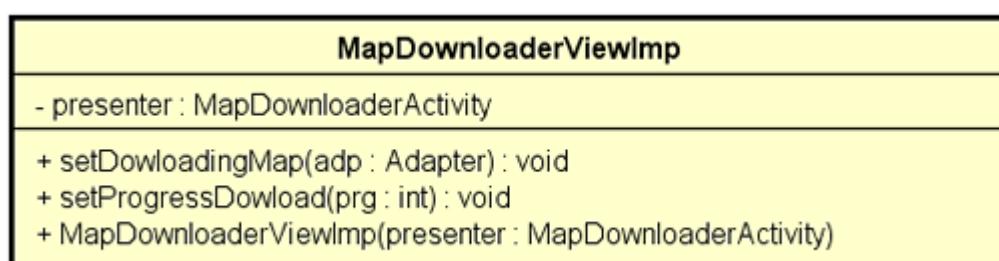
Metodo utilizzato per visualizzare il progresso nel download di una mappa

**Argomenti:**

– prg : int

Attuale progresso del download

#### 4.4.160 view::MapDownloaderViewImp



**Figura 178:** Classe MapDownloaderViewImp

**Nome:** MapDownloaderViewImp;

**Tipo:** Classe;

**Implementa:**

- MapDownloaderView.

**Visibilità:** public;

**Utilizzo:** I dettagli della mappa in download ed il progresso del download stesso possono essere mostrati utilizzando i metodi esposti da questa classe;

**Descrizione:** Classe che si occupa di mostrare il progresso del download di una mappa.;

**Attributi:**

- - **presenter** : MapDownloaderActivity  
Presenter della View

**Metodi:**

- + MapDownloaderViewImp(presenter : MapDownloaderActivity)  
Costruttore della classe MapDownloaderViewImp

**Argomenti:**

- presenter : MapDownloaderActivity  
Presenter della View che viene creata

- + setDownloadingMap(adp : Adapter) : void  
**Override** Metodo utilizzato per visualizzare la mappa che si sta scaricando

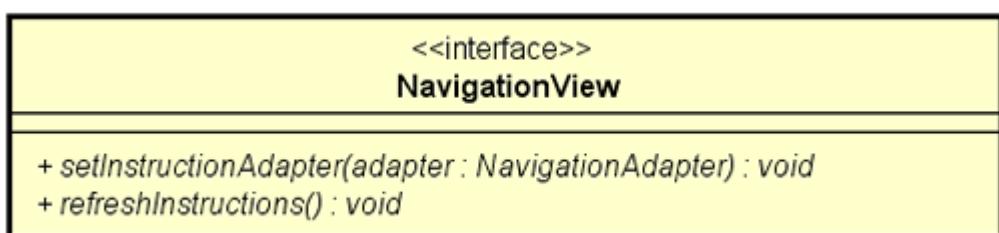
**Argomenti:**

- adp : Adapter  
Collegamento tra la mappa che si sta scaricando e la view in cui essa deve essere mostrata

- + setProgressDowload(prg : int) : void  
**Override** Metodo utilizzato per visualizzare il progresso nel download di una mappa

**Argomenti:**

- prg : int  
Attuale progresso del download

**4.4.161 view::NavigationView****Figura 179:** Interfaccia NavigationView

**Nome:** NavigationView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente le istruzioni di navigazione per raggiungere una certa destinazione;

**Metodi:**

- + *refreshInstructions() : void*  
Metodo utilizzato per aggiornare la lista di istruzioni
- + *setInstructionAdapter(adapter : NavigationAdapter) : void*  
Metodo utilizzato per visualizzare la lista di istruzioni utili a raggiungere un certo POI

**Argomenti:**

- *adapter : NavigationAdapter*  
Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate

#### 4.4.162 view::NavigationViewImp

NavigationViewImp	
- presenter : NavigationActivity	
- instructionAdapter : Adapter	
+ setInstructionAdapter(adapter : NavigationAdapter) : void	
+ refreshInstructions() : void	
+ NavigationViewImp(presenter : NavigationActivity)	

**Figura 180:** Classe NavigationViewImp

**Nome:** NavigationViewImp;

**Tipo:** Classe;

**Implementa:**

- NavigationView.

**Visibilità:** public;

**Utilizzo:** La lista di istruzioni deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista possa reagire alla pressione su di esso. Per recuperare un riferimento alla lista è sufficiente utilizzare la classe R.id di Android;

**Descrizione:** Classe che si occupa di mostrare la lista di istruzioni di navigazione utili per raggiungere un determinato POI. La UI legata a questa classe permette all'utente di accedere alle descrizioni dettagliate delle varie istruzioni;

**Attributi:**

- - `instructionAdapter` : `Adapter`  
Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate
- - `presenter` : `NavigationActivity`  
Presenter della View

**Metodi:**

- + `NavigationViewImp(presenter : NavigationActivity)`  
Costruttore della classe NavigationViewImp

**Argomenti:**

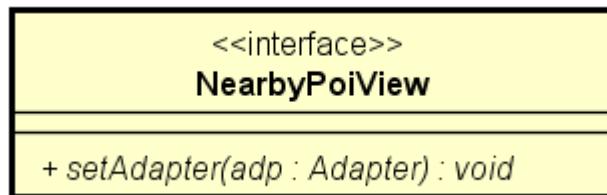
- `presenter` : `NavigationActivity`  
Presenter della View che viene creata

- + `refreshInstructions() : void`  
**Override** Metodo utilizzato per aggiornare la lista di istruzioni
- + `setInstructionAdapter(adapter : NavigationAdapter) : void`  
**Override** Metodo utilizzato per visualizzare la lista di istruzioni utili a raggiungere un certo POI

**Argomenti:**

- `adapter` : `NavigationAdapter`  
Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate

#### 4.4.163 view::NearbyPoiView



**Figura 181:** Interfaccia NearbyPoiView

**Nome:** NearbyPoiView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente la lista dei POI nelle vicinanze dell'utente;

**Metodi:**

- + *setAdapter(adp : Adapter) : void*

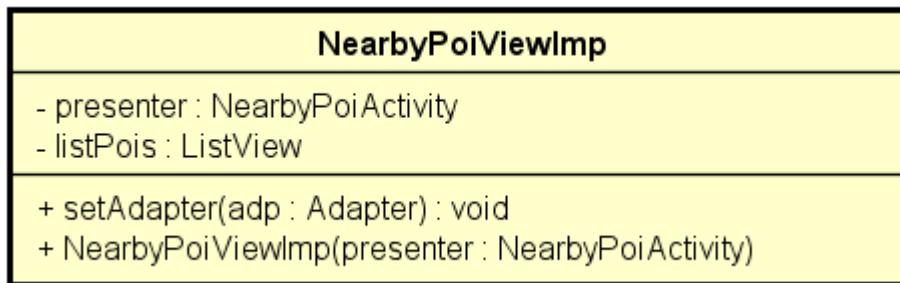
Metodo utilizzato per visualizzare tutti i POI nelle circostanze dell'utente

**Argomenti:**

- adp : Adapter

Collegamento tra la lista dei POI circostanti l'utente e la view in cui essi devono essere mostrati

#### 4.4.164 view::NearbyPoiViewImp



**Figura 182:** Classe NearbyPoiViewImp

**Nome:** NearbyPoiViewImp;

**Tipo:** Classe;

**Implementa:**

- `NearbyPoiView`.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti all'elemento di layout che rappresenta la lista di POI. La lista di POI deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso;

**Descrizione:** Classe che si occupa di mostrare i POI situati nelle vicinanze dell'utente. La UI legata a questa classe permette all'utente di accedere alle informazioni di un certo POI;

**Attributi:**

- `- listPois : ListView`  
View che mostra la lista di POI nelle vicinanze dell'utente
- `- presenter : NearbyPoiActivity`  
Presenter della View

**Metodi:**

- `+ NearbyPoiViewImp(presenter : NearbyPoiActivity)`  
Costruttore della classe NearbyPoiViewImp

**Argomenti:**

- presenter : NearbyPoiActivity  
Presenter della View che viene creata
- + setAdapter(adp : Adapter) : void  
**Override** Metodo utilizzato per visualizzare tutti i POI nelle circostanze dell'utente

**Argomenti:**

- adp : Adapter  
Collegamento tra la lista dei POI circostanti l'utente e la view in cui essi devono essere mostrati

#### 4.4.165 view::PoiCategoryView



**Figura 183:** Interfaccia PoiCategoryView

**Nome:** PoiCategoryView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente la lista dei POI appartenenti ad una data categoria;

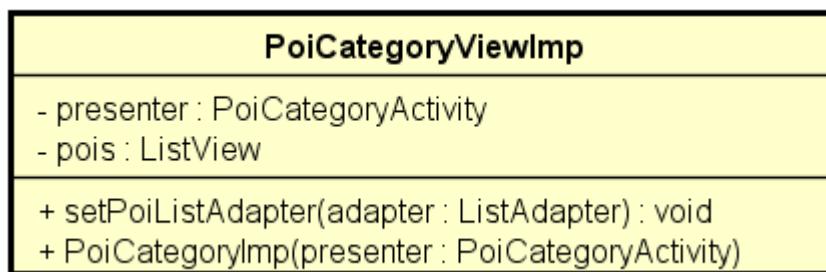
**Metodi:**

- + setPoiListAdapter(adapter : ListAdapter) : void  
Metodo utilizzato per visualizzare tutti i POI appartenenti ad una certa categoria

**Argomenti:**

- **adapter : ListAdapter**  
Collegamento tra la lista delle categorie dei POI e la view in cui essi devono essere mostrati

#### 4.4.166 view::PoiCategoryViewImp



**Figura 184:** Classe PoiCategoryViewImp

**Nome:** PoiCategoryViewImp;

**Tipo:** Classe;

**Implementa:**

- `PoiCategoryView`.

**Visibilità:** public;

**Utilizzo:** Mantiene i riferimenti all'elemento di layout che rappresenta la lista di POI. La lista di POI deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso;

**Descrizione:** Classe che si occupa di mostrare la lista dei POI relativi ad una certa categoria. La UI legata a questa classe permette all'utente di accedere alle informazioni di un certo POI appartenente alla categoria.;

**Attributi:**

- `-pois : ListView`  
View che permette di visualizzare la lista delle categorie di POI
- `-presenter : PoiCategoryActivity`  
Presenter della View

**Metodi:**

- + PoiCategoryViewImp(presenter : PoiCategoryActivity)  
Costruttore della classe PoiCategoryViewImp

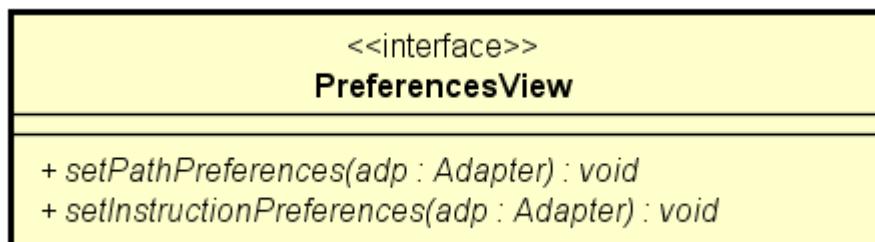
**Argomenti:**

- presenter : PoiCategoryActivity  
Presenter della View che viene creata

- + setPoiListAdapter(adapter : ListAdapter) : void  
**Override** Metodo utilizzato per visualizzare tutti i POI appartenenti ad una certa categoria

**Argomenti:**

- adapter : ListAdapter  
Collegamento tra la lista delle categorie dei POI e la view in cui essi devono essere mostrati

**4.4.167 view::PreferencesView**

**Figura 185:** Interfaccia PreferencesView

**Nome:** PreferencesView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente le preferenze dell'utente rispetto al percorso consigliato e alla modalità di fruizione delle istruzioni di navigazione;

**Metodi:**

- + *setInstructionPreferences(adp : Adapter) : void*  
Metodo utilizzato per visualizzare le preferenze dell'utente riguardo la fruizione delle istruzioni di navigazione

**Argomenti:**

- adp : Adapter

Collegamento tra le preferenze riguardanti la fruizione delle istruzioni di navigazione e la view in cui esse devono essere mostrate

- + *setPathPreferences(adp : Adapter) : void*

Metodo utilizzato per visualizzare le preferenze dell'utente relative al percorso proposto

**Argomenti:**

- adp : Adapter

Collegamento tra le preferenze riguardanti le preferenze del percorso di navigazione e la view in cui esse devono essere mostrate

#### 4.4.168 view::PreferencesViewImp

PreferencesViewImp
- presenter : PreferencesActivity
+ PreferencesViewImp(presenter : PreferencesActivity)
+ setPathPreferences(adp : Adapter) : void
+ setInstructionPreferences(adp : Adapter) : void

**Figura 186:** Classe PreferencesViewImp

**Nome:** PreferencesViewImp;

**Tipo:** Classe;

**Implementa:**

- PreferencesView.

**Visibilità:** public;

**Utilizzo:** Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso. Per recuperare un riferimento alla lista è sufficiente utilizzare la classe R.id di Android;

**Descrizione:** Classe che si occupa di mostrare la UI utile alla modifica delle preferenze dell'utente;

**Attributi:**

- - presenter : PreferencesActivity  
Presenter della View

**Metodi:**

- + PreferencesViewImp(presenter : PreferencesActivity, presenter : PreferencesActivity)  
Costruttore della classe PreferencesViewImp

**Argomenti:**

- presenter : PreferencesActivity  
Presenter della View che viene creata
- presenter : PreferencesActivity  
Presenter della View che viene creata

- + setInstructionPreferences(adp : Adapter) : void

**Override** Metodo utilizzato per visualizzare le preferenze dell'utente riguardo la fruizione delle istruzioni di navigazione

**Argomenti:**

- adp : Adapter  
Collegamento tra le preferenze riguardanti la fruizione delle istruzioni di navigazione e la view in cui esse devono essere mostrate

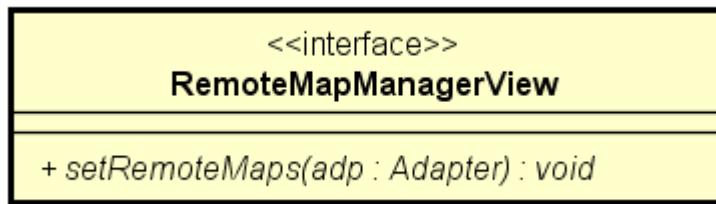
- + setPathPreferences(adp : Adapter) : void

**Override** Metodo utilizzato per visualizzare le preferenze dell'utente relative al percorso proposto

**Argomenti:**

- adp : Adapter  
Collegamento tra le preferenze riguardanti le preferenze del percorso di navigazione e la view in cui esse devono essere mostrate

#### 4.4.169 view::RemoteMapView



**Figura 187:** Interfaccia RemoteMapView

**Nome:** RemoteMapView;

**Tipo:** Interfaccia;

**Visibilità:** public;

**Utilizzo:** È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

**Descrizione:** Interfaccia che espone i metodi per aggiornare la UI contenente le mappe delle quali è possibile effettuare il download dal server;

**Metodi:**

- + *setRemoteMaps(adp : Adapter) : void*

Metodo utilizzato per visualizzare le mappe che è possibile scaricare da un server remoto

**Argomenti:**

- adp : Adapter

Collegamento tra la lista delle mappe che è possibile scaricare e la view in cui esse devono essere mostrate

#### 4.4.170 view::RemoteMapViewImp

RemoteMapViewImp
- presenter : RemoteMapManagerActivity
+ setRemoteMaps(adp : Adapter) : void
+ RemoteMapViewImp(presenter : RemoteMapManagerActivity)

**Figura 188:** Classe RemoteMapViewImp

**Nome:** RemoteMapViewImp;

**Tipo:** Classe;

**Implementa:**

- RemoteMapView.

**Visibilità:** public;

**Utilizzo:** La lista delle mappe deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso.;

**Descrizione:** Classe che si occupa di mostrare le mappe degli edifici disponibili al download. La UI legata a questa classe permette all'utente di accedere alle funzionalità di download di una certa mappa;

**Attributi:**

- - presenter : RemoteMapManagerActivity  
Presenter della View

**Metodi:**

- + RemoteMapViewImp(presenter : RemoteMapManagerActivity)  
Costruttore della classe RemoteMapViewImp

**Argomenti:**

- presenter : RemoteMapManagerActivity  
Presenter della View che viene creata

- + setRemoteMaps(adp : Adapter) : void

**Override** Metodo utilizzate per visualizzare le mappe che è possibile scaricare da un server remoto

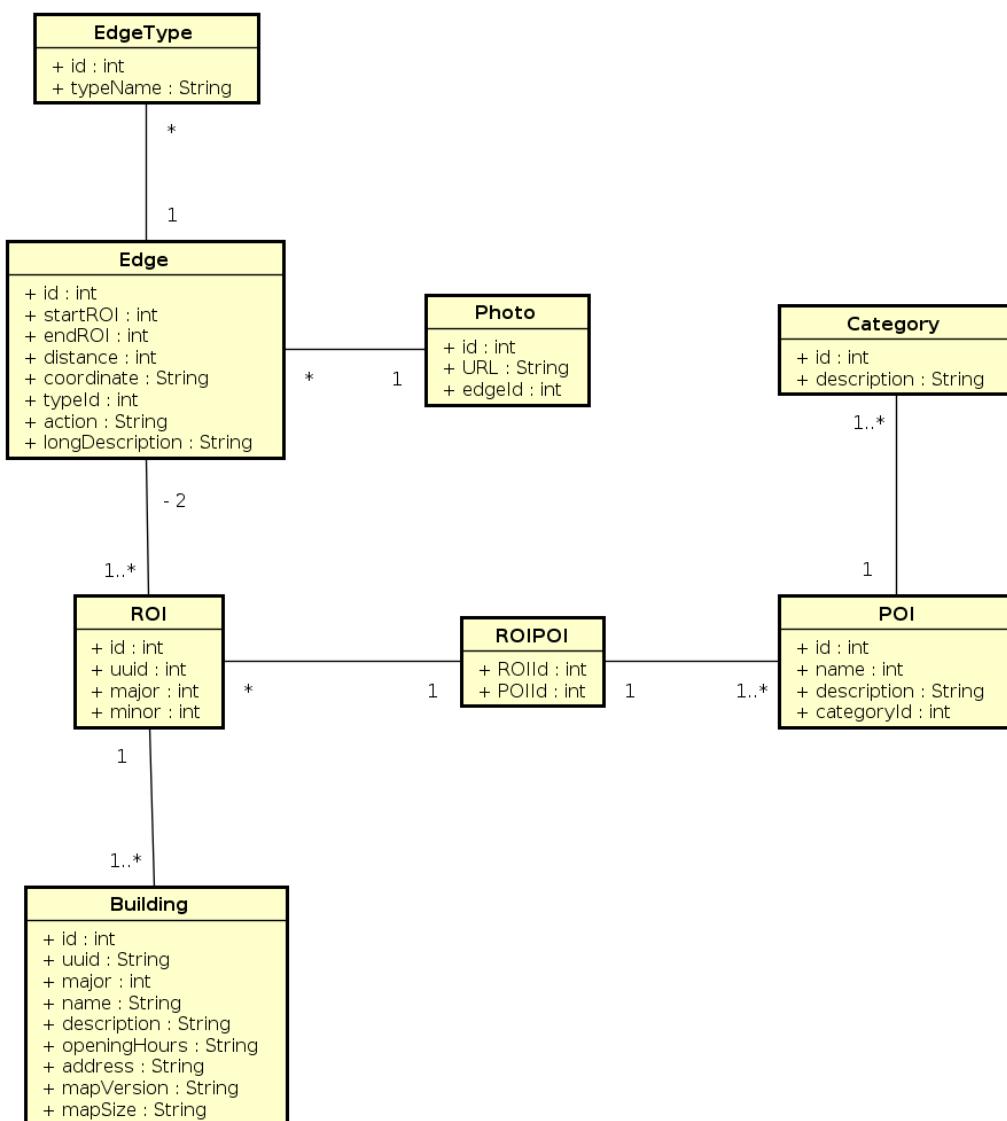
**Argomenti:**

– adp : Adapter

Collegamento tra la lista delle mappe che è possibile scaricare e la view in cui esse devono essere mostrate

## 5 Schema base di dati

Di seguito viene presentata lo schema in UML della base di dati implementata nell'applicativo con SQLite e gestito dal componente **DataManager** e implementata nel server remoto. Lo schema illustra le relazioni tra le entità che costituiscono il grafo rappresentato l'edificio di interesse. Si fa notare che la base di dati non memorizza separatamente gli elementi che compongono i grafici.



**Figura 189:** Schema UML - base di dati

## 6 Diagrammi di sequenza

In questa sezione vengono descritte e rappresentate tramite diagrammi di sequenza UML le sequenze di azioni ritenute più significative con lo scopo di facilitare la comprensione delle comunicazioni tra oggetti facenti parte dell'applicativo Android,. Per quest'ultimo motivo i diagrammi di sequenza non rappresentano l'effettiva realtà ma una versione semplificata e che non rifletterà in tutto l'implementazione.

### 6.1 Avvio Service per il rilevamento beacon

Il diagramma in figura 190 rappresenta l'avvio del service, che si occupa del rilevamento dei beacon, funzionalità focale dell'intero applicativo.

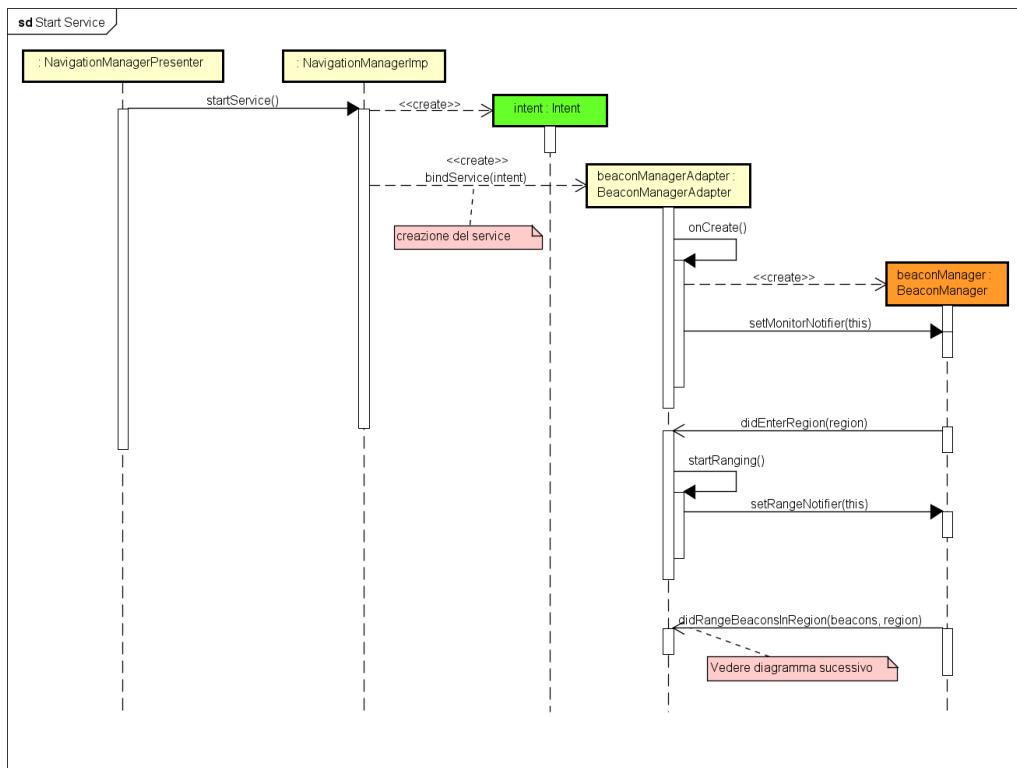
La classe `NavigationManagerPresenter` invoca il metodo `startService()` su `NavigationManagerImp`, all'interno del metodo viene istanziato un oggetto `intent` di tipo `Intent` necessario per creare effettivamente un bind service, `BeaconManagerAdapter`, attraverso la chiamata del metodo `bindService()`, passando come parametro `intent`. Nella fase di creazione del service, di tipo `BeaconManagerAdapter` viene chiamato il metodo `onCreate()` nel quale viene creata un'istanza della classe `BeaconManager` offerta dalla libreria `AltBeacon`. Si effettuano inoltre diverse chiamate per il settaggio e la configurazione di `beaconManager` che non sono rappresentate per mantenere il diagramma più leggibile. Una volta settato `beaconManager` l'oggetto `beaconManagerAdapter` si mette in ascolto di `beaconManager` chiamando il metodo `setMonitorNotifier` iniziando la fase di monitoring,.

A questo punto `beaconManagerAdapter` è un listener di `beaconManager` il quale una volta rilevata la region dei beacon in cui il device si trova scatena l'evento `didEnterRegion()` notificando i propri listener, ossia l'oggetto di tipo `beaconManagerAdapter`.

Individuata la region tramite l'evento `beaconManagerAdapter` effettua un controllo per capire se la region è riconosciuta dall'applicativo, se lo è `beaconManagerAdapter` entra nella fase di ranging, in cui saranno raccolti dettagliatamente i dati di tutti i beacon rilevati. `beaconManagerAdapter` si mette in ascolto in modalità ranging di `beaconManager` tramite la chiamata del metodo `setRangeNotifier()`.

A questo punto `beaconManagerAdapter` riceve l'evento di rilevazione beacon attraverso il metodo `didRangeBeaconsInRegion()` il quale restituisce una `Collection` di `Beacon` e la `Region` di appartenenza.

Per la gestione degli elementi all'interno della `Collection` si rimanda al diagramma successivo.



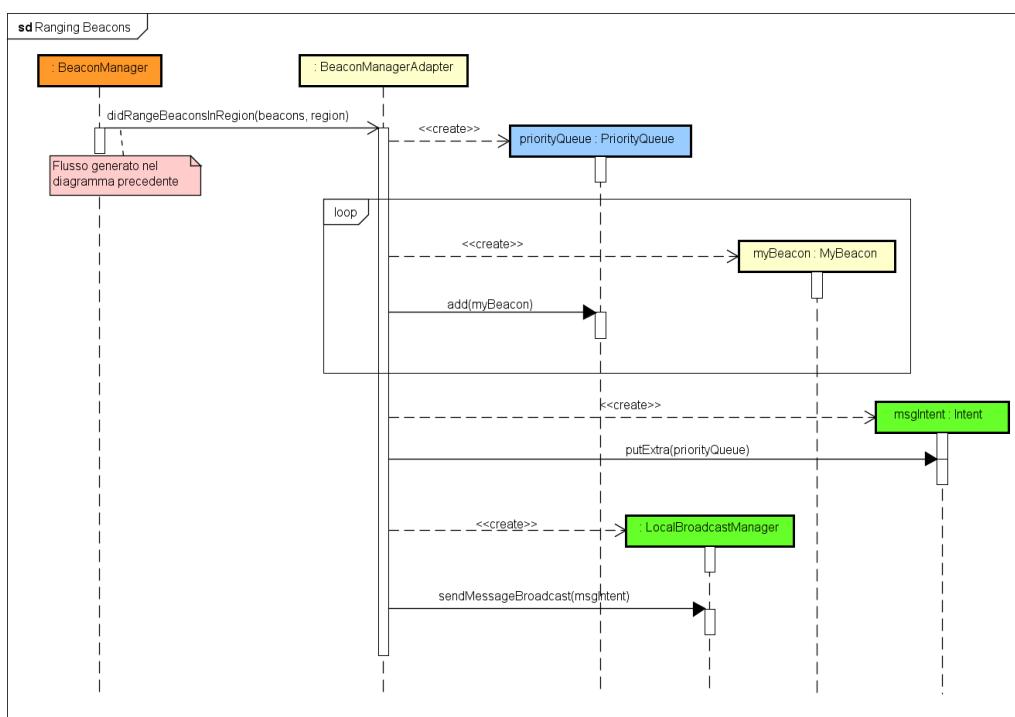
**Figura 190:** Diagramma di sequenza - Avvio di un service<sub>g</sub> per il rilevamento beacon

## 6.2 Elaborazione beacon rilevati e comunicazione broadcast

Il diagramma in figura 191 rappresenta l'interazione che avviene tra i componenti dell'applicativo allo scopo di rilevare dettagliatamente i dati trasmessi dai beacon circostanti al device.

L'oggetto di tipo `BeaconManagerAdapter` è un service, e implementa il listener di `BeaconManager`: `RangeNotifier` il quale scatenerà, dopo una scansione, l'evento `didRangBeaconsInRegion()` passando come parametri una `Collection` di `Beacon` rilevati e la `Region` di appartenenza. I parametri vengono elaborati da `BeaconManagerAdapter` il quale dopo aver creato una `PriorityQueue` costruisce un wrapper, (`MyBeacon`) di ogni `Beacon` aggiungendolo alla `PriorityQueue` tramite `add()`.

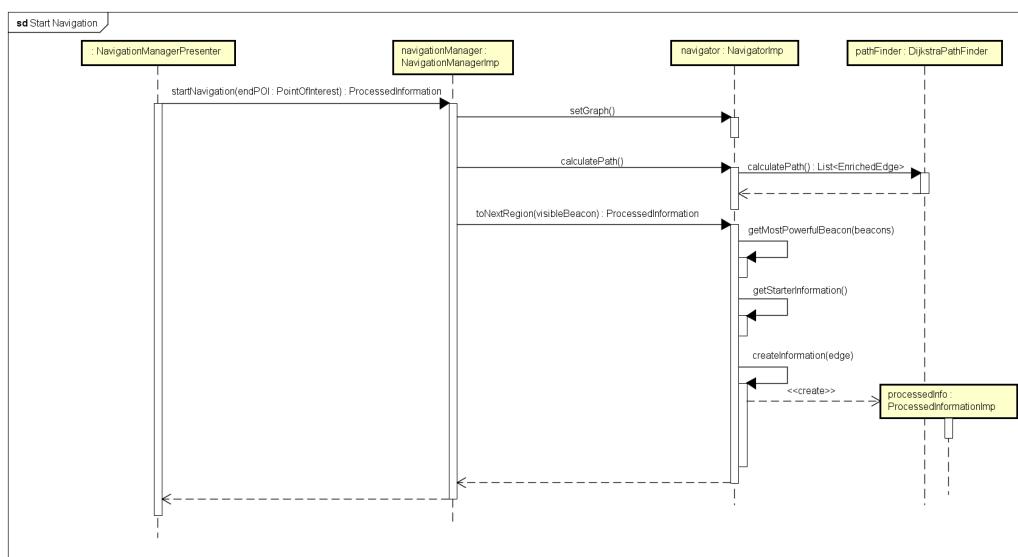
Una volta elaborati tutti i `Beacon` ricevuti `BeaconManagerAdapter` crea un messaggio `Intent` in cui inserisce la `PriorityQueue` tramite la chiamata del metodo `putExtra()`. Costruisce l'oggetto `LocalBroadcastManager` per utilizzarlo nella chiamata del metodo `sendMessageBroadcast()` che si occuperà di inviare l'`Intent` in altre parti dell'applicazione costruite appositamente per ricevere il messaggio ed elaborarlo, queste parti estenderanno la classe `BroadcastReceiver` offerta dal SDK Android.



**Figura 191:** Diagramma di sequenza - Elaborazione beacon rilevati e comunicazione broadcast

### 6.3 Avvio navigazione

Il diagramma in figura 192 rappresenta il flusso d'eventi generato nelle classi del model qualora si richiedesse l'avvio della navigazione. La richiesta parte da `NavigationManagerPresenter` con la chiamata del metodo `startNavigation()` sull'oggetto `NavigationManagerImp` passando come parametri la destinazione identificata dall'oggetto di tipo `PointOfInterest`. Il `NavigatorManagerImp` si occupa quindi di impostare il grafo all'oggetto di tipo `NavigatorImp` con il metodo `setGraph()` dopodiché invoca il metodo `calculatePath()` in cui è calcolato il percorso da seguire durante la navigazione attraverso l'oggetto `DijkstraPathFinder` che restituisce una `List` di `EnrichedEdge` salvata in `navigator` in un campo dati. A questo punto `navigator` è pronto per restituire le informazioni (`ProcessedInformation`) richieste dalla classe `NavigationManagerImp`, quest'ultimo invoca il metodo `toNextRegion()` passando come parametri la lista di beacon, rilevati e ricevuti tramite l'oggetto `BroadcastReceiver`. `navigator` ricava dai beacon, rilevati il beacon, il cui segnale risulta essere il più potente (`getMostPowerfulBEacon()`), quindi controlla che il beacon ritenuto più vicino all'utente appartiene alla region of interest (ROI<sub>g</sub>) del percorso previsto, infine costruisce le `ProcessedInformation` richieste grazie all'oggetto `Edge` identificato come prossimo tratto di percorso da percorrere. Le `ProcessedInformation` vengono quindi ritornate a `NavigationManagerImp` che le restituisce a `NavigationManagerImp` il quale le scompatterà e le restituirà alla view e quindi all'utente.

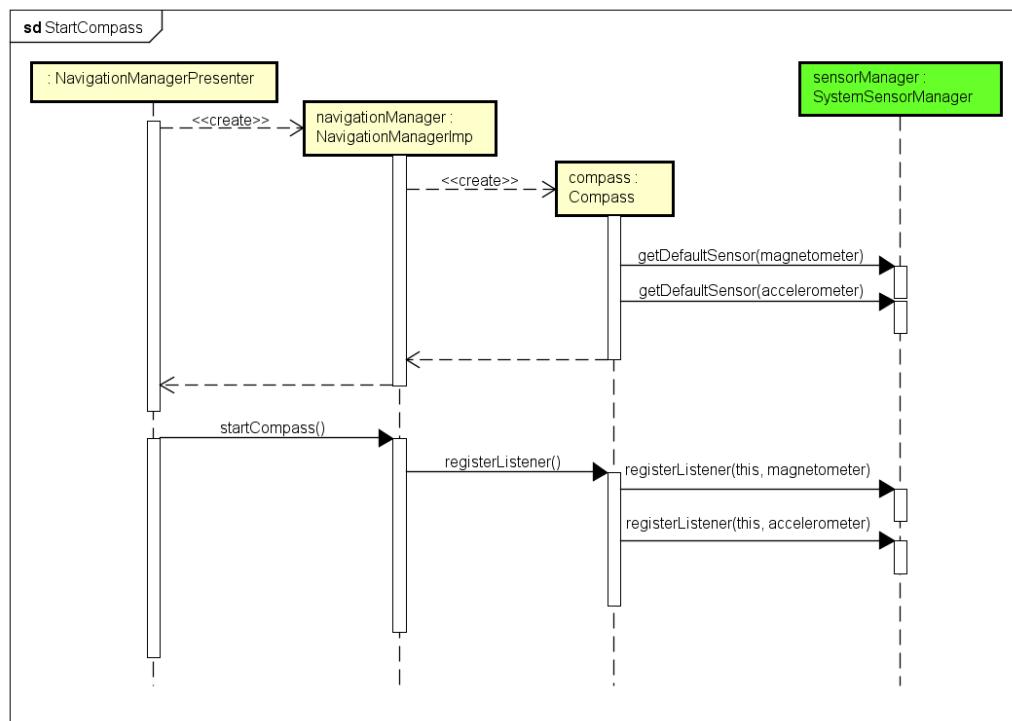


**Figura 192:** Diagramma di sequenza - Avvio navigazione

## 6.4 Avvio della bussola

Il diagramma in figura 193 rappresenta il flusso generato dall'oggetto della classe `NavigationManagerPresenter`, esso effettua due operazioni principali:

- Creazione di `NavigationManager` che causa la creazione dell'oggetto `Compass` a cui viene passato come parametro del costruttore il riferimento a `SensorManager`, classe della libreria Android che permette di recuperare i riferimenti ai sensori del device attraverso la chiamata del metodo `getDefaultSensor(typeSensor)`. `Compass` per calcolare l'orientamento del device necessita dei dati provenienti dal magnetometro e accellerometro.
- `startCompass()` invece accende la bussola `Compass` attraverso la classe `NavigationManager` il quale chiama il metodo `registerListener()`, tale metodo tramite il riferimento a `sensorManager` chiama il metodo `registerListener()` e imposta l'oggetto `compass` observer dei sensori.



**Figura 193:** Diagramma di sequenza - Avvio della bussola

## 7 Tracciamento

### 7.1 Tracciamento Classi-Requisiti

Classe	Requisiti
model::InformationManagerImp	RDesF12.3.5 RObbF12.3 RObbF7.1 RObbF9 ROpzF9.2.2
model::MessageSendType	RDesF12.3.5 RDesF7.4.2.3 RObbF12.3 RObbF7 RObbF7.1 RObbF7.4 RObbF7.4.2 RObbF7.5 RObbF9 ROpzF9.2.2
model::NavigationManagerImp	RDesF7.4.2.3 RObbF7 RObbF7.4 RObbF7.4.2 RObbF7.5 ROpzF7.4.2.6
model::NoBeaconSeenException	ROpzF7.4.2.6
model::ServiceConnectionImp	RDesF12.3.5 RDesF7.4.2.3 RObbF12.3 RObbF7 RObbF7.1 RObbF7.4 RObbF7.4.2 RObbF7.5 RObbF9 ROpzF9.2.2

Classe	Requisiti
model::beacon::BeaconManagerAdapter	RDesF7.4.2.3 RObbF7 RObbF8 RObbF8.1 RObbF8.1.1 RObbF8.2 RObbF8.3 RObbF8.4 RObbF8.5 RObbF8.6 RObbF9.10 RObbF9.3 RObbF9.4 RObbF9.5 RObbF9.6 RObbF9.9 ROpzF7.4.2.5 ROpzF7.4.2.6 ROpzF9.2.2
model::beacon::LocalBinder	RDesF12.3.5 RDesF7.4.2.3 RObbF12.3 RObbF7 RObbF7.1 RObbF7.4 RObbF7.4.2 RObbF7.5 RObbF9 ROpzF9.2.2
model::beacon::LoggerImp	RDesF12.3.1 RDesF12.3.2 RDesF12.3.3 RDesF12.3.5 RObbF12.3

Classe	Requisiti
model::beacon::MyBeaconImp	RDesF7.4.2.3 RObbF8 RObbF8.1.1 RObbF8.1.2 RObbF8.2 RObbF8.3 RObbF8.4 RObbF8.5 RObbF8.6
model::beacon::MyDistanceCalculator	RObbF8.4
model::compass::Compass	RObbF7.4.2
model::dataaccess::dao::BuildingContract	RDesF10.2.1
model::dataaccess::dao::BuildingTable	RDesF10.2 RObbF7.7
model::dataaccess::dao::CategoryContract	RDesF10.2.1
model::dataaccess::dao::CategoryTable	RDesF10.2
model::dataaccess::dao::DaoFactoryHelper	RDesF10.2 RDesF10.2.1 RDesF10.2.2
model::dataaccess::dao::EdgeContract	RDesF10.2.1
model::dataaccess::dao::EdgeTable	RDesF10.2
model::dataaccess::dao::EdgeTypeContract	RDesF10.2.1
model::dataaccess::dao::EdgeTypeTable	RDesF10.2
model::dataaccess::dao::MapsDbContract	RDesF10.2.1
model::dataaccess::dao::MapsDbHelper	RDesF10.2.1
model::dataaccess::dao::PhotoContract	RDesF10.2.1
model::dataaccess::dao::PhotoTable	RDesF10.2
model::dataaccess::dao::PointOfInterestContract	RDesF10.2.1

Classe	Requisiti
model::dataaccess::dao::PointOfInterestTable	RDesF10.2
model::dataaccess::dao::RegionOfInterestContract	RDesF10.2.1
model::dataaccess::dao::RegionOfInterestTable	RDesF10.2
model::dataaccess::dao::RemoteBuildingDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.1 RDesF10.2.2.2 RDesF10.2.2.3 RDesF10.2.2.3.1 RDesF10.2.2.3.2 RDesF10.2.2.3.3 RDesF10.2.2.3.4 RDesF10.2.2.3.5
model::dataaccess::dao::RemoteCategoryDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.2
model::dataaccess::dao::RemoteDaoFactory	RDesF10.2.2
model::dataaccess::dao::RemoteEdgeDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.2
model::dataaccess::dao::RemoteEdgeTypeDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.2
model::dataaccess::dao::RemotePhotoDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.2
model::dataaccess::dao::RemotePointOfInterestDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.2
model::dataaccess::dao::RemoteRegionOfInterestDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.2

Classe	Requisiti
model::dataaccess::dao::RemoteRoiPoiDao	RDesF10.2.1.2 RDesF10.2.2 RDesF10.2.2.2
model::dataaccess::dao::RoiPoiContract	RDesF10.2.1
model::dataaccess::dao::RoiPoiTable	RDesF10.2
model::dataaccess::dao::SQLDao	RDesF10.2.1
model::dataaccess::dao::SQLiteBuildingDao	RDesF10.2.1 RDesF10.2.1.1 RDesF10.2.1.2 RDesF10.2.1.3 RDesF10.2.1.4 RDesF10.2.1.4.1 RDesF10.2.1.4.2 RDesF10.2.1.4.3 RDesF10.2.1.4.4 RDesF10.2.1.4.5 RDesF10.2.2.2 RObbF7.7 RObbF7.8
model::dataaccess::dao::SQLiteCategoryDao	RDesF10.2.1 RDesF10.2.1.2 RDesF10.2.1.4 RDesF10.2.2.2
model::dataaccess::dao::SQLiteDaoFactory	RDesF10.2.1
model::dataaccess::dao::SQLiteEdgeDao	RDesF10.2.1 RDesF10.2.1.2 RDesF10.2.1.4 RDesF10.2.2.2
model::dataaccess::dao::SQLiteEdgeTypeDao	RDesF10.2.1 RDesF10.2.1.2 RDesF10.2.1.4 RDesF10.2.2.2

Classe	Requisiti
model::dataaccess::dao::SQLitePhotoDao	RDesF10.2.1 RDesF10.2.1.2 RDesF10.2.1.4 RDesF10.2.2.2
model::dataaccess::dao::SQLitePointOfInterestDao	RDesF10.2.1 RDesF10.2.1.2 RDesF10.2.1.4 <u>RDesF10.2.2.2</u>
model::dataaccess::dao::SQLiteRegionOfInterestDao	RDesF10.2.1 RDesF10.2.1.2 RDesF10.2.1.4 <u>RDesF10.2.2.2</u>
model::dataaccess::dao::SQLiteRoiPoiDao	RDesF10.2.1 RDesF10.2.1.2 RDesF10.2.1.4 <u>RDesF10.2.2.2</u>

Classe	Requisiti
model::dataaccess::service::BuildingService	RDesF10.2 RDesF10.2.1 RDesF10.2.1.1 RDesF10.2.1.2 RDesF10.2.1.3 RDesF10.2.1.4 RDesF10.2.1.4.1 RDesF10.2.1.4.2 RDesF10.2.1.4.3 RDesF10.2.1.4.4 RDesF10.2.1.4.5 RDesF10.2.2 RDesF10.2.2.1 RDesF10.2.2.2 RDesF10.2.2.3 RDesF10.2.2.3.1 RDesF10.2.2.3.2 RDesF10.2.2.3.3 RDesF10.2.2.3.4 RDesF10.2.2.3.5 RDesF10.2.2.4 RDesF10.2.3 RObbF7.7 RObbF9.10
model::dataaccess::service::EdgeService	RDesF10.2 RDesF10.2.1 RDesF10.2.1.4 RDesF10.2.2 RDesF10.2.2.2
model::dataaccess::service::PhotoService	RDesF10.2 RDesF10.2.1 RDesF10.2.1.4 RDesF10.2.2 RDesF10.2.2.2

Classe	Requisiti
model::dataaccess::service::PointOfInterestService	RDesF10.2 RDesF10.2.1 RDesF10.2.1.4 RDesF10.2.2 <u>RDesF10.2.2.2</u>
model::dataaccess::service::RegionOfInterestService	RDesF10.2 RDesF10.2.1 RDesF10.2.1.4 RDesF10.2.2 <u>RDesF10.2.2.2</u>
model::dataaccess::service::ServiceHelper	RDesF10.2 RDesF10.2.1 <u>RDesF10.2.2</u>
model::navigator::BuildingInformation	RObbF7.1 RObbF9 RObbF9.10 RObbF9.3 RObbF9.4 RObbF9.5 RObbF9.6 RObbF9.9 ROpzF9.2
model::navigator::BuildingMapImp	RObbF7.1 RObbF9 RObbF9.1 ROpzF9.2 ROpzF9.2.2

Classe	Requisiti
model::navigator::NavigatorImp	RDesF7.1.1 RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2 RDesF7.3.1.3 RDesF7.4.2.2 RDesF7.4.2.3 RDesF7.4.3.1 RObbF7 RObbF7.1.2 RObbF7.3 RObbF7.4 RObbF7.4.2 RObbF7.4.2.1 RObbF7.5 RObbF8.1.1 ROpzF7.4.2.4 ROpzF7.4.2.5 ROpzF7.4.3.2 ROpzF7.4.3.3
model::navigator::NoGraphSetException	RObbF7
model::navigator::NoNavigationInformationException	RObbF7
model::navigator::PathException	RDesF7.4.2.3
model::navigator::ProcessedInformationImp	RObbF10.1.2.1 RObbF7.4.2 ROpzF7.4.2.4
model::navigator::algorithm::DijkstraPathFinder	RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2 RDesF7.3.1.3 RObbF7.3
model::navigator::graph::MapGraph	RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2 RObbF7.3

Classe	Requisiti
model::navigator::graph::area::PointOfInterestImp	RDesF7.1.1 RObbF7.1 RObbF7.1.2 RObbF8.1.1 RObbF8.1.2 RObbF9.2.3 RObbF9.2.4 ROpzF9.2 ROpzF9.2.1
model::navigator::graph::area::PointOfInterestInformation	RObbF9.2.3 RObbF9.2.4 ROpzF9.2 ROpzF9.2.1
model::navigator::graph::area::RegionOfInterestImp	RDesF7.3.1 RDesF7.3.1.1 RObbF7.3
model::navigator::graph::edge::DefaultEdge	RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2
model::navigator::graph::edge::ElevatorEdge	RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2
model::navigator::graph::edge::StairEdge	RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2
model::navigator::graph::navigationinformation::BasicInformation	RObbF10.1.2.1 RObbF7.4.2 RObbF7.4.2.1
model::navigator::graph::navigationinformation::DetailedInformation	RDesF7.4.3 RObbF10.1.2.1 RObbF7.4.2 ROpzF7.4.3.2

Classe	Requisiti
model::navigator::graph::-navigationinformation::NavigationInformationImp	RDesF7.1.1 RObbF7 RObbF7.4.2 RObbF7.4.2.1
model::navigator::graph::-navigationinformation::PhotoInformation	RDesF7.4.3.1 RObbF7.4.2
model::navigator::graph::-navigationinformation::PhotoRef	RDesF7.4.3.1
model::navigator::graph::vertex::VertexImp	RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2 RObbF7.3
model::usersetting::DeveloperCodeManager	RObbF10.3
model::usersetting::InstructionPreference	RDesF10.1.2
model::usersetting::PathPreference	RDesF7.3.1 ROpzF10.1
model::usersetting::SettingImp	RDesF10.1.2 RDesF10.1.2.2 RDesF10.1.2.3 RDesF10.1.2.4 RDesF10.1.2.5 RDesF7.3.1 RDesF7.3.1.1 RDesF7.3.1.2 RObbF10.1.1.3 RObbF10.3 ROpzF10 ROpzF10.1 ROpzF10.1.1 ROpzF10.1.1.1 ROpzF10.1.1.2
presenter::Checker	RObbF7.4.1
presenter::DatabaseServiceManager	RObbF9

Classe	Requisiti
presenter::DetailedInformationActivity	RDesF7.4.3.4 ROpzF7.4.3.2
presenter::DeveloperUnlockerActivity	RObbF10.3.3
presenter::HelpActivity	ROpzF11
presenter::HomeActivity	RObbF7.1.2 RObbF7.1.2.1 RObbF7.1.3 RObbF7.1.4 RObbF7.2 RObbF7.4.1 RObbF7.4.1.1 RObbF7.4.1.2 RObbF7.4.1.3 RObbF7.4.4 RObbF7.8 RObbF9.10 RObbF9.9
presenter::ImageAdapter	RDesF7.4.3.1
presenter::ImageDetailActivity	RDesF7.4.3.1
presenter::ImageListFragment	RDesF7.4.3.1
presenter::ImagePageAdapter	RDesF7.4.3.1
presenter::InformationManagerPresenter	RObbF9
presenter::LocalMapActivity	RDesF10.2.1.4.1 RDesF10.2.1.4.2 RDesF10.2.1.4.3 RDesF10.2.1.4.4 RDesF10.2.1.4.5 RDesF10.2.2.4
presenter::LoggingActivity	RObbF12.6 RObbF12.8 RObbF12.9
presenter::LogInformationActivity	RDesF12.3.4

Classe	Requisiti
presenter::MainActivity	RObbF7.4.1
presenter::MainDeveloperActivity	RObbF12.3
presenter::MainDeveloperPresenter	RObbF12.3
presenter::MapDownloaderActivity	RDesF10.2.3
presenter::NavigationActivity	RDesF7.4.2.3 RObbF7.4.1 RObbF7.4.1.1 RObbF7.4.1.2 RObbF7.4.1.3 RObbF7.4.2.1 RObbF7.6 RObbF7.7 ROpzF7.4.2.5 ROpzF7.4.2.6 ROpzF7.4.3.3
presenter::NavigationAdapter	RObbF7.4.2.1
presenter::NavigationManagerPresenter	RObbF7.4.2.1
presenter::NearbyPoiActivity	RObbF12.7
presenter::PoiCategoryActivity	RObbF7.1 RObbF7.1.2 RObbF7.2 RObbF7.4.4
presenter::PreferencesActivity	ROpzF10.1.1.1
presenter::RemoteMapManagerActivity	RDesF10.2.2.1 RDesF10.2.2.3 RDesF10.2.2.3.1 RDesF10.2.2.3.2 RDesF10.2.2.3.3 RDesF10.2.2.3.4 RDesF10.2.2.3.5 RDesF10.2.2.4 RDesF10.2.3

Classe	Requisiti
presenter::SearchSuggestionsProvider	RDesF7.1.1 RDesF7.1.1.1
presenter::SettingManager	ROpzF10.1.1.1
view::DetailedInformationViewImp	RDesF7.4.3.4 ROpzF7.4.3.2
view::DeveloperUnlockerViewImp	RObbF10.3.2 RObbF10.3.3
view::HelpViewImp	ROpzF11
view::HomeViewImp	RDesF7.1.1 RDesF7.1.1.1 RObbF7.1 RObbF7.1.2 RObbF7.1.2.1 RObbF7.1.3 RObbF7.1.4 RObbF7.2 RObbF7.4.1 RObbF7.4.1.1 RObbF7.4.1.2 RObbF7.4.1.3 RObbF7.4.4 RObbF7.8 RObbF9.10
view::ImageDetailViewImp	RDesF7.4.3.1
view::ImageListFragmentViewImp	RDesF7.4.3.1
view::LocalMapManagerViewImp	RDesF10.2.1.4.1 RDesF10.2.1.4.2 RDesF10.2.1.4.3 RDesF10.2.1.4.4 RDesF10.2.1.4.5 RDesF10.2.2.4

Classe	Requisiti
view::LoggingViewImp	RObbF12 RObbF12.1 RObbF12.2 RObbF12.5 RObbF12.6 RObbF12.8 RObbF12.9 ROpzF12.4
view::LogInformationViewImp	RDesF12.3.4
view::MainDeveloperViewImp	RObbF12.3
view::MainViewImp	RObbF7.4.1
view::MapDownloaderViewImp	RDesF10.2.3
view::NavigationViewImp	RDesF7.4.2.3 RObbF7.4.1 RObbF7.4.1.1 RObbF7.4.1.2 RObbF7.4.1.3 RObbF7.4.2.1 RObbF7.6 RObbF7.7 ROpzF7.4.2.5 ROpzF7.4.2.6 ROpzF7.4.3.3
view::NearbyPoiViewImp	RObbF12.7
view::PoiCategoryViewImp	RObbF7.1 RObbF7.1.2 RObbF7.2 RObbF7.4.4
view::PreferencesViewImp	ROpzF10.1.1.1

Classe	Requisiti
view::RemoteMapViewManagerViewImp	RDesF10.2.2.1
	RDesF10.2.2.1.1
	RDesF10.2.2.3
	RDesF10.2.2.3.1
	RDesF10.2.2.3.2
	RDesF10.2.2.3.3
	RDesF10.2.2.3.4
	RDesF10.2.2.3.5
	RDesF10.2.2.4
	RDesF10.2.3

**Tabella 1:** Tabella classi / requisiti

## 7.2 Requisiti-Classi

Requisito	Classi
RObbV1	
RObbQ2	
RObbQ2.1	
RObbQ2.2	
RObbQ2.3	
RObbQ2.4	
RObbQ2.5	
RObbQ2.6	
RObbV3	
RObbV3.1	
RObbV3.2	
RObbV3.3	
RObbV3.3.1	

Requisito	Classi
ROpzV3.3.1.1	
RObbV3.3.1.2	
ROpzV3.3.1.3	
ROpzV3.3.1.4	
RObbV4	
RObbV4.1	
RObbV4.1.1	
RObbV4.1.2	
RObbV4.2	
RObbV4.2.1	
RObbV4.3	
RObbV4.3.1	
RObbV4.3.2	
RObbV4.3.3	
RObbV4.4	
RObbV4.5	
RObbV4.6	
RObbV5	
RObbV5.1	
RObbV5.1.1	
RObbV5.1.2	
RObbV5.1.3	
RObbV5.2	

Requisito	Classi
RObbV5.2.1	
RObbV5.2.2	
RObbV5.3	
RObbV5.3.1	
RObbV5.3.2	
RObbV5.3.3	
RObbV5.4	
RObbV5.4.1	
RObbV5.4.2	
RObbV5.4.3	
RObbV5.4.4	
RObbV5.5	
RObbV6	
RObbF7	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::BeaconManagerAdapter model::beacon::LocalBinder model::navigator::NavigatorImp model::navigator::NoGraphSetException model::navigator::NoNavigationInformationException model::navigator::graph::-navigationinformation::NavigationInformationImp

---

Requisito	Classi
RObbF7.1	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::LocalBinder model::navigator::BuildingInformation model::navigator::BuildingMapImp model::navigator::graph::area::PointOfInterestImp presenter::PoiCategoryActivity view::HomeViewImp view::PoiCategoryViewImp
RDesF7.1.1	model::navigator::NavigatorImp model::navigator::graph::area::PointOfInterestImp model::navigator::graph::-navigationinformation::NavigationInformationImp presenter::SearchSuggestionsProvider view::HomeViewImp
RDesF7.1.1.1	presenter::SearchSuggestionsProvider view::HomeViewImp
RObbF7.1.2	model::navigator::NavigatorImp model::navigator::graph::area::PointOfInterestImp presenter::HomeActivity presenter::PoiCategoryActivity view::HomeViewImp view::PoiCategoryViewImp
RObbF7.1.2.1	presenter::HomeActivity view::HomeViewImp
RObbF7.1.3	presenter::HomeActivity view::HomeViewImp
RObbF7.1.4	presenter::HomeActivity view::HomeViewImp
RObbF7.2	presenter::HomeActivity presenter::PoiCategoryActivity view::HomeViewImp view::PoiCategoryViewImp

Requisito	Classi
RObbF7.3	model::navigator::NavigatorImp model::navigator::algorithm::DijkstraPathFinder model::navigator::graph::MapGraph model::navigator::graph::area::RegionOfInterestImp model::navigator::graph::vertex::VertexImp
RDesF7.3.1	model::navigator::NavigatorImp model::navigator::algorithm::DijkstraPathFinder model::navigator::graph::MapGraph model::navigator::graph::area::RegionOfInterestImp model::navigator::graph::edge::DefaultEdge model::navigator::graph::edge::ElevatorEdge model::navigator::graph::edge::StairEdge model::navigator::graph::vertex::VertexImp model::usersetting::PathPreference model::usersetting::SettingImp
RDesF7.3.1.1	model::navigator::NavigatorImp model::navigator::algorithm::DijkstraPathFinder model::navigator::graph::MapGraph model::navigator::graph::area::RegionOfInterestImp model::navigator::graph::edge::DefaultEdge model::navigator::graph::edge::ElevatorEdge model::navigator::graph::edge::StairEdge model::navigator::graph::vertex::VertexImp model::usersetting::SettingImp
RDesF7.3.1.2	model::navigator::NavigatorImp model::navigator::algorithm::DijkstraPathFinder model::navigator::graph::MapGraph model::navigator::graph::edge::DefaultEdge model::navigator::graph::edge::ElevatorEdge model::navigator::graph::edge::StairEdge model::navigator::graph::vertex::VertexImp model::usersetting::SettingImp
RDesF7.3.1.3	model::navigator::NavigatorImp model::navigator::algorithm::DijkstraPathFinder

---

Requisito	Classi
RObbF7.4	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::LocalBinder model::navigator::NavigatorImp
RObbF7.4.1	presenter::Checker presenter::HomeActivity presenter::MainActivity presenter::NavigationActivity view::HomeViewImp view::MainViewImp view::NavigationViewImp
RObbF7.4.1.1	presenter::HomeActivity presenter::NavigationActivity view::HomeViewImp view::NavigationViewImp
RObbF7.4.1.2	presenter::HomeActivity presenter::NavigationActivity view::HomeViewImp view::NavigationViewImp
RObbF7.4.1.3	presenter::HomeActivity presenter::NavigationActivity view::HomeViewImp view::NavigationViewImp

Requisito	Classi
RObbF7.4.2	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::LocalBinder model::compass::Compass model::navigator::NavigatorImp model::navigator::ProcessedInformationImp model::navigator::graph::-navigationinformation::BasicInformation model::navigator::graph::-navigationinformation::DetailedInformation model::navigator::graph::-navigationinformation::NavigationInformationImp model::navigator::graph::-navigationinformation::PhotoInformation
RObbF7.4.2.1	model::navigator::NavigatorImp model::navigator::graph::-navigationinformation::BasicInformation model::navigator::graph::-navigationinformation::NavigationInformationImp presenter::NavigationActivity presenter::NavigationAdapter presenter::NavigationManagerPresenter view::NavigationViewImp
RDesF7.4.2.2	model::navigator::NavigatorImp model::navigator::graph::-navigationinformation::NavigationInformation
RDesF7.4.2.3	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::BeaconManagerAdapter model::beacon::LocalBinder model::beacon::MyBeaconImp model::navigator::NavigatorImp model::navigator::PathException presenter::NavigationActivity view::NavigationViewImp

---

Requisito	Classi
ROpzF7.4.2.4	model::navigator::NavigatorImp model::navigator::ProcessedInformationImp
ROpzF7.4.2.5	model::beacon::BeaconManagerAdapter model::navigator::NavigatorImp presenter::NavigationActivity view::NavigationViewImp
ROpzF7.4.2.6	model::NavigationManagerImp model::NoBeaconSeenException model::beacon::BeaconManagerAdapter presenter::NavigationActivity view::NavigationViewImp
RDesF7.4.3	model::navigator::Navigator model::navigator::graph::-navigationinformation::DetailedInformation
RDesF7.4.3.1	model::navigator::NavigatorImp model::navigator::graph::-navigationinformation::PhotoInformation model::navigator::graph::-navigationinformation::PhotoRef presenter::ImageAdapter presenter::ImageDetailActivity presenter::ImageListFragment presenter::ImagePageAdapter view::ImageDetailViewImp view::ImageListFragmentViewImp
ROpzF7.4.3.2	model::navigator::NavigatorImp model::navigator::graph::-navigationinformation::DetailedInformation presenter::DetailedInformationActivity view::DetailedInformationViewImp
ROpzF7.4.3.3	model::navigator::NavigatorImp presenter::NavigationActivity view::NavigationViewImp
RDesF7.4.3.4	presenter::DetailedInformationActivity view::DetailedInformationViewImp

Requisito	Classi
RObbF7.4.4	presenter::HomeActivity presenter::PoiCategoryActivity view::HomeViewImp view::PoiCategoryViewImp
RObbF7.5	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::LocalBinder model::navigator::NavigatorImp
RObbF7.6	presenter::NavigationActivity view::NavigationViewImp
RObbF7.7	model::dataaccess::dao::BuildingTable model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::NavigationActivity view::NavigationViewImp
RObbF7.8	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::DatabaseService presenter::HomeActivity view::HomeViewImp
RObbF8	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF8.1	model::beacon::BeaconManagerAdapter
RObbF8.1.1	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp model::navigator::NavigatorImp model::navigator::graph::area::PointOfInterestImp
RObbF8.1.2	model::beacon::MyBeaconImp model::navigator::Navigator model::navigator::graph::area::PointOfInterestImp
RObbF8.2	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp

Requisito	Classi
RObbF8.3	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF8.4	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp model::beacon::MyDistanceCalculator
RObbF8.5	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF8.6	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF9	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::LocalBinder model::navigator::BuildingInformation model::navigator::BuildingMapImp presenter::DatabaseServiceManager presenter::InformationManagerPresenter
RObbF9.1	model::navigator::BuildingMapImp
ROpzF9.2	model::navigator::BuildingInformation model::navigator::BuildingMapImp model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation
ROpzF9.2.1	model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation
ROpzF9.2.2	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::BeaconManagerAdapter model::beacon::LocalBinder model::navigator::BuildingMapImp
RObbF9.2.3	model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation
RObbF9.2.4	model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation

Requisito	Classi
RObbF9.3	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbF9.4	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbF9.5	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbF9.6	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbQ9.7	
RObbQ9.8	
RObbF9.9	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation presenter::HomeActivity
RObbF9.10	model::beacon::BeaconManagerAdapter model::dataaccess::service::BuildingService model::navigator::BuildingInformation presenter::HomeActivity view::HomeViewImp
ROpzF10	model::usersetting::SettingImp
ROpzF10.1	model::usersetting::PathPreference model::usersetting::SettingImp
ROpzF10.1.1	model::usersetting::SettingImp
ROpzF10.1.1.1	model::usersetting::SettingImp presenter::PreferencesActivity presenter::SettingManager view::PreferencesViewImp
ROpzF10.1.1.2	model::usersetting::SettingImp
RObbF10.1.1.3	model::usersetting::SettingImp
RDesF10.1.2	model::usersetting::InstructionPreference model::usersetting::SettingImp

Requisito	Classi
RObbF10.1.2.1	model::navigator::ProcessedInformationImp model::navigator::graph::-navigationinformation::BasicInformation model::navigator::graph::-navigationinformation::DetailedInformation model::navigator::graph::-navigationinformation::NavigationInformation
RDesF10.1.2.2	model::usersetting::SettingImp
RDesF10.1.2.3	model::usersetting::SettingImp
RDesF10.1.2.4	model::usersetting::SettingImp
RDesF10.1.2.5	model::usersetting::SettingImp
RDesF10.2	model::dataaccess::dao::BuildingTable model::dataaccess::dao::CategoryTable model::dataaccess::dao::DaoFactoryHelper model::dataaccess::dao::EdgeTable model::dataaccess::dao::EdgeTypeTable model::dataaccess::dao::PhotoTable model::dataaccess::dao::PointOfInterestTable model::dataaccess::dao::RegionOfInterestTable model::dataaccess::dao::RoiPoiTable model::dataaccess::service::BuildingService model::dataaccess::service::EdgeService model::dataaccess::service::PhotoService model::dataaccess::service::PointOfInterestService model::dataaccess::service::RegionOfInterestService model::dataaccess::service::ServiceHelper

---

Requisito	Classi
RDesF10.2.1	model::dataaccess::dao::BuildingContract model::dataaccess::dao::CategoryContract model::dataaccess::dao::DaoFactoryHelper model::dataaccess::dao::EdgeContract model::dataaccess::dao::EdgeTypeContract model::dataaccess::dao::MapsDbContract model::dataaccess::dao::MapsDbHelper model::dataaccess::dao::PhotoContract model::dataaccess::dao::PointOfInterestContract model::dataaccess::dao::RegionOfInterestContract model::dataaccess::dao::RoiPoiContract model::dataaccess::dao::SQLDao model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::dao::SQLiteCategoryDao model::dataaccess::dao::SQLiteDaoFactory model::dataaccess::dao::SQLiteEdgeDao model::dataaccess::dao::SQLiteEdgeTypeDao model::dataaccess::dao::SQLitePhotoDao model::dataaccess::dao::SQLitePointOfInterestDao model::dataaccess::dao::SQLiteRegionOfInterestDao model::dataaccess::dao::SQLiteRoiPoiDao model::dataaccess::service::BuildingService model::dataaccess::service::EdgeService model::dataaccess::service::PhotoService model::dataaccess::service::PointOfInterestService model::dataaccess::service::RegionOfInterestService model::dataaccess::service::ServiceHelper
RDesF10.2.1.1	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService

Requisito	Classi
RDesF10.2.1.2	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::dao::RemoteCategoryDao model::dataaccess::dao::RemoteEdgeDao model::dataaccess::dao::RemoteEdgeTypeDao model::dataaccess::dao::RemotePhotoDao model::dataaccess::dao::RemotePointOfInterestDao model::dataaccess::dao::RemoteRegionOfInterestDao model::dataaccess::dao::RemoteRoiPoiDao model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::dao::SQLiteCategoryDao model::dataaccess::dao::SQLiteEdgeDao model::dataaccess::dao::SQLiteEdgeTypeDao model::dataaccess::dao::SQLitePhotoDao model::dataaccess::dao::SQLitePointOfInterestDao model::dataaccess::dao::SQLiteRegionOfInterestDao model::dataaccess::dao::SQLiteRoiPoiDao model::dataaccess::service::BuildingService
RDesF10.2.1.3	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService
RDesF10.2.1.4	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::dao::SQLiteCategoryDao model::dataaccess::dao::SQLiteEdgeDao model::dataaccess::dao::SQLiteEdgeTypeDao model::dataaccess::dao::SQLitePhotoDao model::dataaccess::dao::SQLitePointOfInterestDao model::dataaccess::dao::SQLiteRegionOfInterestDao model::dataaccess::dao::SQLiteRoiPoiDao model::dataaccess::service::BuildingService model::dataaccess::service::EdgeService model::dataaccess::service::PhotoService model::dataaccess::service::PointOfInterestService model::dataaccess::service::RegionOfInterestService
RDesF10.2.1.4.1	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapViewImp

---

Requisito	Classi
RDesF10.2.1.4.2	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF10.2.1.4.3	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF10.2.1.4.4	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF10.2.1.4.5	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF10.2.2	model::dataaccess::dao::DaoFactoryHelper model::dataaccess::dao::RemoteBuildingDao model::dataaccess::dao::RemoteCategoryDao model::dataaccess::dao::RemoteDaoFactory model::dataaccess::dao::RemoteEdgeDao model::dataaccess::dao::RemoteEdgeTypeDao model::dataaccess::dao::RemotePhotoDao model::dataaccess::dao::RemotePointOfInterestDao model::dataaccess::dao::RemoteRegionOfInterestDao model::dataaccess::dao::RemoteRoiPoiDao model::dataaccess::service::BuildingService model::dataaccess::service::EdgeService model::dataaccess::service::PhotoService model::dataaccess::service::PointOfInterestService model::dataaccess::service::RegionOfInterestService model::dataaccess::service::ServiceHelper
RDesF10.2.2.1	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp

---

Requisito	Classi
RDesF10.2.2.1.1	view::RemoteMapManagerViewImp
RDesF10.2.2.2	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::dao::RemoteCategoryDao model::dataaccess::dao::RemoteEdgeDao model::dataaccess::dao::RemoteEdgeTypeDao model::dataaccess::dao::RemotePhotoDao model::dataaccess::dao::RemotePointOfInterestDao model::dataaccess::dao::RemoteRegionOfInterestDao model::dataaccess::dao::RemoteRoiPoiDao model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::dao::SQLiteCategoryDao model::dataaccess::dao::SQLiteEdgeDao model::dataaccess::dao::SQLiteEdgeTypeDao model::dataaccess::dao::SQLitePhotoDao model::dataaccess::dao::SQLitePointOfInterestDao model::dataaccess::dao::SQLiteRegionOfInterestDao model::dataaccess::dao::SQLiteRoiPoiDao model::dataaccess::service::BuildingService model::dataaccess::service::EdgeService model::dataaccess::service::PhotoService model::dataaccess::service::PointOfInterestService model::dataaccess::service::RegionOfInterestService
RDesF10.2.2.3	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF10.2.2.3.1	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF10.2.2.3.2	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp

Requisito	Classi
RDesF10.2.2.3.3	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF10.2.2.3.4	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF10.2.2.3.5	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF10.2.2.4	model::dataaccess::service::BuildingService presenter::LocalMapActivity presenter::RemoteMapManagerActivity view::LocalMapManagerViewImp view::RemoteMapManagerViewImp
RDesF10.2.3	model::dataaccess::service::BuildingService presenter::MapDownloaderActivity presenter::RemoteMapManagerActivity view::MapDownloaderViewImp view::RemoteMapManagerViewImp
RObbF10.3	model::usersetting::DeveloperCodeManager model::usersetting::SettingImp
RObbF10.3.1	view::DeveloperUnlockerView
RObbF10.3.2	view::DeveloperUnlockerViewImp
RObbF10.3.3	presenter::DeveloperUnlockerActivity view::DeveloperUnlockerViewImp
ROpzF11	presenter::HelpActivity view::HelpViewImp
RObbF12	view::LoggingViewImp
RObbF12.1	view::LoggingViewImp

Requisito	Classi
RObbF12.2	view::LoggingViewImp
RObbF12.3	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::LocalBinder model::beacon::LoggerImp presenter::MainDeveloperActivity presenter::MainDeveloperPresenter view::MainDeveloperViewImp
RDesF12.3.1	model::InformationManager model::beacon::LoggerImp
RDesF12.3.2	model::InformationManager model::beacon::LoggerImp
RDesF12.3.3	model::InformationManager model::beacon::LoggerImp
RDesF12.3.4	presenter::LogInformationActivity view::LogInformationViewImp
RDesF12.3.5	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::LocalBinder model::beacon::LoggerImp
ROpzF12.4	view::LoggingViewImp
RObbF12.5	view::LoggingViewImp
RObbF12.6	presenter::LoggingActivity view::LoggingViewImp
RObbF12.7	presenter::NearbyPoiActivity view::NearbyPoiViewImp
RObbF12.8	presenter::LoggingActivity view::LoggingViewImp
RObbF12.9	presenter::LoggingActivity view::LoggingViewImp

Requisito	Classi
ROpzQ13	
RDesP14	
ROpzP15	

**Tabella 2:** Tabella requisiti / classi

### 7.3 Tracciamento Metodi - test di unità

Metodo	Test
model::InformationManagerImp::getAllVisibleBeacons()	TU41
model::InformationManagerImp::getBuildingMap()	TU41
model::InformationManagerImp::getDatabaseService()	TU41
model::InformationManagerImp::getNearbyPOIs()	TU41
model::InformationManagerImp::- saveRecordedBeaconInformation()	TU43
model::InformationManagerImp::startRecordingBeacons()	TU43
model::NavigationManagerImp::addBeaconListener()	TU42
model::NavigationManagerImp::- getAllNavigationInstruction()	TU44
model::NavigationManagerImp::getNextInstruction()	TU44
model::NavigationManagerImp::removeBeaconListener()	TU42
model::NavigationManagerImp::startNavigation()	TU44
model::NavigationManagerImp::stopNavigation()	TU44
model::beacon::BeaconManagerAdapter::modifyScanPeriod()	TU40
model::beacon::BeaconManagerAdapter::- setBackgroundMode()	TU39
model::beacon::Logger::add()	TU37

Metodo	Test
model::beacon::Logger::save()	TU38
model::beacon::MyBeacon::getBatteryLevel()	TU36
model::beacon::MyBeacon::getBeaconTypeCode()	TU36
model::beacon::MyBeacon::getBluetoothAddress()	TU36
model::beacon::MyBeacon::getDistance()	TU36
model::beacon::MyBeacon::getMajor()	TU36
model::beacon::MyBeacon::getMinor()	TU36
model::beacon::MyBeacon::getRssi()	TU36
model::beacon::MyBeacon::getTxPower()	TU36
model::beacon::MyBeacon::getUUID()	TU36
model::compass::Compass::getLastCoordinate()	TU112
model::compass::Compass::registerListener()	TU110
model::compass::Compass::unregisterListener()	TU111
model::dataaccess::dao::BuildingTable::getAddress()	TU79
model::dataaccess::dao::BuildingTable::getDescription()	TU79
model::dataaccess::dao::BuildingTable::getId()	TU79
model::dataaccess::dao::BuildingTable::getMajor()	TU79
model::dataaccess::dao::BuildingTable::getName()	TU79
model::dataaccess::dao::BuildingTable::getOpeningHours()	TU79
model::dataaccess::dao::BuildingTable::getSize()	TU79
model::dataaccess::dao::BuildingTable::getUUID()	TU79
model::dataaccess::dao::BuildingTable::getVersion()	TU79
model::dataaccess::dao::CategoryTable::getDescription()	TU85

Metodo	Test
model::dataaccess::dao::CategoryTable::getId()	TU85
model::dataaccess::dao::DaoFactoryHelper::- getRemoteDaoFactory()	TU69
model::dataaccess::dao::DaoFactoryHelper::- getSQLiteDaoFactory()	TU69
model::dataaccess::dao::EdgeTable::getAction()	TU83
model::dataaccess::dao::EdgeTable::getCoordinate()	TU83
model::dataaccess::dao::EdgeTable::getDistance()	TU83
model::dataaccess::dao::EdgeTable::getEndROI()	TU83
model::dataaccess::dao::EdgeTable::getId()	TU83
model::dataaccess::dao::EdgeTable::getLongDescription()	TU83
model::dataaccess::dao::EdgeTable::getStartROI()	TU83
model::dataaccess::dao::EdgeTable::getTypeId()	TU83
model::dataaccess::dao::EdgeTypeTable::getId()	TU84
model::dataaccess::dao::EdgeTypeTable::get TypeName()	TU84
model::dataaccess::dao::MapsDbHelper::getInstance()	TU106
model::dataaccess::dao::MapsDbHelper::- getRemoteDatabaseURL()	TU109
model::dataaccess::dao::MapsDbHelper::onCreate()	TU107
model::dataaccess::dao::MapsDbHelper::onUpgrade()	TU108
model::dataaccess::dao::PhotoTable::getEdgeId()	TU86
model::dataaccess::dao::PhotoTable::getId()	TU86
model::dataaccess::dao::PhotoTable::getUrl()	TU86
model::dataaccess::dao::PointOfInterestTable::- getCategoryId()	TU80

Metodo	Test
model::dataaccess::dao::PointOfInterestTable::- getDescription()	TU80
model::dataaccess::dao::PointOfInterestTable::getId()	TU80
model::dataaccess::dao::PointOfInterestTable::getName()	TU80
model::dataaccess::dao::RegionOfInterestTable::getId()	TU81
model::dataaccess::dao::RegionOfInterestTable::getMajor()	TU81
model::dataaccess::dao::RegionOfInterestTable::getMinor()	TU81
model::dataaccess::dao::RegionOfInterestTable::getUUID()	TU81
model::dataaccess::dao::RemoteBuildingDao::- fromJSONToTable()	TU71
model::dataaccess::dao::RemoteCategoryDao::- fromJSONToTable()	TU77
model::dataaccess::dao::RemoteDaoFactory::- getBuildingDao()	TU70
model::dataaccess::dao::RemoteDaoFactory::- getCategoryDao()	TU70
model::dataaccess::dao::RemoteDaoFactory::getEdgeDao()	TU70
model::dataaccess::dao::RemoteDaoFactory::- getEdgeTypeDao()	TU70
model::dataaccess::dao::RemoteDaoFactory::getPhotoDao()	TU70
model::dataaccess::dao::RemoteDaoFactory::- getPointOfInterestDao()	TU70
model::dataaccess::dao::RemoteDaoFactory::- getRegionOfInterestDao()	TU70
model::dataaccess::dao::RemoteDaoFactory::getRoiPoiDao()	TU70
model::dataaccess::dao::RemoteEdgeDao::- fromJSONToTable()	TU75

Metodo	Test
model::dataaccess::dao::RemoteEdgeTypeDao::- fromJSONToTable()	TU76
model::dataaccess::dao::RemotePhotoDao::- fromJSONToTable()	TU78
model::dataaccess::dao::RemotePointOfInterestDao::- fromJSONToTable()	TU72
model::dataaccess::dao::RemoteRegionOfInterestDao::- fromJSONToTable()	TU73
model::dataaccess::dao::RemoteRoiPoiDao::- fromJSONToTable()	TU74
model::dataaccess::dao::RoiPoiTable::getPoiID()	TU82
model::dataaccess::dao::RoiPoiTable::getRoiID()	TU82
model::dataaccess::dao::SQLDao::delete()	TU105
model::dataaccess::dao::SQLDao::insert()	TU105
model::dataaccess::dao::SQLDao::query()	TU105
model::dataaccess::dao::SQLDao::rawQuery()	TU105
model::dataaccess::dao::SQLDao::update()	TU105
model::dataaccess::dao::SQLiteBuildingDao::cursorToType()	TU89
model::dataaccess::dao::SQLiteBuildingDao::deleteBuilding()	TU88
model::dataaccess::dao::SQLiteBuildingDao::- findAllBuildings()	TU88
model::dataaccess::dao::SQLiteBuildingDao::- findBuildingById()	TU88
model::dataaccess::dao::SQLiteBuildingDao::- findBuildingByMajor()	TU88
model::dataaccess::dao::SQLiteBuildingDao::insertBuilding()	TU88
model::dataaccess::dao::SQLiteBuildingDao::- isBuildingMapPresent()	TU90

Metodo	Test
model::dataaccess::dao::SQLiteBuildingDao::- updateBuilding()	TU88
model::dataaccess::dao::SQLiteCategoryDao::cursorToType()	TU100
model::dataaccess::dao::SQLiteCategoryDao::- deleteCategory()	TU99
model::dataaccess::dao::SQLiteCategoryDao::findCategory()	TU99
model::dataaccess::dao::SQLiteCategoryDao::- insertCategory()	TU99
model::dataaccess::dao::SQLiteCategoryDao::- updateCategory()	TU99
model::dataaccess::dao::SQLiteDaoFactory::getBuildingDao()	TU87
model::dataaccess::dao::SQLiteDaoFactory::- getCategoryDao()	TU87
model::dataaccess::dao::SQLiteDaoFactory::getEdgeDao()	TU87
model::dataaccess::dao::SQLiteDaoFactory::- getEdgeTypeDao()	TU87
model::dataaccess::dao::SQLiteDaoFactory::getPhotoDao()	TU87
model::dataaccess::dao::SQLiteDaoFactory::- getPointOfInterestDao()	TU87
model::dataaccess::dao::SQLiteDaoFactory::- getRegionOfInterestDao()	TU87
model::dataaccess::dao::SQLiteDaoFactory::getRoiPoiDao()	TU87
model::dataaccess::dao::SQLiteEdgeDao::cursorToType()	TU98
model::dataaccess::dao::SQLiteEdgeDao::deleteEdge()	TU97
model::dataaccess::dao::SQLiteEdgeDao::- findAllEdgesOfBuilding()	TU97
model::dataaccess::dao::SQLiteEdgeDao::findEdge()	TU97

Metodo	Test
model::dataaccess::dao::SQLiteEdgeDao::insertEdge()	TU97
model::dataaccess::dao::SQLiteEdgeDao::updateEdge()	TU97
model::dataaccess::dao::SQLiteEdgeTypeDao::-cursorToType()	TU102
model::dataaccess::dao::SQLiteEdgeTypeDao::-deleteEdgeType()	TU101
model::dataaccess::dao::SQLiteEdgeTypeDao::-findEdgeType()	TU101
model::dataaccess::dao::SQLiteEdgeTypeDao::-insertEdgeType()	TU101
model::dataaccess::dao::SQLiteEdgeTypeDao::-updateEdgeType()	TU101
model::dataaccess::dao::SQLitePhotoDao::cursorToType()	TU104
model::dataaccess::dao::SQLitePhotoDao::deletePhoto()	TU103
model::dataaccess::dao::SQLitePhotoDao::-findAllPhotosOfEdge()	TU103
model::dataaccess::dao::SQLitePhotoDao::findPhoto()	TU103
model::dataaccess::dao::SQLitePhotoDao::insertPhoto()	TU103
model::dataaccess::dao::SQLitePhotoDao::updatePhoto()	TU103
model::dataaccess::dao::SQLitePointOfInterestDao::-cursorToType()	TU92
model::dataaccess::dao::SQLitePointOfInterestDao::-deletePointOfInterest()	TU91
model::dataaccess::dao::SQLitePointOfInterestDao::-findAllPointsWithMajor()	TU91
model::dataaccess::dao::SQLitePointOfInterestDao::-findPointOfInterest()	TU91
model::dataaccess::dao::SQLitePointOfInterestDao::-insertPointOfInterest()	TU91

Metodo	Test
model::dataaccess::dao::SQLitePointOfInterestDao::- updatePointOfInterest()	TU91
model::dataaccess::dao::SQLiteRegionOfInterestDao::- cursorToType()	TU94
model::dataaccess::dao::SQLiteRegionOfInterestDao::- deleteRegionOfInterest()	TU93
model::dataaccess::dao::SQLiteRegionOfInterestDao::- findAllRegionsWithMajor()	TU93
model::dataaccess::dao::SQLiteRegionOfInterestDao::- findRegionOfInterest()	TU93
model::dataaccess::dao::SQLiteRegionOfInterestDao::- insertRegionOfInterest()	TU93
model::dataaccess::dao::SQLiteRegionOfInterestDao::- updateRegionOfInterest()	TU93
model::dataaccess::dao::SQLiteRoiPoiDao::cursorToType()	TU96
model::dataaccess::dao::SQLiteRoiPoiDao::- deleteRoiPoisWherePoi()	TU95
model::dataaccess::dao::SQLiteRoiPoiDao::- deleteRoiPoisWhereRoi()	TU95
model::dataaccess::dao::SQLiteRoiPoiDao::- findAllPointsWithRoi()	TU95
model::dataaccess::dao::SQLiteRoiPoiDao::- findAllRegionsWithPoi()	TU95
model::dataaccess::dao::SQLiteRoiPoiDao::insertRoiPoi()	TU95
model::dataaccess::dao::SQLiteRoiPoiDao::updateRoiPoi()	TU95
model::dataaccess::service::BuildingService::- convertAndInsert()	TU66
model::dataaccess::service::BuildingService::deleteBuilding()	TU63

Metodo	Test
model::dataaccess::service::BuildingService::- findAllBuildings()	TU63
model::dataaccess::service::BuildingService::- findAllRemoteBuildings()	TU64
model::dataaccess::service::BuildingService::- findBuildingByMajor()	TU63
model::dataaccess::service::BuildingService::- findRemoteBuildingByMajor()	TU64
model::dataaccess::service::BuildingService::fromTableToBo()	TU68
model::dataaccess::service::BuildingService::- isBuildingMapPresent()	TU67
model::dataaccess::service::BuildingService::- isBuildingMapUpdated()	TU67
model::dataaccess::service::BuildingService::- retrieveAndInsertMap()	TU65
model::dataaccess::service::BuildingService::- updateBuildingMap()	TU67
model::dataaccess::service::EdgeService::convertAndInsert()	TU53
model::dataaccess::service::EdgeService::- convertAndInsertEdgeType()	TU54
model::dataaccess::service::EdgeService::deleteEdge()	TU52
model::dataaccess::service::EdgeService::- findAllEdgesOfBuilding()	TU52
model::dataaccess::service::EdgeService::findEdge()	TU52
model::dataaccess::service::EdgeService::fromTableToBo()	TU55
model::dataaccess::service::PhotoService::convertAndInsert()	TU48
model::dataaccess::service::PhotoService::deletePhoto()	TU46
model::dataaccess::service::PhotoService::- findAllPhotosOfEdge()	TU46

Metodo	Test
model::dataaccess::service::PhotoService::findPhoto()	TU46
model::dataaccess::service::PhotoService::fromTableToBo()	TU47
model::dataaccess::service::PointOfInterestService::- convertAndInsert()	TU59
model::dataaccess::service::PointOfInterestService::- convertAndInsertCategory()	TU60
model::dataaccess::service::PointOfInterestService::- convertAndInsertRoiPoi()	TU61
model::dataaccess::service::PointOfInterestService::- deletePointOfInterest()	TU58
model::dataaccess::service::PointOfInterestService::- findAllPointsWithMajor()	TU58
model::dataaccess::service::PointOfInterestService::- findPointOfInterest()	TU58
model::dataaccess::service::PointOfInterestService::- fromTableToBo()	TU62
model::dataaccess::service::RegionOfInterestService::- convertAndInsert()	TU51
model::dataaccess::service::RegionOfInterestService::- deleteRegionOfInterest()	TU49
model::dataaccess::service::RegionOfInterestService::- findAllRegionsWithMajor()	TU49
model::dataaccess::service::RegionOfInterestService::- findRegionOfInterest()	TU49
model::dataaccess::service::RegionOfInterestService::- fromTableToBo()	TU50
model::dataaccess::service::ServiceHelper::getService()	TU45
model::navigator::BuildingInformation::getAddress()	TU4
model::navigator::BuildingInformation::getDescription()	TU4

Metodo	Test
model::navigator::BuildingInformation::getName()	TU4
model::navigator::BuildingInformation::getOpeningHours()	TU4
model::navigator::BuildingInformation::toString()	TU4
model::navigator::BuildingMapImp::getAddress()	TU5
model::navigator::BuildingMapImp::- getAllBuildingInformation()	TU5
model::navigator::BuildingMapImp::getAllEdges()	TU5
model::navigator::BuildingMapImp::getAllNearbyPOIs()	TU6
model::navigator::BuildingMapImp::getAllPOIs()	TU5
model::navigator::BuildingMapImp::getAllROIs()	TU5
model::navigator::BuildingMapImp::getDescription()	TU5
model::navigator::BuildingMapImp::getId()	TU5
model::navigator::BuildingMapImp::getName()	TU5
model::navigator::BuildingMapImp::getOpeningHours()	TU5
model::navigator::BuildingMapImp::getSize()	TU5
model::navigator::BuildingMapImp::getVersion()	TU5
model::navigator::NavigatorImp::calculatePath()	TU33
model::navigator::NavigatorImp::getAllInstructions()	TU34
model::navigator::NavigatorImp::getPath()	TU33
model::navigator::NavigatorImp::setGraph()	TU33
model::navigator::NavigatorImp::toNextRegion()	TU35
model::navigator::ProcessedInformation::- getDetailedInstruction()	TU7
model::navigator::ProcessedInformation::- getPhotoInstruction()	TU7

Metodo	Test
model::navigator::ProcessedInformation::- getProcessedBasicInstruction()	TU7
model::navigator::algorithm::DijkstraPathFinder::- calculatePath()	TU32
model::navigator::graph::MapGraph::addAllEdges()	TU30
model::navigator::graph::MapGraph::addAllRegions()	TU30
model::navigator::graph::MapGraph::addEdge()	TU30
model::navigator::graph::MapGraph::getGraph()	TU31
model::navigator::graph::area::PointOfInterestImp::- getAllBelongingROIs()	TU25
model::navigator::graph::area::PointOfInterestImp::- getCategory()	TU24
model::navigator::graph::area::PointOfInterestImp::- getDescription()	TU24
model::navigator::graph::area::PointOfInterestImp::getId()	TU24
model::navigator::graph::area::PointOfInterestImp::- getName()	TU24
model::navigator::graph::area::PointOfInterestImp::- setBelongingROIs()	TU25
model::navigator::graph::area::PointOfInterestInformation::- getCategory()	TU23
model::navigator::graph::area::PointOfInterestInformation::- getDescription()	TU23
model::navigator::graph::area::PointOfInterestInformation::- getName()	TU23
model::navigator::graph::area::RegionOfInterestImp::- contains()	TU28
model::navigator::graph::area::RegionOfInterestImp::- getAllNearbyPOIs()	TU29

Metodo	Test
model::navigator::graph::area::RegionOfInterestImp::- getFloor()	TU27
model::navigator::graph::area::RegionOfInterestImp::- getMajor()	TU26
model::navigator::graph::area::RegionOfInterestImp::- getMinor()	TU26
model::navigator::graph::area::RegionOfInterestImp::- getUUID()	TU26
model::navigator::graph::area::RegionOfInterestImp::- setNearbyPOIs()	TU29
model::navigator::graph::edge::AbsEnrichedEdge::- getCoordinate()	TU14
model::navigator::graph::edge::AbsEnrichedEdge::- getEndPoint()	TU14
model::navigator::graph::edge::AbsEnrichedEdge::-getId()	TU14
model::navigator::graph::edge::AbsEnrichedEdge::- getNavigationInformation()	TU16
model::navigator::graph::edge::AbsEnrichedEdge::- getPhotoInformation()	TU14
model::navigator::graph::edge::AbsEnrichedEdge::- getStarterPoint()	TU14
model::navigator::graph::edge::AbsEnrichedEdge::- setUserPreference()	TU15
model::navigator::graph::edge::DefaultEdge::- getBasicInformation()	TU17
model::navigator::graph::edge::DefaultEdge::- getDetailedInformation()	TU17
model::navigator::graph::edge::DefaultEdge::-getWeight()	TU22
model::navigator::graph::edge::ElevatorEdge::- getBasicInformation()	TU19

Metodo	Test
model::navigator::graph::edge::ElevatorEdge::- getDetailedInformation()	TU19
model::navigator::graph::edge::ElevatorEdge::getWeight()	TU21
model::navigator::graph::edge::StairEdge::- getBasicInformation()	TU18
model::navigator::graph::edge::StairEdge::- getDetailedInformation()	TU18
model::navigator::graph::edge::StairEdge::getWeight()	TU20
model::navigator::graph::navigationinformation::- BasicInformation::getBasicInformation()	TU9
model::navigator::graph::navigationinformation::- DetailedInformation::getDetailedInformation()	TU10
model::navigator::graph::navigationinformation::- NavigationInformation::getBasicInformation()	TU13
model::navigator::graph::navigationinformation::- NavigationInformation::getDetailedInformation()	TU13
model::navigator::graph::navigationinformation::- NavigationInformation::getPhotoInformation()	TU13
model::navigator::graph::navigationinformation::- PhotoInformation::getPhotoInformation()	TU12
model::navigator::graph::navigationinformation::PhotoRef::- getPhotoUri()	TU11
model::navigator::graph::vertex::VertexImp::getId()	TU8
model::usersetting::DeveloperCodeManager::isValid()	TU2
model::usersetting::SettingImp::getInstructionPreference()	TU1
model::usersetting::SettingImp::getPathPreference()	TU1
model::usersetting::SettingImp::isDeveloper()	TU3
model::usersetting::SettingImp::setInstructionPreference()	TU1

Metodo	Test
model::usersetting::SettingImp::setPathPreference()	TU1
model::usersetting::SettingImp::unlockDeveloper()	TU3
presenter::Checker::isBeaconDetected()	TU113
presenter::Checker::isBluetoothEnabled()	TU113
presenter::Checker::isTheBuildingMapInLocalDB()	TU113
presenter::DatabaseServiceManager::getDatabaseService()	TU116
presenter::DetailedInformationActivity::-updateDetailedDescription()	TU145
presenter::DetailedInformationActivity::updatePhoto()	TU145
presenter::DeveloperUnlockerActivity::unlockDeveloper()	TU136
presenter::HelpActivity::onCreate()	TU142
presenter::HomeActivity::enableSuggestions()	TU124
presenter::HomeActivity::showHelp()	TU127
presenter::HomeActivity::showLocalMaps()	TU129
presenter::HomeActivity::showPoisCategory()	TU125
presenter::HomeActivity::showPreferences()	TU126
presenter::HomeActivity::startNavigation()	TU128
presenter::HomeActivity::updateBuildingAddress()	TU123
presenter::HomeActivity::updateBuildingDescription()	TU120
presenter::HomeActivity::updateBuildingName()	TU119
presenter::HomeActivity::updateBuildingOpeningHours()	TU121
presenter::HomeActivity::updatePoiCategoryList()	TU122
presenter::ImageAdapter::getCount()	TU133

Metodo	Test
presenter::ImageAdapter::getItem()	TU133
presenter::ImageAdapter::getItemId()	TU147
presenter::ImageAdapter::getView()	TU133
presenter::ImageDetailActivity::onCreate()	TU143
presenter::ImageListFragment::newInstance()	TU134
presenter::ImageListFragment::onCreateView()	TU134
presenter::ImageListFragment::onItemClick()	TU134
presenter::ImagePagerAdapter::getCount()	TU146
presenter::ImagePagerAdapter::getItem()	TU146
presenter::InformationManagerPresenter::-getInformationManager()	TU115
presenter::LocalMapActivity::deleteMap()	TU141
presenter::LocalMapActivity::updateMap()	TU141
presenter::LoggingActivity::stopLogging()	TU149
presenter::LogInformationActivity::deleteLog()	TU150
presenter::MainActivity::onCreate()	TU148
presenter::MainDeveloperActivity::showDetailedLog()	TU137
presenter::MainDeveloperActivity::startNewLog()	TU137
presenter::MainDeveloperPresenter::isDeveloper()	TU135
presenter::MainDeveloperPresenter::startDeveloperOptions()	TU135
presenter::MainDeveloperPresenter::startDeveloperUnlocker()	TU135
presenter::MapDownloaderActivity::downloadFinished()	TU139
presenter::NavigationActivity::informationUpdate()	TU131

---

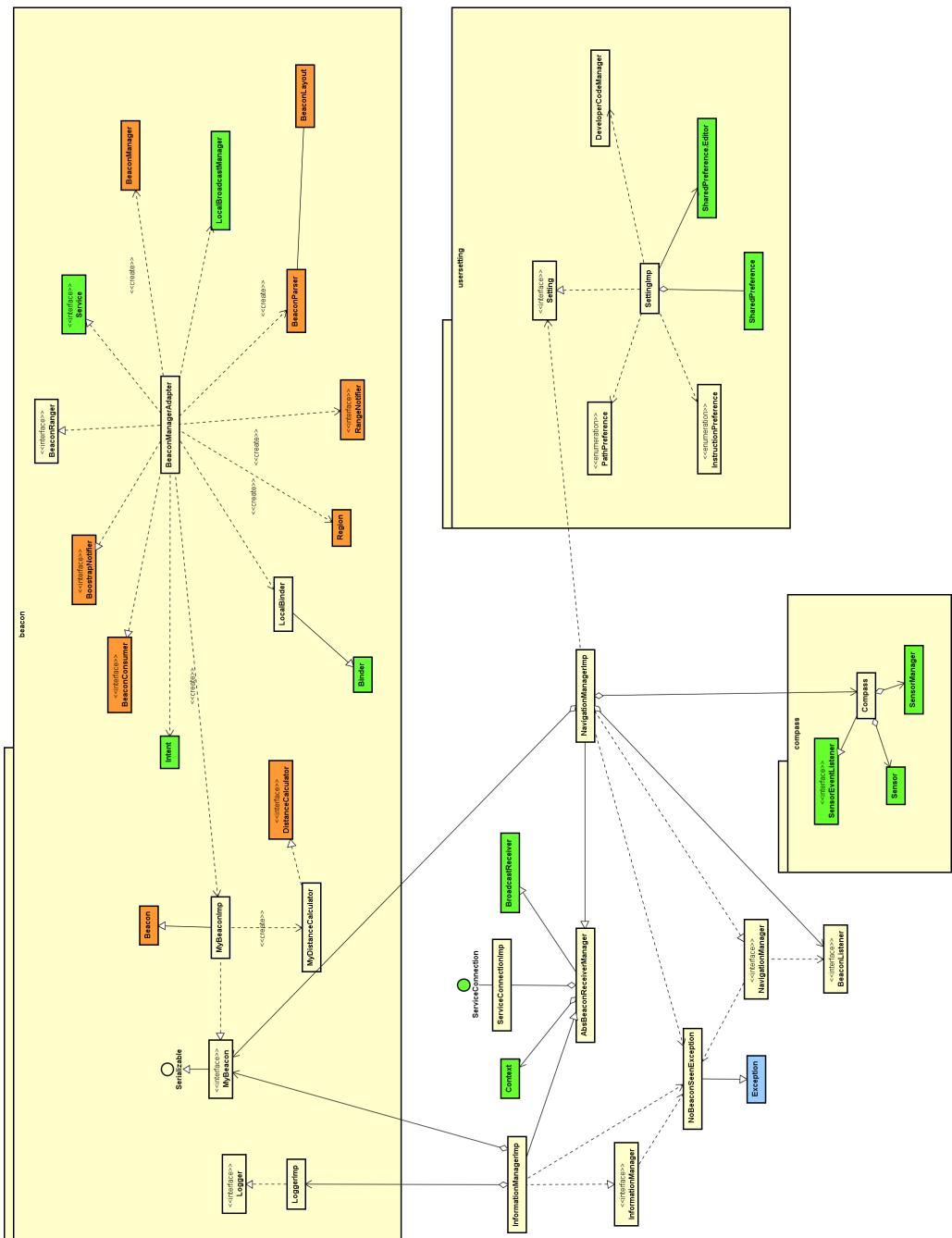
Metodo	Test
presenter::NavigationActivity::pathError()	TU131
presenter::NavigationActivity::showDetailedInformation()	TU132
presenter::NavigationAdapter::getCount()	TU151
presenter::NavigationAdapter::getItem()	TU151
presenter::NavigationAdapter::getItemId()	TU151
presenter::NavigationAdapter::getView()	TU151
presenter::NavigationManagerPresenter::-getNavigationManager()	TU114
presenter::NearbyPoiActivity::onCreate()	TU144
presenter::PoiCategoryActivity::startNavigation()	TU130
presenter::PreferencesActivity::savePreferences()	TU138
presenter::RemoteMapManagerActivity::downloadMap()	TU140
presenter::SearchSuggestionsProvider::getType()	TU118
presenter::SearchSuggestionsProvider::insert()	TU118
presenter::SearchSuggestionsProvider::onCreate()	TU118
presenter::SearchSuggestionsProvider::query()	TU118
presenter::SearchSuggestionsProvider::update()	TU118
presenter::SettingManager::getSetting()	TU117
view::DetailedInformationViewImp::setDetailedDescription()	TU155
view::DetailedInformationViewImp::setPhotos()	TU155
view::DeveloperUnlockerViewImp::showWrongCode()	TU163
view::HelpViewImp::setHelp()	TU158
view::HomeViewImp::setBuildingAddress()	TU152

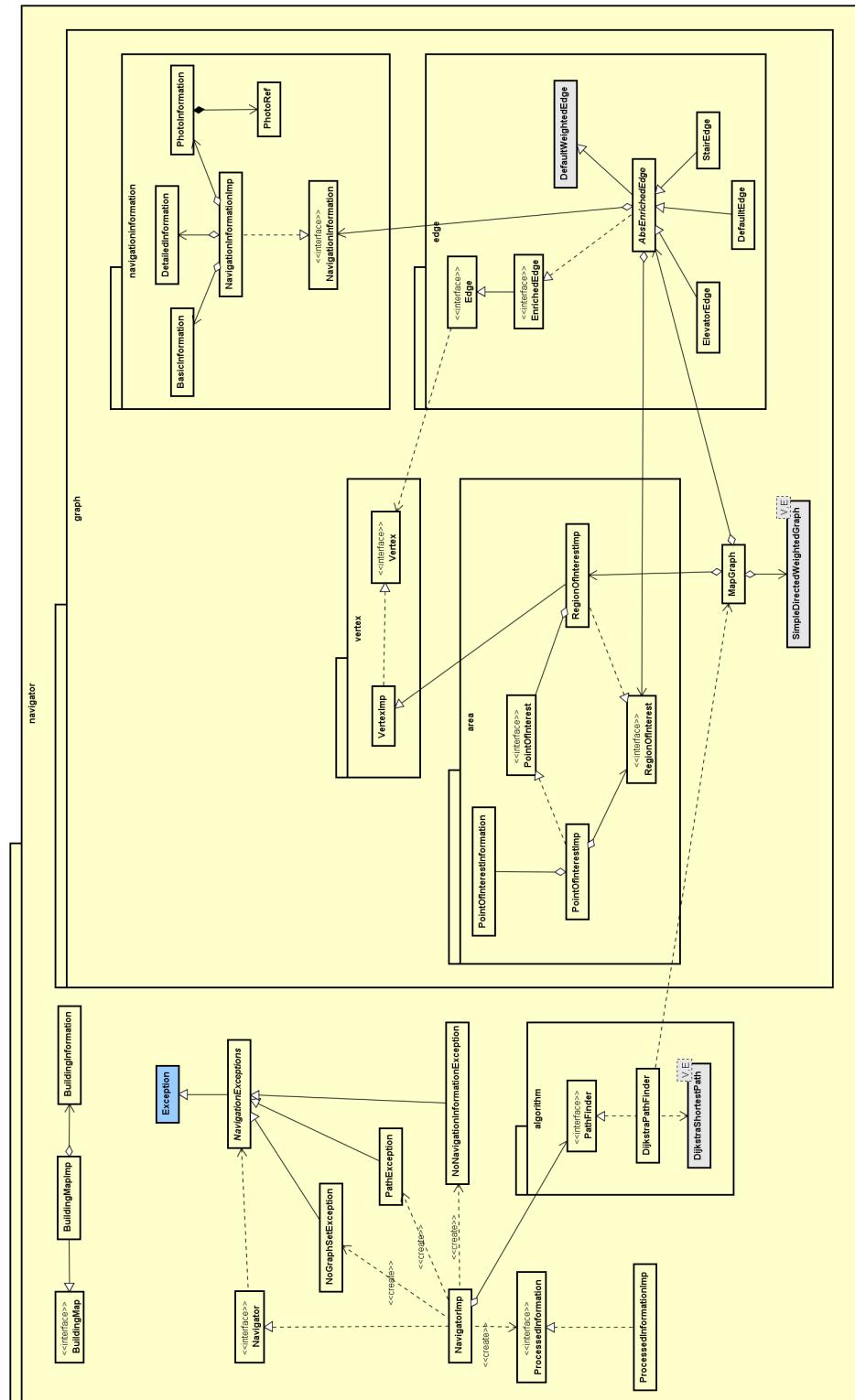
Metodo	Test
view::HomeViewImp::setBuildingName()	TU152
view::HomeViewImp::setBuildingOpeningHours()	TU152
view::HomeViewImp::setPoiCategoryListAdapter()	TU152
view::ImageDetailViewImp::setAdapter()	TU157
view::ImageListFragmentViewImp::setImageAdapter()	TU167
view::LocalMapManagerViewImp::refreshMaps()	TU159
view::LocalMapManagerViewImp::setAdapter()	TU159
view::LoggingViewImp::setBeaconListAdapter()	TU165
view::LogInformationViewImp::setBeaconAdapter()	TU166
view::MainDeveloperViewImp::setLogsAdapter()	TU164
view::MainView:: setLoadingView()	TU168
view::MapDownloaderView::setDowloadingMap()	TU161
view::MapDownloaderView::setProgressDowload()	TU161
view::NavigationViewImp::refreshInstructions()	TU153
view::NavigationViewImp::setInstructionAdapter()	TU153
view::NearbyPoiViewImp::setAdapter()	TU156
view::PoiCategoryViewImp::setPoiListAdapter()	TU154
view::PreferencesViewImp::setInstructionPreferences()	TU162
view::PreferencesViewImp::setPathPreferences()	TU162
view::RemoteMapManagerViewImp::setRemoteMaps()	TU160

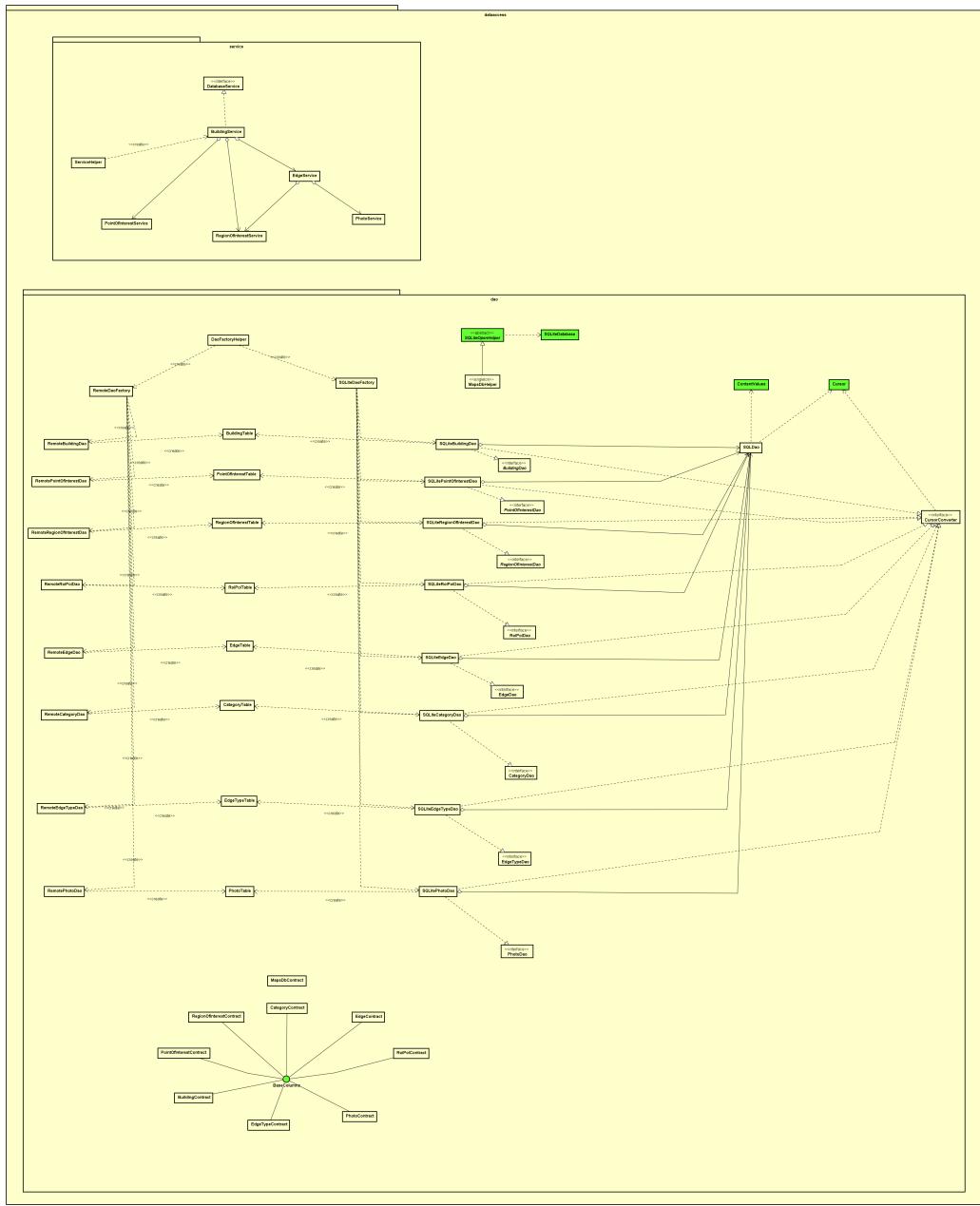
**Tabella 3:** Tabella metodi / test unità

## A Diagrammi riassuntivi package significativi

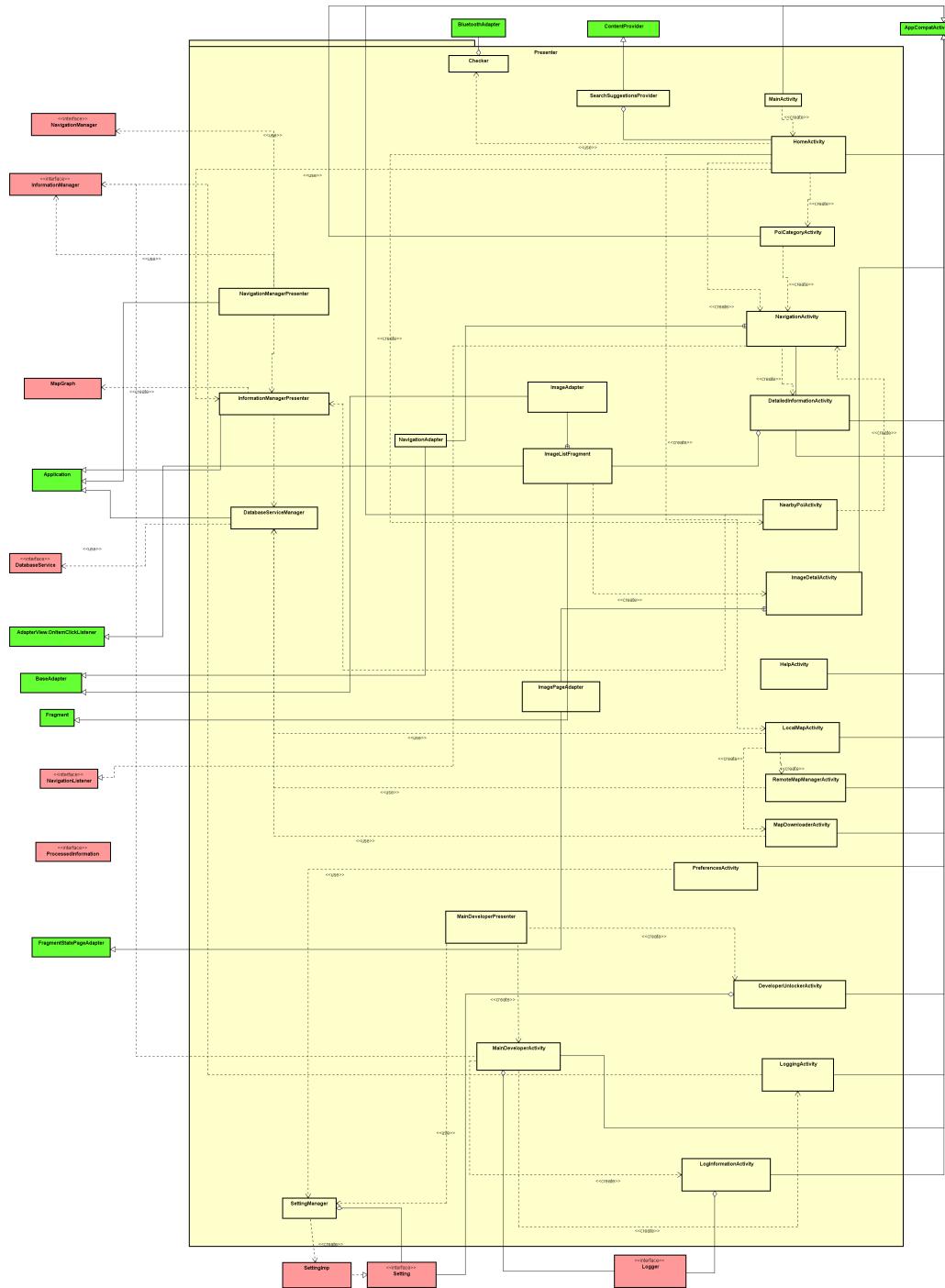
Di seguito sono riportati tutti i package dell'applicativo ad eccezione della `view`, essendo applicato il pattern MVP, per chiarire la relazione tra le componenti e le classi al suo interno. Per chiarezza ed esigenza di spazio le classi rappresentate all'interno dei package sono senza metodi e attributi.


**Figura 194:** Diagramma delle classi - model


**Figura 195:** Diagramma delle classi - model::navigator



**Figura 196:** Diagramma delle classi - model::dataaccess



**Figura 197:** Diagramma delle classi - presenter