

CLIPS

Communication & Localization with Indoor Positioning Systems

UNIVERSITÀ DI PADOVA

DEFINIZIONE DI PRODOTTO v2.00



leaf.gruppo@gmail.com

Versione	2.00
Data Redazione	2016-04-23
Redazione	Marco Zanella Andrea Tombolato
Verifica	Oscar Elia Conti
Approvazione	Davide Castello
Uso	Esterno
Distribuzione	Prof. Vardanega Tullio Prof. Cardin Riccardo Miriade S.p.A.

Diario delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
2.02	2016-05-30	Eduard Bicego	Programmatore	Miglioramento leggibilità diagrammi riassunti package
2.01	2016-05-28	Marco Zanella	Programmatore	Correzione descrizione metodi <code>toInt()</code> e <code>fromInt()</code> classi <code>PathPreference</code> e <code>InstructionPreference</code>
2.00	2016-04-23	Davide Castello	Responsabile di progetto	Approvazione del documento
1.04	2016-04-23	Oscar Elia Conti	Verificatore	Verifica del documento
1.03	2016-04-21	Marco Zanella	Progettista	Aggiungi diagrammi di sequenza
1.02	2016-04-20	Andrea Tombolato	Progettista	Aggiunta sezione architettura applicazione e diagramma di sequenza Avvio bussola
1.01	2016-04-19	Andrea Tombolato	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 Andriguetto	Verificatore	Verifica del documento
0.19	2016-04-10	Davide Castello	Progettista	Aggiunto tracciamento classi-requisiti e viceversa

Versione	Data	Autore	Ruolo	Descrizione
0.18	2016-04-09	Eduard Bicego	Progettista	Correzioni delle classi ImageDetailActivity, ImageListFragment, ImageListFragmentViewImp, PreferenceViewImp, LoggingView, LoggingViewImp, DeveloperUnlockedViewImp, HelpViewImp, MapDownloaderActivity e LogInformationActivity
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 Andriguetto	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

Versione	Data	Autore	Ruolo	Descrizione
0.11	2016-04-07	Cristian Andrichetto	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	Aggiunti diagrammi di sequenza Avvio Service, Ranging Beacons e avvio navigazione
0.05	2016-04-03	Eduard Bicego	Progettista	Completata sezione Standard di progetto
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

1	Introduzione	1
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
2	Standard di progetto	3
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
3	Architettura applicazione	4
3.1	Architettura ad alto livello	4
3.2	Implementazione nell'applicativo	5
4	Specifiche dei componenti	6
4.1	Metodo e formalismo di specifica	6
4.2	Sistema CLIPS	7
4.3	Componenti	8
4.3.1	di	8
4.3.2	di::component	8
4.3.3	di::module	9
4.3.4	model	10
4.3.5	model::beacon	11
4.3.6	model::compass	12
4.3.7	model::dataaccess	12
4.3.8	model::dataaccess::dao	13
4.3.9	model::dataaccess::service	15
4.3.10	model::navigator	16
4.3.11	model::navigator::algorithm	17
4.3.12	model::navigator::graph	18
4.3.13	model::navigator::graph::area	19
4.3.14	model::navigator::graph::edge	19
4.3.15	model::navigator::graph::navigationinformation	20
4.3.16	model::navigator::graph::vertex	21
4.3.17	model::usersetting	21

4.3.18	presenter	22
4.3.19	view	24
4.4	Classi	25
4.4.1	di	25
4.4.1.1	di::component::InfoComponent	25
4.4.1.2	di::module::AppModule	30
4.4.1.3	di::module::DatabaseModule	31
4.4.1.4	di::module::InfoModule	33
4.4.1.5	di::module::NavModule	34
4.4.1.6	di::module::SettingModule	35
4.4.2	model	36
4.4.2.1	model::AbsBeaconReceiverManager	36
4.4.2.2	model::InformationListener	38
4.4.2.3	model::InformationManager	40
4.4.2.4	model::InformationManagerImp	43
4.4.2.5	model::Listener	48
4.4.2.6	model::MessageSendType	49
4.4.2.7	model::NavigationListener	49
4.4.2.8	model::NavigationManager	51
4.4.2.9	model::NavigationManagerImp	52
4.4.2.10	model::NoBeaconSeenException	55
4.4.2.11	model::ServiceConnectionImp	57
4.4.2.12	model::beacon::BeaconManagerAdapter . . .	58
4.4.2.13	model::beacon::BeaconRanger	61
4.4.2.14	model::beacon::LocalBinder	63
4.4.2.15	model::beacon::Logger	64
4.4.2.16	model::beacon::LoggerImp	65
4.4.2.17	model::beacon::MyBeacon	67
4.4.2.18	model::beacon::MyBeaconImp	69
4.4.2.19	model::beacon::MyDistanceCalculator	71
4.4.2.20	model::beacon::PeriodType	72
4.4.2.21	model::compass::Compass	72
4.4.2.22	model::compass::CompassListener	76
4.4.2.23	model::dataaccess::dao::BuildingContract . .	76
4.4.2.24	model::dataaccess::dao::BuildingDao	78
4.4.2.25	model::dataaccess::dao::BuildingTable	80
4.4.2.26	model::dataaccess::dao::CategoryContract . .	82
4.4.2.27	model::dataaccess::dao::CategoryDao	83
4.4.2.28	model::dataaccess::dao::CategoryTable	84
4.4.2.29	model::dataaccess::dao::CursorConverter . . .	86
4.4.2.30	model::dataaccess::dao::DaoFactoryHelper . .	87

4.4.2.31	model::dataaccess::dao::EdgeContract	88
4.4.2.32	model::dataaccess::dao::EdgeDao	89
4.4.2.33	model::dataaccess::dao::EdgeTable	91
4.4.2.34	model::dataaccess::dao::EdgeTypeContract . .	93
4.4.2.35	model::dataaccess::dao::EdgeTypeDao	94
4.4.2.36	model::dataaccess::dao::EdgeTypeTable	95
4.4.2.37	model::dataaccess::dao::MapsDbHelper	97
4.4.2.38	model::dataaccess::dao::PhotoContract	98
4.4.2.39	model::dataaccess::dao::PhotoDao	99
4.4.2.40	model::dataaccess::dao::PhotoTable	101
4.4.2.41	model::dataaccess::dao::PointOfInterestContract	102
4.4.2.42	model::dataaccess::dao::PointOfInterestDao .	103
4.4.2.43	model::dataaccess::dao::PointOfInterestTable .	105
4.4.2.44	model::dataaccess::dao::RegionOfInterestContract	106
4.4.2.45	model::dataaccess::dao::RegionOfInterestDao .	107
4.4.2.46	model::dataaccess::dao::RegionOfInterestTable	109
4.4.2.47	model::dataaccess::dao::RemoteBuildingDao .	111
4.4.2.48	model::dataaccess::dao::RemoteCategoryDao .	112
4.4.2.49	model::dataaccess::dao::RemoteDaoFactory .	113
4.4.2.50	model::dataaccess::dao::RemoteEdgeDao . . .	114
4.4.2.51	model::dataaccess::dao::RemoteEdgeTypeDao	116
4.4.2.52	model::dataaccess::dao::RemotePhotoDao . . .	117
4.4.2.53	model::dataaccess::dao::RemotePointOfInterestDao	118
4.4.2.54	model::dataaccess::dao::RemoteRegionOfInterestDao	119
4.4.2.55	model::dataaccess::dao::RemoteRoiPoiDao . .	120
4.4.2.56	model::dataaccess::dao::RoiPoiContract	121
4.4.2.57	model::dataaccess::dao::RoiPoiDao	121
4.4.2.58	model::dataaccess::dao::RoiPoiTable	124
4.4.2.59	model::dataaccess::dao::SQLDao	125
4.4.2.60	model::dataaccess::dao::SQLiteBuildingDao .	127
4.4.2.61	model::dataaccess::dao::SQLiteCategoryDao .	130
4.4.2.62	model::dataaccess::dao::SQLiteDaoFactory .	132
4.4.2.63	model::dataaccess::dao::SQLiteEdgeDao . . .	134
4.4.2.64	model::dataaccess::dao::SQLiteEdgeTypeDao	136
4.4.2.65	model::dataaccess::dao::SQLitePhotoDao . . .	139
4.4.2.66	model::dataaccess::dao::SQLitePointOfInterestDao	141
4.4.2.67	model::dataaccess::dao::SQLiteRegionOfInterestDao	143
4.4.2.68	model::dataaccess::dao::SQLiteRoiPoiDao . .	145
4.4.2.69	model::dataaccess::service::BuildingService .	148
4.4.2.70	model::dataaccess::service::DatabaseService .	152
4.4.2.71	model::dataaccess::service::EdgeService	154

4.4.2.72	model::dataaccess::service::PhotoService . . .	157
4.4.2.73	model::dataaccess::service::PointOfInterestService	159
4.4.2.74	model::dataaccess::service::RegionOfInterestService	162
4.4.2.75	model::dataaccess::service::ServiceHelper . . .	165
4.4.2.76	model::navigator::BuildingInformation	166
4.4.2.77	model::navigator::BuildingMap	168
4.4.2.78	model::navigator::BuildingMapImp	170
4.4.2.79	model::navigator::NavigationDirection	174
4.4.2.80	model::navigator::NavigationExceptions . . .	175
4.4.2.81	model::navigator::Navigator	176
4.4.2.82	model::navigator::NavigatorImp	177
4.4.2.83	model::navigator::NoGraphSetException . . .	181
4.4.2.84	model::navigator::NoNavigationInformationException	182
4.4.2.85	model::navigator::PathException	184
4.4.2.86	model::navigator::ProcessedInformation . . .	185
4.4.2.87	model::navigator::ProcessedInformationImp .	186
4.4.2.88	model::navigator::algorithm::DijkstraPathFinder	189
4.4.2.89	model::navigator::algorithm::PathFinder . . .	190
4.4.2.90	model::navigator::graph::MapGraph	191
4.4.2.91	model::navigator::graph::area::PointOfInterest	193
4.4.2.92	model::navigator::graph::area::PointOfInterestImp	194
4.4.2.93	model::navigator::graph::area::PointOfInterestInformation	196
4.4.2.94	model::navigator::graph::area::RegionOfInterest	197
4.4.2.95	model::navigator::graph::area::RegionOfInterestImp	199
4.4.2.96	model::navigator::graph::edge::AbsEnrichedEdge	202
4.4.2.97	model::navigator::graph::edge::DefaultEdge .	205
4.4.2.98	model::navigator::graph::edge::Edge	206
4.4.2.99	model::navigator::graph::edge::ElevatorEdge .	207
4.4.2.100	model::navigator::graph::edge::EnrichedEdge .	209
4.4.2.101	model::navigator::graph::edge::StairEdge . . .	211
4.4.2.102	model::navigator::graph::navigationinformation::BasicInformation	212
4.4.2.103	model::navigator::graph::navigationinformation::DetailedInformation	212
4.4.2.104	model::navigator::graph::navigationinformation::NavigationInformation	
4.4.2.105	model::navigator::graph::navigationinformation::NavigationInformation	
4.4.2.106	model::navigator::graph::navigationinformation::PhotoInformation	216
4.4.2.107	model::navigator::graph::navigationinformation::PhotoRef	217
4.4.2.108	model::navigator::graph::vertex::Vertex	219
4.4.2.109	model::navigator::graph::vertex::VertexImp . .	219
4.4.2.110	model::usersetting::DeveloperCodeManager .	220
4.4.2.111	model::usersetting::InstructionPreference . . .	221
4.4.2.112	model::usersetting::PathPreference	223

4.4.2.113	model::usersetting::Setting	224
4.4.2.114	model::usersetting::SettingImp	225
4.4.3	presenter	228
4.4.3.1	presenter::BeaconPos	228
4.4.3.2	presenter::BeaconPowerAreaActivity	229
4.4.3.3	presenter::BeaconPowerPos	232
4.4.3.4	presenter::BlankHomeFragment	233
4.4.3.5	presenter::CompleteHomeFragment	234
4.4.3.6	presenter::DetailedInformationActivity	237
4.4.3.7	presenter::DeveloperUnlockerActivity	239
4.4.3.8	presenter::HelpActivity	240
4.4.3.9	presenter::HomeActivity	241
4.4.3.10	presenter::ImageAdapter	246
4.4.3.11	presenter::ImageDetailActivity	248
4.4.3.12	presenter::ImageDetailFragment	249
4.4.3.13	presenter::ImageListFragment	251
4.4.3.14	presenter::LocalMapActivity	253
4.4.3.15	presenter::LocalMapAdapter	254
4.4.3.16	presenter::LoggingActivity	256
4.4.3.17	presenter::LogInformationActivity	258
4.4.3.18	presenter::MainActivity	259
4.4.3.19	presenter::MainDeveloperActivity	260
4.4.3.20	presenter::MainDeveloperPresenter	261
4.4.3.21	presenter::MyApplication	263
4.4.3.22	presenter::NavigationActivity	264
4.4.3.23	presenter::NavigationAdapter	267
4.4.3.24	presenter::NearbyPoiActivity	269
4.4.3.25	presenter::NoInternetAlert	270
4.4.3.26	presenter::PoiActivity	271
4.4.3.27	presenter::PoiCategoryActivity	273
4.4.3.28	presenter::PoiDescriptionActivity	274
4.4.3.29	presenter::PreferencesActivity	275
4.4.3.30	presenter::RemoteMapManagerActivity	277
4.4.3.31	presenter::RemoteMapManagerAdapter	278
4.4.3.32	presenter::SearchSuggestionsProvider	280
4.4.4	view	282
4.4.4.1	view::BeaconPowerArea	282
4.4.4.2	view::BeaconPowerAreaView	284
4.4.4.3	view::BeaconPowerAreaViewImp	285
4.4.4.4	view::DescriptionView	287
4.4.4.5	view::DescriptionViewImp	288

4.4.4.6	view::DetailedInformationView	289
4.4.4.7	view::DetailedInformationViewImp	290
4.4.4.8	view::DeveloperUnlockerView	292
4.4.4.9	view::DeveloperUnlockerViewImp	292
4.4.4.10	view::HelpView	294
4.4.4.11	view::HelpViewImp	295
4.4.4.12	view::HomeView	296
4.4.4.13	view::HomeViewImp	298
4.4.4.14	view::ImageDetailView	300
4.4.4.15	view::ImageDetailViewImp	301
4.4.4.16	view::LocalMapManagerView	302
4.4.4.17	view::LocalMapManagerViewImp	303
4.4.4.18	view::LoggingView	304
4.4.4.19	view::LoggingViewImp	305
4.4.4.20	view::LogInformationView	306
4.4.4.21	view::LogInformationViewImp	307
4.4.4.22	view::MainDeveloperView	309
4.4.4.23	view::MainDeveloperViewImp	309
4.4.4.24	view::NavigationView	311
4.4.4.25	view::NavigationViewImp	312
4.4.4.26	view::NearbyPoiView	314
4.4.4.27	view::NearbyPoiViewImp	315
4.4.4.28	view::PoiCategoryView	317
4.4.4.29	view::PoiCategoryViewImp	317
4.4.4.30	view::PoiView	319
4.4.4.31	view::PoiViewImp	320
4.4.4.32	view::PreferencesView	321
4.4.4.33	view::PreferencesViewImp	322
4.4.4.34	view::RemoteMapManagerView	323
4.4.4.35	view::RemoteMapManagerViewImp	324
5	Schema base di dati	326
6	Diagrammi di sequenza	327
6.1	Avvio Service per il rilevamento beacon	327
6.2	Elaborazione beacon rilevati e comunicazione broadcast	329
6.3	Avvio navigazione	331
6.4	Avvio della bussola	332
6.5	Caricamento della mappa dal database	333
6.6	Costruzione di oggetti-Table da Json	334
6.7	Modifica delle preferenze	337

7 Tracciamento	338
7.1 Tracciamento Classi-Requisiti	338
7.2 Requisiti-Classi	353
7.3 Tracciamento Metodi - test di unità	371

Elenco delle figure

1	Architettura a layer	4
2	Architettura a layer implementata nell'applicazione	5
3	Diagramma dei package - sistema CLIPS	7
4	Componente di	8
5	Componente di::component	8
6	Componente di::module	9
7	Componente model	10
8	Componente model::beacon	11
9	Componente model::compass	12
10	Componente model::dataaccess	12
11	Componente model::dataaccess::dao	13
12	Componente model::dataaccess::service	15
13	Componente model::navigator	16
14	Componente model::navigator::algorithm	17
15	Componente model::navigator::graph	18
16	Componente model::navigator::graph::area	19
17	Componente model::navigator::graph::edge	19
18	Componente model::navigator::graph::navigationinformation .	20
19	Componente model::navigator::graph::vertex	21
20	Componente model::usersetting	21
21	Componente presenter	22
22	Componente view	24
23	Interfaccia InfoComponent	26
24	Classe AppModule	30
25	Classe DatabaseModule	31
26	Classe InfoModule	33
27	Classe NavModule	34
28	Classe SettingModule	35
29	Classe astratta AbsBeaconReceiverManager	36
30	Interfaccia InformationListener	39
31	Interfaccia InformationManager	41
32	Classe InformationManagerImp	44
33	Interfaccia Listener	48
34	Classe MessageSendType	49
35	Interfaccia NavigationListener	50
36	Interfaccia NavigationManager	51
37	Classe NavigationManagerImp	53
38	Classe NoBeaconSeenException	56
39	Classe ServiceConnectionImp	57

40	Classe BeaconManagerAdapter	58
41	Interfaccia BeaconRanger	62
42	Classe LocalBinder	63
43	Interfaccia Logger	64
44	Classe LoggerImp	65
45	Interfaccia MyBeacon	67
46	Classe MyBeaconImp	69
47	Classe MyDistanceCalculator	71
48	Classe PeriodType	72
49	Classe Compass	73
50	Interfaccia CompassListener	76
51	Classe BuildingContract	77
52	Interfaccia BuildingDao	78
53	Classe BuildingTable	80
54	Classe CategoryContract	82
55	Interfaccia CategoryDao	83
56	Classe CategoryTable	85
57	Interfaccia CursorConverter	86
58	Classe DaoFactoryHelper	87
59	Classe EdgeContract	88
60	Interfaccia EdgeDao	89
61	Classe EdgeTable	91
62	Classe EdgeTypeContract	93
63	Interfaccia EdgeTypeDao	94
64	Classe EdgeTypeTable	96
65	Classe MapsDbHelper	97
66	Classe PhotoContract	98
67	Interfaccia PhotoDao	99
68	Classe PhotoTable	101
69	Classe PointOfInterestContract	102
70	Interfaccia PointOfInterestDao	103
71	Classe PointOfInterestTable	105
72	Classe RegionOfInterestContract	107
73	Interfaccia RegionOfInterestDao	108
74	Classe RegionOfInterestTable	110
75	Classe RemoteBuildingDao	111
76	Classe RemoteCategoryDao	112
77	Classe RemoteDaoFactory	113
78	Classe RemoteEdgeDao	115
79	Classe RemoteEdgeTypeDao	116
80	Classe RemotePhotoDao	117

81	Classe RemotePointOfInterestDao	118
82	Classe RemoteRegionOfInterestDao	119
83	Classe RemoteRoiPoiDao	120
84	Classe RoiPoiContract	121
85	Interfaccia RoiPoiDao	122
86	Classe RoiPoiTable	124
87	Classe SQLDao	125
88	Classe SQLiteBuildingDao	128
89	Classe SQLiteCategoryDao	130
90	Classe SQLiteDaoFactory	133
91	Classe SQLiteEdgeDao	134
92	Classe SQLiteEdgeTypeDao	137
93	Classe SQLitePhotoDao	139
94	Classe SQLitePointOfInterestDao	141
95	Classe SQLiteRegionOfInterestDao	143
96	Classe SQLiteRoiPoiDao	145
97	Classe BuildingService	148
98	Interfaccia DatabaseService	152
99	Classe EdgeService	154
100	Classe PhotoService	157
101	Classe PointOfInterestService	159
102	Classe RegionOfInterestService	163
103	Classe ServiceHelper	165
104	Classe BuildingInformation	166
105	Interfaccia BuildingMap	168
106	Classe BuildingMapImp	170
107	Classe NavigationDirection	174
108	Classe astratta NavigationExceptions	175
109	Interfaccia Navigator	176
110	Classe NavigatorImp	178
111	Classe NoGraphSetException	181
112	Classe NoNavigationInformationException	183
113	Classe PathException	184
114	Interfaccia ProcessedInformation	185
115	Classe ProcessedInformationImp	187
116	Classe DijkstraPathFinder	189
117	Interfaccia PathFinder	190
118	Classe MapGraph	191
119	Interfaccia PointOfInterest	193
120	Classe PointOfInterestImp	194
121	Classe PointOfInterestInformation	196

122	Interfaccia RegionOfInterest	198
123	Classe RegionOfInterestImp	200
124	Classe astratta AbsEnrichedEdge	202
125	Classe DefaultEdge	205
126	Interfaccia Edge	207
127	Classe ElevatorEdge	208
128	Interfaccia EnrichedEdge	209
129	Classe StairEdge	211
130	Classe BasicInformation	212
131	Classe DetailedInformation	213
132	Interfaccia NavigationInformation	214
133	Classe NavigationInformationImp	215
134	Classe PhotoInformation	217
135	Classe PhotoRef	218
136	Interfaccia Vertex	219
137	Classe VertexImp	219
138	Classe DeveloperCodeManager	220
139	Classe InstructionPreference	222
140	Classe PathPreference	223
141	Interfaccia Setting	224
142	Classe SettingImp	225
143	Classe BeaconPos	228
144	Classe BeaconPowerAreaActivity	230
145	Classe BeaconPowerPos	232
146	Classe BlankHomeFragment	233
147	Classe CompleteHomeFragment	235
148	Classe DetailedInformationActivity	237
149	Classe DeveloperUnlockerActivity	239
150	Classe HelpActivity	240
151	Classe HomeActivity	242
152	Classe ImageAdapter	246
153	Classe ImageDetailActivity	248
154	Classe ImageDetailFragment	249
155	Classe ImageListFragment	251
156	Classe LocalMapActivity	253
157	Classe LocalMapAdapter	255
158	Classe LoggingActivity	257
159	Classe LogInformationActivity	258
160	Classe MainActivity	259
161	Classe MainDeveloperActivity	260
162	Classe MainDeveloperPresenter	262

163	Classe MyApplication	263
164	Classe NavigationActivity	264
165	Classe NavigationAdapter	267
166	Classe NearbyPoiActivity	269
167	Classe NoInternetAlert	270
168	Classe PoiActivity	272
169	Classe PoiCategoryActivity	273
170	Classe PoiDescriptionActivity	274
171	Classe PreferencesActivity	276
172	Classe RemoteMapManagerActivity	277
173	Classe RemoteMapManagerAdapter	278
174	Classe SearchSuggestionsProvider	280
175	Classe BeaconPowerArea	282
176	Interfaccia BeaconPowerAreaView	284
177	Classe BeaconPowerAreaViewImp	285
178	Interfaccia DescriptionView	287
179	Classe DescriptionViewImp	288
180	Interfaccia DetailedInformationView	289
181	Classe DetailedInformationViewImp	290
182	Interfaccia DeveloperUnlockerView	292
183	Classe DeveloperUnlockerViewImp	293
184	Interfaccia HelpView	294
185	Classe HelpViewImp	295
186	Interfaccia HomeView	296
187	Classe HomeViewImp	298
188	Interfaccia ImageDetailView	300
189	Classe ImageDetailViewImp	301
190	Interfaccia LocalMapManagerView	302
191	Classe LocalMapManagerViewImp	303
192	Interfaccia LoggingView	304
193	Classe LoggingViewImp	305
194	Interfaccia LogInformationView	306
195	Classe LogInformationViewImp	307
196	Interfaccia MainDeveloperView	309
197	Classe MainDeveloperViewImp	310
198	Interfaccia NavigationView	311
199	Classe NavigationViewImp	313
200	Interfaccia NearbyPoiView	315
201	Classe NearbyPoiViewImp	315
202	Interfaccia PoiCategoryView	317
203	Classe PoiCategoryViewImp	318

204	Interfaccia PoiView	319
205	Classe PoiViewImp	320
206	Interfaccia PreferencesView	321
207	Classe PreferencesViewImp	322
208	Interfaccia RemoteMapManagerView	323
209	Classe RemoteMapManagerViewImp	324
210	Schema ER - base di dati	326
211	Diagramma di sequenza - Avvio di un service, per il rilevamento beacon	328
212	Diagramma di sequenza - Elaborazione beacon rilevati e comunicazione broadcast	330
213	Diagramma di sequenza - Avvio navigazione	331
214	Diagramma di sequenza - Avvio della bussola	332
215	Diagramma di sequenza - Caricamento della mappa dal database	333
216	Diagramma di sequenza - Costruzione di EdgeTable da Json .	334
217	Diagramma di sequenza - Costruzione di RoiPoiTable da Json	336
218	Diagramma di sequenza - Modifica delle preferenze	337

1 Introduzione

1.1 Scopo del documento

Questo documento definisce nel dettaglio la struttura e le relazioni tra le parti del prodotto_g, 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 v5.00* .

1.2 Scopo del prodotto

Lo scopo del prodotto_g è implementare un metodo di navigazione indoor_g che sia funzionale alla tecnologia Bluetooth Low Energy (BLE_g). Il prodotto_g comprenderà un prototipo software_g che permetta la navigazione all'interno di un'area predefinita, basandosi sui concetti di Indoor Positioning System (IPS_g) e smart place_g.

1.3 Glossario

Allo scopo di rendere più semplice e chiara la comprensione dei documenti viene allegato il *Glossario v5.00* 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 _g.

1.4 Riferimenti utili

1.4.1 Riferimenti normativi

- capitolato d'appalto C2: CLIPS_g : Communication & Localization with Indoor Positioning Systems: <http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C2.pdf>;
- *Norme di progetto v5.00*.

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_g - Diagrammi delle classi: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E03.pdf>;
- Materiale di riferimento del corso di Ingegneria del Software_g - Model View Presenter: 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_g - Layer Architecture: 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 v5.00*.

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 v5.00*.

2.3 Standard di programmazione

Per gli standard di programmazione si fa riferimento al documento *Norme di progetto v5.00*.

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 v5.00*.

3 Architettura applicazione

3.1 Architettura ad alto livello

L’architettura dell’applicazione rispecchia il pattern architettonico a livelli presentando cinque 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;

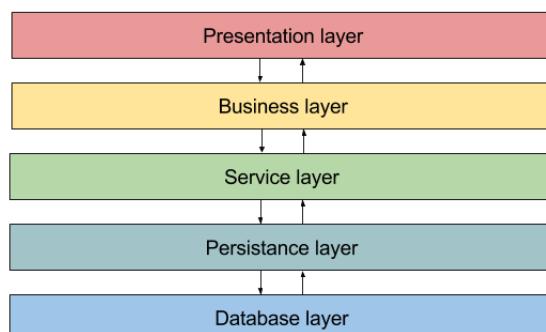


Figura 1: Architettura a layer

3.2 Implementazione nell'applicativo

Seguendo il pattern presentato in precedenza si mostra nel diagramma in figura 2 quali package che compongono l'applicazione corrispondono ai layer prima presentati.

Presentation layer: comprende i package `view` e `presenter`;

Business layer: comprende i package interni al package `model` ossia `navigator`, `compass`, `usersetting` e `compass`;

Service layer: comprende il package `service` contenuto nel package `dataaccess`;

Persistence layer: comprende il package `dao` contenuto nel package `dataaccess`;

Database layer: è il database SQLite.

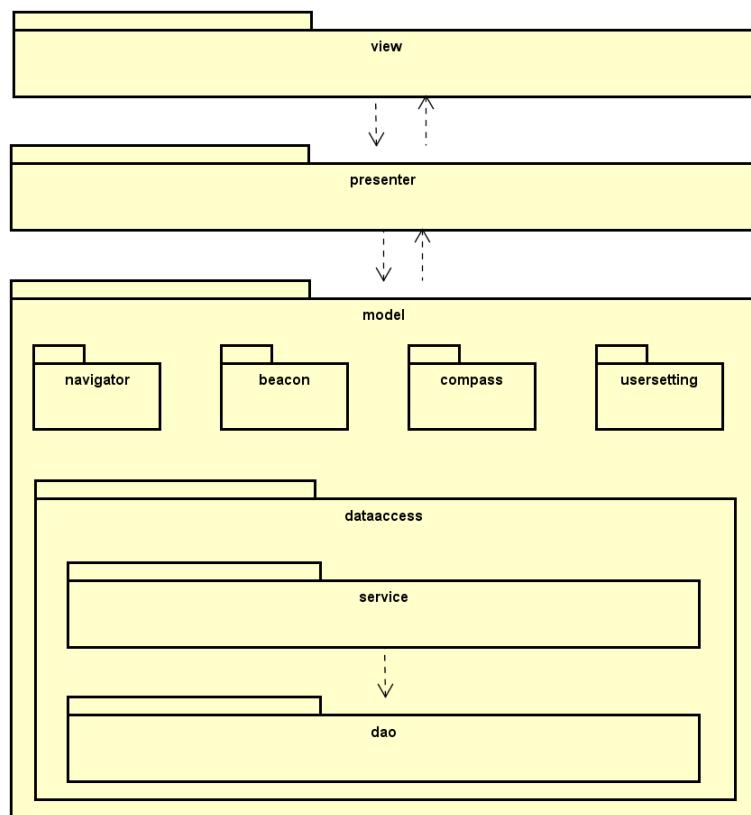


Figura 2: Architettura a layer implementata nell'applicazione

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.

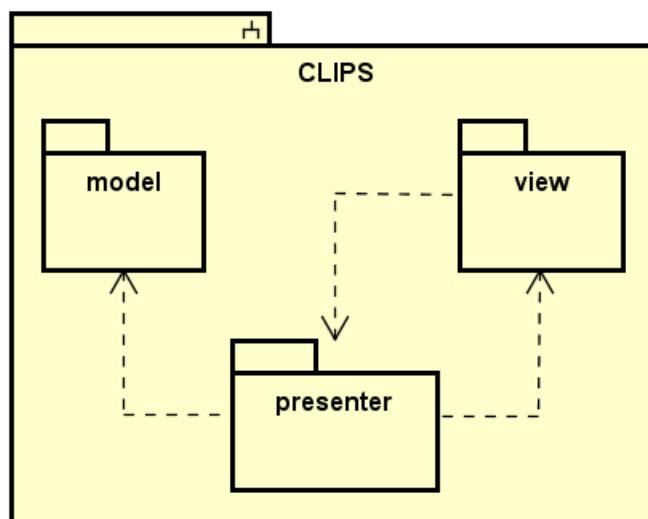


Figura 3: Diagramma dei package - sistema CLIPS

4.3 Componenti

4.3.1 di

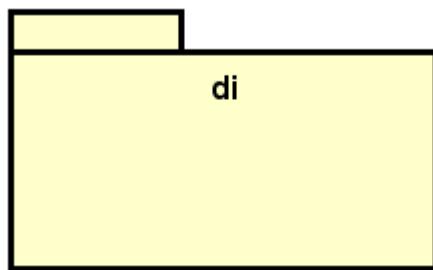


Figura 4: Componente di

- **Descrizione:** Questo package ha il compito di raggruppare tutti i componenti che permettono di gestire la dependency injection;
- **Package Contenuti:**
 - component;
 - module.

4.3.2 di::component

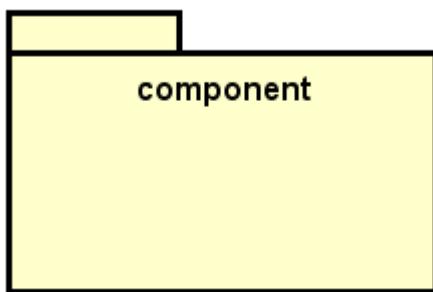


Figura 5: Componente di::component

- **Descrizione:** Questo package raggruppa le interfacce che permettono di eseguire la dependency injection;
- **Padre:** di;

- **Interfacce Contenute:**

- InfoComponent.

4.3.3 di::module

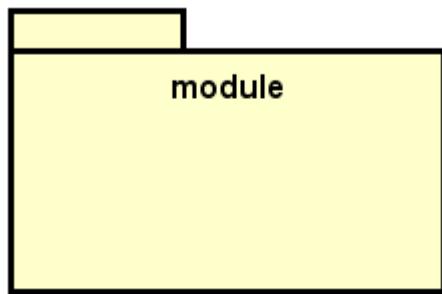


Figura 6: Componente di::module

- **Descrizione:** Questo package raggruppa le classi che permettono di risolvere le dipendenze tra gli oggetti presenti nell'applicazione. Le classi presenti in questo package hanno inoltre il compito di definire la cardinalità delle istanze di un oggetto;
- **Padre:** di;
- **Classi Contenute:**
 - AppModule;
 - DatabaseModule;
 - InfoModule;
 - NavModule;
 - SettingModule.

4.3.4 model

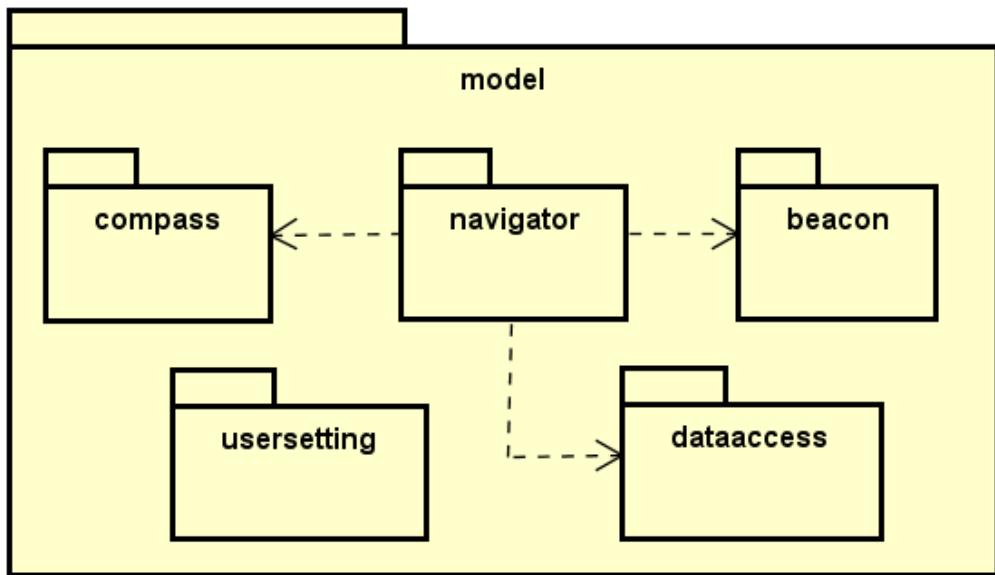


Figura 7: Componente model

- **Descrizione:** Package per il componente Model del pattern architettonale 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:**

- InformationListener;
- InformationManager;
- Listener;
- NavigationListener;
- NavigationManager.

4.3.5 model::beacon

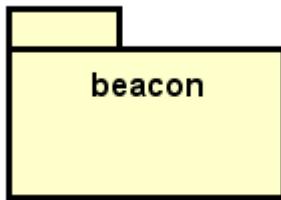


Figura 8: Componente model::beacon

- **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;
 - PeriodType.
- **Interfacce Contenute:**
 - BeaconRanger;
 - Logger;
 - MyBeacon.

4.3.6 model::compass

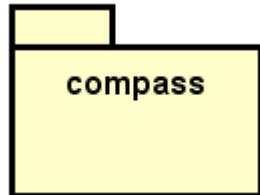


Figura 9: Componente model::compass

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

4.3.7 model::dataaccess

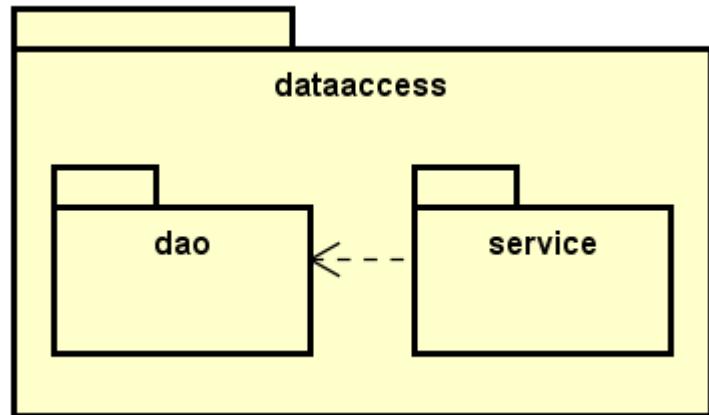


Figura 10: Componente model::dataaccess

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

- **Padre:** model;
- **Package Contenuti:**
 - dao;
 - service.

4.3.8 model::dataaccess::dao

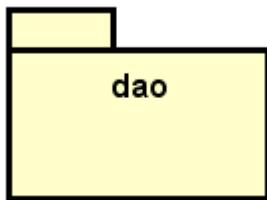


Figura 11: Componente model::dataaccess::dao

- **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;
 - 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.

4.3.9 model::dataaccess::service

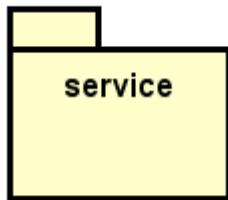


Figura 12: Componente model::dataaccess::service

- **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.

4.3.10 model::navigator

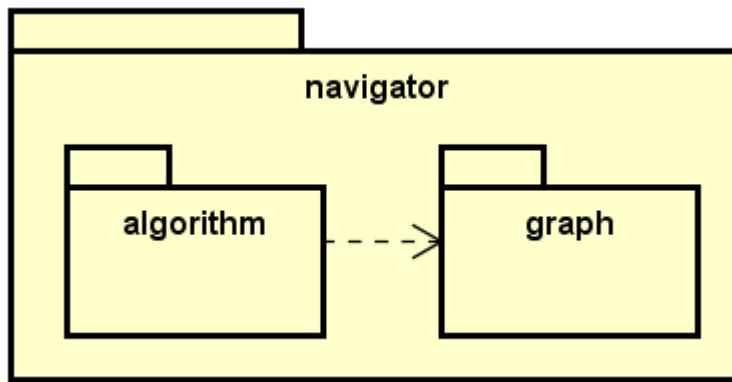


Figura 13: Componente model::navigator

- **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;
 - NavigationDirection;
 - NavigationExceptions;
 - NavigatorImp;
 - NoGraphSetException;
 - NoNavigationInformationException;
 - PathException;
 - ProcessedInformationImp.
- **Interfacce Contenute:**

- BuildingMap;
- Navigator;
- ProcessedInformation.

4.3.11 model::navigator::algorithm

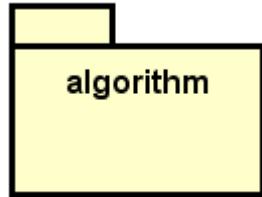


Figura 14: Componente model::navigator::algorithm

- **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.

4.3.12 model::navigator::graph

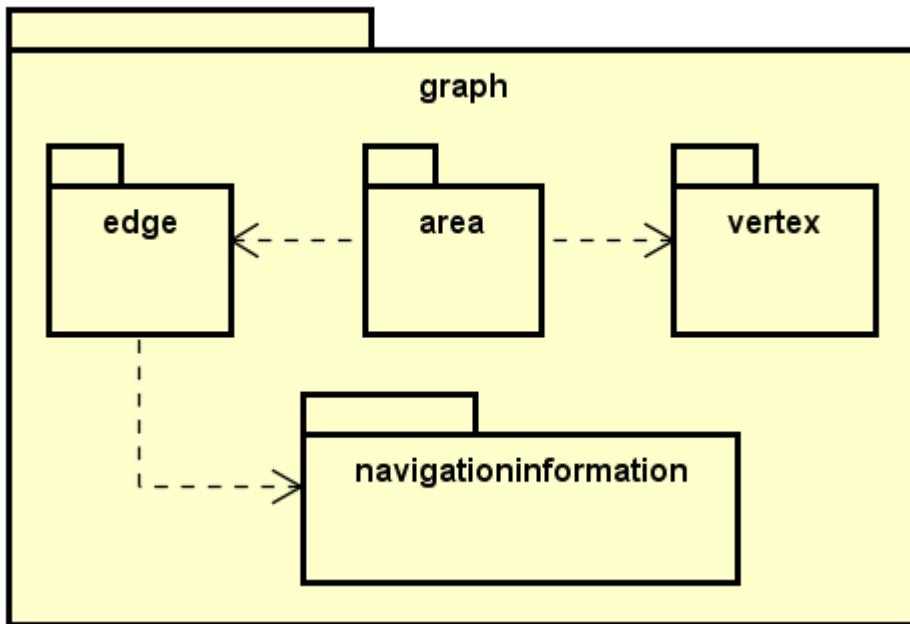


Figura 15: Componente model::navigator::graph

- **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`.

4.3.13 model::navigator::graph::area

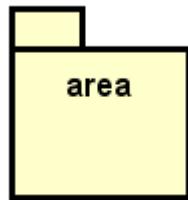


Figura 16: Componente model::navigator::graph::area

- **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.

4.3.14 model::navigator::graph::edge

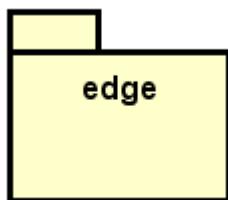


Figura 17: Componente model::navigator::graph::edge

- **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.

4.3.15 model::navigator::graph::navigationinformation



Figura 18: 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.

4.3.16 model::navigator::graph::vertex

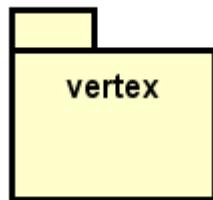


Figura 19: Componente model::navigator::graph::vertex

- **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.

4.3.17 model::usersetting

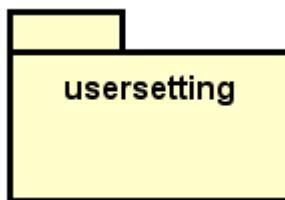


Figura 20: Componente model::usersetting

- **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.

4.3.18 presenter

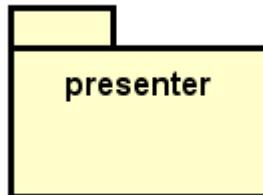


Figura 21: Componente presenter

- **Descrizione:** Package per il componente Presenter del pattern architetture MVP;
- **Classi Contenute:**
 - BeaconPos;
 - BeaconPowerAreaActivity;
 - BeaconPowerPos;
 - BlankHomeFragment;
 - CompleteHomeFragment;
 - DetailedInformationActivity;

- DeveloperUnlockerActivity;
- HelpActivity;
- HomeActivity;
- ImageAdapter;
- ImageDetailActivity;
- ImageDetailFragment;
- ImageListFragment;
- LocalMapActivity;
- LocalMapAdapter;
- LoggingActivity;
- LogInformationActivity;
- MainActivity;
- MainDeveloperActivity;
- MainDeveloperPresenter;
- MyApplication;
- NavigationActivity;
- NavigationAdapter;
- NearbyPoiActivity;
- NoInternetAlert;
- PoiActivity;
- PoiCategoryActivity;
- PoiDescriptionActivity;
- PreferencesActivity;
- RemoteMapManagerActivity;
- RemoteMapManagerAdapter;
- SearchSuggestionsProvider.

4.3.19 view

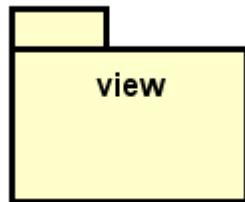


Figura 22: Componente view

- **Descrizione:** Package per il componente View del pattern architetturel MVP. Questo pacchetto contiene tutte le classi che compongono l'application logic;
- **Classi Contenute:**
 - BeaconPowerArea;
 - BeaconPowerAreaViewImp;
 - DescriptionViewImp;
 - DetailedInformationViewImp;
 - DeveloperUnlockerViewImp;
 - HelpViewImp;
 - HomeViewImp;
 - ImageDetailViewImp;
 - LocalMapManagerViewImp;
 - LoggingViewImp;
 - LogInformationViewImp;
 - MainDeveloperViewImp;
 - NavigationViewImp;
 - NearbyPoiViewImp;
 - PoiCategoryViewImp;
 - PoiViewImp;
 - PreferencesViewImp;
 - RemoteMapManagerViewImp.

- **Interfacce Contenute:**

- BeaconPowerAreaView;
- DescriptionView;
- DetailedInformationView;
- DeveloperUnlockerView;
- HelpView;
- HomeView;
- ImageDetailView;
- LocalMapManagerView;
- LoggingView;
- LogInformationView;
- MainDeveloperView;
- NavigationView;
- NearbyPoiView;
- PoiCategoryView;
- PoiView;
- PreferencesView;
- RemoteMapManagerView.

4.4 Classi

4.4.1 di

4.4.1.1 di::component::InfoComponent

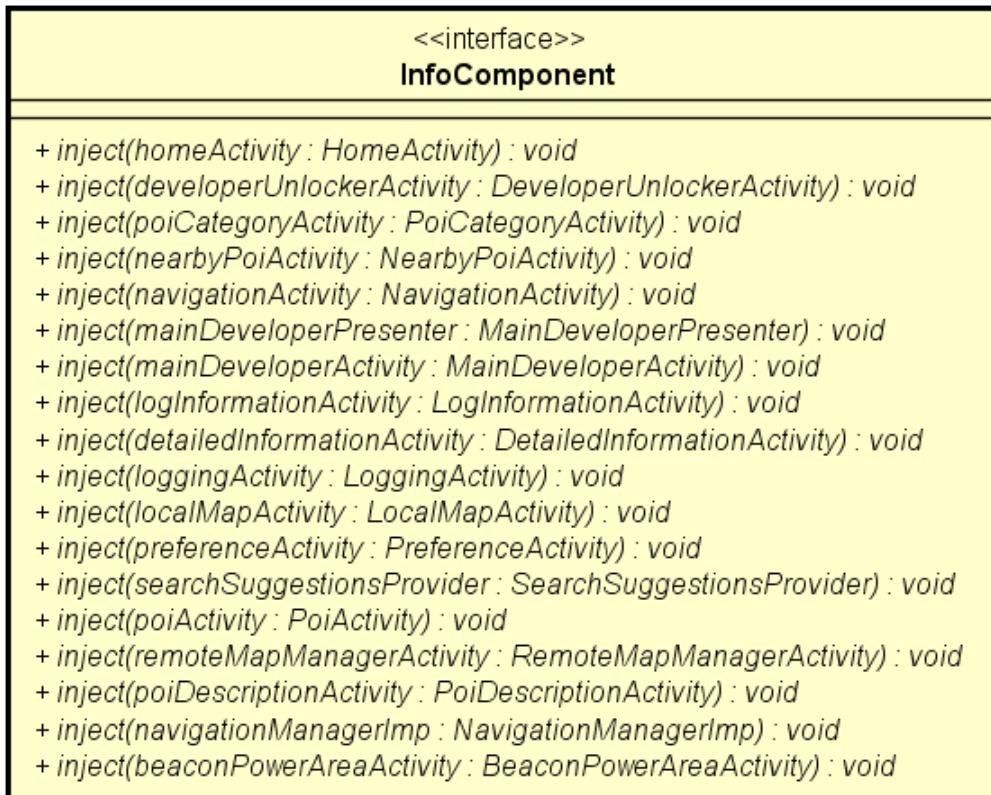


Figura 23: Interfaccia InfoComponent

Nome: InfoComponent;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Interfaccia che viene utilizzata da Dagger 2 per eseguire le iniezioni;

Descrizione: Interfaccia che viene implementata in autonomia da Dagger2 nella quale devono essere dichiarati i metodi inject che richiedono come parametro la classe in cui sono presenti campi dati annotati con Inject. Con l'annotazione Component devono essere dichiarate le classi annotate con l'annotazione Module che permettono di risolvere le dipendenze;

Metodi:

- + inject(homeActivity : HomeActivity) : void

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo HomeActivity

Argomenti:

- homeActivity : HomeActivity

Oggetto in cui devono essere iniettate le dipendenze

- + inject(developerUnlockerActivity : DeveloperUnlockerActivity) : void

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo DeveloperUnlockerActivity

Argomenti:

- developerUnlockerActivity : DeveloperUnlockerActivity

Oggetto in cui devono essere iniettate le dipendenze

- + inject(poiCategoryActivity : PoiCategoryActivity) : void

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo PoiCategoryActivity

Argomenti:

- poiCategoryActivity : PoiCategoryActivity

Oggetto in cui devono essere iniettate le dipendenze

- + inject(nearbyPoiActivity : NearbyPoiActivity) : void

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo NearbyPoiActivity

Argomenti:

- nearbyPoiActivity : NearbyPoiActivity

Oggetto in cui devono essere iniettate le dipendenze

- + inject(navigationActivity : NavigationActivity) : void

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo NavigationActivity

Argomenti:

- navigationActivity : NavigationActivity

Oggetto in cui devono essere iniettate le dipendenze

- + inject(mainDeveloperPresenter : MainDeveloperPresenter) : void

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo MainDeveloperPresenter

Argomenti:

- `mainDeveloperPresenter : MainDeveloperPresenter`
Oggetto in cui devono essere iniettate le dipendenze
- + `inject(mainDeveloperActivity : MainDeveloperActivity) : void`
Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo MainDeveloperActivity

Argomenti:

- `mainDeveloperActivity : MainDeveloperActivity`
Oggetto in cui devono essere iniettate le dipendenze
- + `inject(logInformationActivity : LogInformationActivity) : void`
Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo LogInformationActivity

Argomenti:

- `logInformationActivity : LogInformationActivity`
Oggetto in cui devono essere iniettate le dipendenze
- + `inject(loggingActivity : LoggingActivity) : void`
Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo LoggingActivity

Argomenti:

- `loggingActivity : LoggingActivity`
Oggetto in cui devono essere iniettate le dipendenze
- + `inject(detailedInformationActivity : DetailedInformationActivity) : void`
Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo DetailedInformationActivity

Argomenti:

- `detailedInformationActivity : DetailedInformationActivity`
Oggetto in cui devono essere iniettate le dipendenze
- + `inject(localMapActivity : LocalMapActivity) : void`
Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo LocalMapActivity

Argomenti:

- `localMapActivity : LocalMapActivity`
Oggetto in cui devono essere iniettate le dipendenze

- + *inject(preferenceActivity : PreferencesActivity) : void*

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo PreferencesActivity

Argomenti:

- preferenceActivity : PreferencesActivity
Oggetto in cui devono essere iniettate le dipendenze

- + *inject(searchSuggestionsProvider : SearchSuggestionsProvider) : void*

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo SearchSuggestionsProvider

Argomenti:

- searchSuggestionsProvider : SearchSuggestionsProvider
Oggetto in cui devono essere iniettate le dipendenze

- + *inject(poiActivity : PoiActivity) : void*

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo PoiActivity

Argomenti:

- poiActivity : PoiActivity
Oggetto in cui devono essere iniettate le dipendenze

- + *inject(remoteMapManagerActivity : RemoteMapManagerActivity) : void*

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo RemoteMapManagerActivity

Argomenti:

- remoteMapManagerActivity : RemoteMapManagerActivity
Oggetto in cui devono essere iniettate le dipendenze

- + *inject(poiDescriptionActivity : PoiDescriptionActivity) : void*

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo PoiDescriptionActivity

Argomenti:

- poiDescriptionActivity : PoiDescriptionActivity
Oggetto in cui devono essere iniettate le dipendenze

- + *inject(navigationManagerImp : NavigationManagerImp) : void*

Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo NavigationManagerImp

Argomenti:

- navigationManagerImp : NavigationManagerImp
Oggetto in cui devono essere iniettate le dipendenze
- + *inject(beaconPowerAreaActivity : BeaconPowerAreaActivity)*
: void
Metodo che permette di iniettare i campi annotati con Inject negli oggetti di tipo BeaconPowerAreaActivity

Argomenti:

- beaconPowerAreaActivity : BeaconPowerAreaActivity
Oggetto in cui devono essere iniettate le dipendenze

4.4.1.2 di::module:: AppModule

AppModule
- mApplication : MyApplication {readOnly} - compass : Compass {readOnly}
+ AppModule(myApplication : MyApplication) + providesContext() : Context + providesApplication() : Application + providesCompass() : Compass

Figura 24: Classe AppModule

Nome: AppModule;

Tipo: Classe;

Visibilità: public;

Utilizzo: Viene utilizzata dalle classi implementate automaticamente da Dagger2 per rendere il Context e l'Application Singleton e risolvere le dipendenze;

Descrizione: Classe nella quale vengono dichiarate le dipendenze verso oggetti di tipo Context e Application;

Attributi:

- - `compass : Compass {readOnly}`
Riferimento ad un oggetto Compass per la lettura dei dati della bussola
- - `mApplication : MyApplication {readOnly}`
Applicazione in esecuzione nella quale ci sono oggetti in cui fare la dependency injection

Metodi:

- + `AppModule(application : MyApplication)`
Costruttore della classe AppModule

Argomenti:

- `application : MyApplication`
Applicazione in esecuzione nella quale ci sono oggetti in cui fare la dependency injection

- + `providesApplication() : Application`

Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo Application. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo

- + `providesCompass() : Compass`

Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo Compass. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo.

- + `providesContext() : Context`

Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo Context. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo

4.4.1.3 di::module::DatabaseModule

DatabaseModule
- <code>remoteURL : String {readOnly}</code>
+ <code>DatabaseModule(remoteURL : String)</code>
+ <code>provideSQLiteDaoFactory(context : Context) : SQLiteDaoFactory</code>
+ <code>provideRemoteDaoFactory() : RemoteDaoFactory</code>
+ <code>providesDatabaseService(sqliteDaoFactory : SQLiteDaoFactory, remoteDaoFactory : RemoteDaoFactory) : DatabaseService</code>

Figura 25: Classe DatabaseModule

Nome: DatabaseModule;

Tipo: Classe;

Visibilità: public;

Utilizzo: Viene utilizzata dalle classi implementate automaticamente da Dagger2 per rendere DatabaseService Singleton e risolvere le dipendenze;

Descrizione: Classe nella quale vengono dichiarate le dipendenze verso oggetti di tipo SQLiteDaoFactory, RemoteDaoFactory e DatabaseService;

Attributi:

- - **remoteURL** : String
Stringa che rappresenta l'URL del database remoto

Metodi:

- + **DatabaseModule(remoteURL : String)**
Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo SQLiteDaoFactory. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo

Argomenti:

- **remoteURL** : String
Stringa che rappresenta l'URL del database remoto

- + **provideRemoteDaoFactory() : RemoteDaoFactory**
Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo DatabaseService ritornando un oggetto di tipo BuildingServiceImp. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo
- + **provideSQLiteDaoFactory(context : Context) : SQLiteDaoFactory**
Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo SQLiteDaoFactory. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo

Argomenti:

- **context** : Context
Contesto di esecuzione dell'applicazione
- + **providesDatabaseService(sqliteDaoFactory : SQLiteDaoFactory, remoteDaoFactory : RemoteDaoFactory) : DatabaseService**
Metodo che permette di risolvere le dipendenze verso campi dati

annotati con Inject e di tipo DatabaseService ritornando un oggetto di tipo BuildingServiceImp. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo

Argomenti:

- **sqLiteDaoFactory** : `SQLiteDaoFactory`
Factory che permette di costruire oggetti con cui accedere ai dati salvati nel database locale
- **remoteDaoFactory** : `RemoteDaoFactory`
Factory che permette di costruire oggetti scaricati dal database remoto

4.4.1.4 di::module::InfoModule

InfoModule
+ providesInformationManager(service : DatabaseService, context : Context) : void

Figura 26: Classe InfoModule

Nome: InfoModule;

Tipo: Classe;

Visibilità: public;

Utilizzo: Viene utilizzata dalle classi implementate automaticamente da Dagger2 per rendere la classe DatabaseService Singleton e risolvere le dipendenze;

Descrizione: Classe nella quale vengono dichiarate le dipendenze verso oggetti di tipo InformationManager;

Metodi:

- + `providesInformationManager(service : DatabaseService, context : Context) : InformationManager`
Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo InformationManager ritornando un oggetto di tipo InformationManagerImp. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo

Argomenti:

- **service** : DatabaseService
Oggetto che permette di accedere al database sia locale che remoto
- **context** : Context
Contesto di esecuzione dell'applicazione

4.4.1.5 di::module::NavModule

NavModule
+ provideMapGraph(informationManager : InformationManager) : MapGraph
+ providesNavigationManager(mapGraph : MapGraph, context : Context) : NavigationManager

Figura 27: Classe NavModule

Nome: NavModule;

Tipo: Classe;

Visibilità: public;

Utilizzo: Viene utilizzata dalle classi implementate automaticamente da Dagger2 per risolvere le dipendenze verso NavigationManager e MapGraph;

Descrizione: Classe nella quale vengono dichiarate le dipendenze verso oggetti di tipo NavigationManager;

Metodi:

- + **provideMapGraph(informationManager : InformationManager) : MapGraph**
Metodo che permette di risolvere le dipendenze verso gli oggetti della classe MapGraph

Argomenti:

- **informationManager : InformationManager**
Riferimento all'oggetto che gestisce le informazioni provenienti dal database

- + **providesNavigationManager(mapGraph : MapGraph, context : Context) : NavigationManager**
Metodo che permette di risolvere le dipendenze verso campi dati

annotati con Inject e di tipo NavigationManager ritornando un oggetto di tipo NavigationManagerImp. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo

Argomenti:

- mapGraph : MapGraph
Oggetto che rappresenta il grafo tramite la classe MapGraph
- context : Context
Contesto di esecuzione dell'applicazione

4.4.1.6 di::module::SettingModule

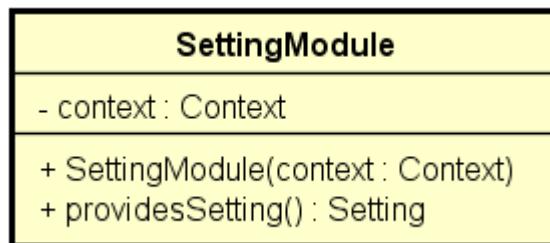


Figura 28: Classe SettingModule

Nome: SettingModule;

Tipo: Classe;

Visibilità: public;

Utilizzo: Viene utilizzata dalle classi implementate automaticamente da Dagger2 per rendere la classe Setting Singleton e risolvere le dipendenze;

Descrizione: Classe nella quale vengono dichiarate le dipendenze verso oggetti di tipo Setting;

Attributi:

- - context : Context {readOnly}
Contesto di esecuzione dell'applicazione

Metodi:

- + **SettingModule(context : Context)**

Costruttore della classe SettingModule

Argomenti:

- context : Context

Contesto di esecuzione dell'applicazione

- + **providesSetting() : Setting**

Metodo che permette di risolvere le dipendenze verso campi dati annotati con Inject e di tipo Setting ritornando un oggetto di tipo SettingImp. L'istanza ritornata sarà sempre la stessa utilizzando lo stesso modulo.

4.4.2 model

4.4.2.1 model::AbsBeaconReceiverManager

AbsBeaconReceiverManager	
- context : Context {readOnly}	
- serviceStart : Intent {readOnly}	
- beaconManagerAdapter : BeaconRanger	
- serviceConnection : ServiceConnection	
- isBound : boolean	
# listeners : Collection	
+ AbsBeaconReceiverManager(context : Context)	
+ modifyScanningPeriod(p : long, backOrFore : boolean, betweenOrNot : boolean) : void	
+ startService() : void	
+ stopService() : void	
+ onReceive(context : Context, intent : Intent) : void	
# getContext() : Context	
# addListener(listener : Listener) : void	
# removeListener(listener : Listener) : void	

Figura 29: Classe astratta AbsBeaconReceiverManager

Nome: *AbsBeaconReceiverManager*;

Tipo: Classe astratta;

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`
Service che si occupa del rilevamento dei beacon
- - `context` : `Context {readOnly}`
Contesto dell'applicazione
- - `isBound` : `boolean`
Variabile booleana che indica l'avvenuta connessione tra il Service e la ServiceConnection
- # `listeners` : `Collection<Listener>`
Collezione contenenti tutti I listener dell'oggetto
- - `serviceConnection` : `ServiceConnection`
Connessione con il Service per la comunicazione con il service stesso
- - `serviceStart` : `Intent {readOnly}`
Intent per ricevere I beacon inviati dal BeaconManagerAdapter

Metodi:

- + `AbsBeaconReceiverManager(context : Context)`
Costruttore della classe AbsBeaconReceiverManager

Argomenti:

- `context` : `Context`
Contesto dell'applicazione

- # `addListener(listener : Listener) : void`
Metodo che permette di registrare come listener un oggetto che implementa l'interfaccia Listener

Argomenti:

- `listener` : `Listener`
Listener che deve essere aggiunto alla lista dei listener registrati

- # `getContext() : Context`
Metodo che ritorna il contesto dell'applicazione

- + `modifyScanningPeriod(p : long, type : periodType) : 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
- `type : periodType`
Parametro per decidere se cambiare il periodo di scansione in Foreground o in Background, di scansione o di non scansione

- + `onReceive(context : Context, intent : Intent) : void`

Metodo astratto che permette di eseguire delle azioni alla ricezione di una PriorityQueue<MyBeacon> contenente l'insieme dei beacon visibili in un determinato momento

Argomenti:

- `context : Context`
Contesto dell'applicazione
- `intent : Intent`
Intent contenente le informazioni del messaggio

- # `removeListener(listener : Listener) : void`

Metodo che permette la rimozione di un listener precedentemente registrato

Argomenti:

- `listener : Listener`
Listener da rimuovere dalla lista dei listener registrati

- + `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

4.4.2.2 model::InformationListener

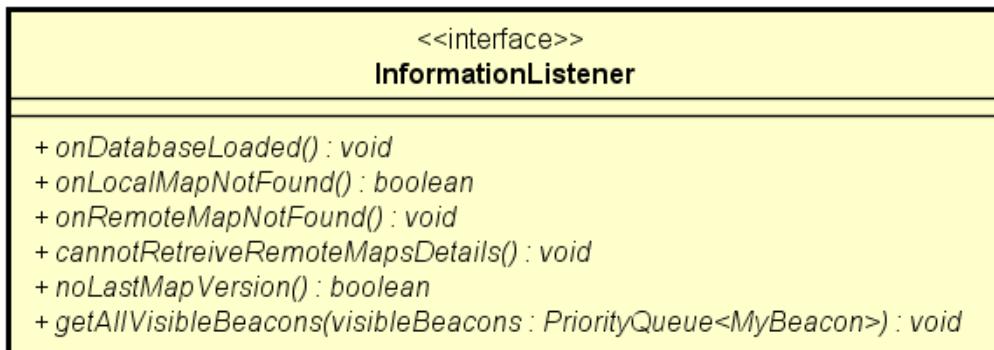


Figura 30: Interfaccia InformationListener

Nome: InformationListener;

Tipo: Interfaccia;

Implementa:

- Listener.

Visibilità: public;

Utilizzo: È utilizzata per esporre un insieme di metodi che devono essere implementati per utilizzare correttamente le classi che implementano l'interfaccia InformationManager, in particolare per la gestione dei problemi nel recupero delle informazioni;

Descrizione: Interfaccia che espone i metodi che devono essere implementati dalle classi che vogliono registrarsi come listener delle classi che implementano l'interfaccia InformationManager;

Metodi:

- + cannotRetrieiveRemoteMapsDetails() : void
Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui non sia possibile recuperare le informazioni riguardanti le mappe dal database remoto
- + getAllVisibleBeacons(visibleBeacons : PriorityQueue<MyBeacon>) : void
Metodo che viene invocato ogni volta che vengono rilevati dei beacon, passando la coda di beacon trovati

Argomenti:

- **visibleBeacons** : `PriorityQueue<MyBeacon>`
Coda dei beacon rilevati ordinata per potenza
- + **noLastMapVersion()** : `boolean`
Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui la versione della mappa locale differisca dalla versione della mappa in remoto
- + **onDatabaseLoaded()** : `void`
Metodo che viene invocato quando il database è stato caricato dalla classe in cui l'oggetto si è registrato come listener
- + **onLocalMapNotFound()** : `boolean`
Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui non sia stata trovata una mappa nel database locale
- + **onRemoteMapNotFound()** : `void`
Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui non sia stata trovata un'acerta mappa nel database remoto

4.4.2.3 model::InformationManager

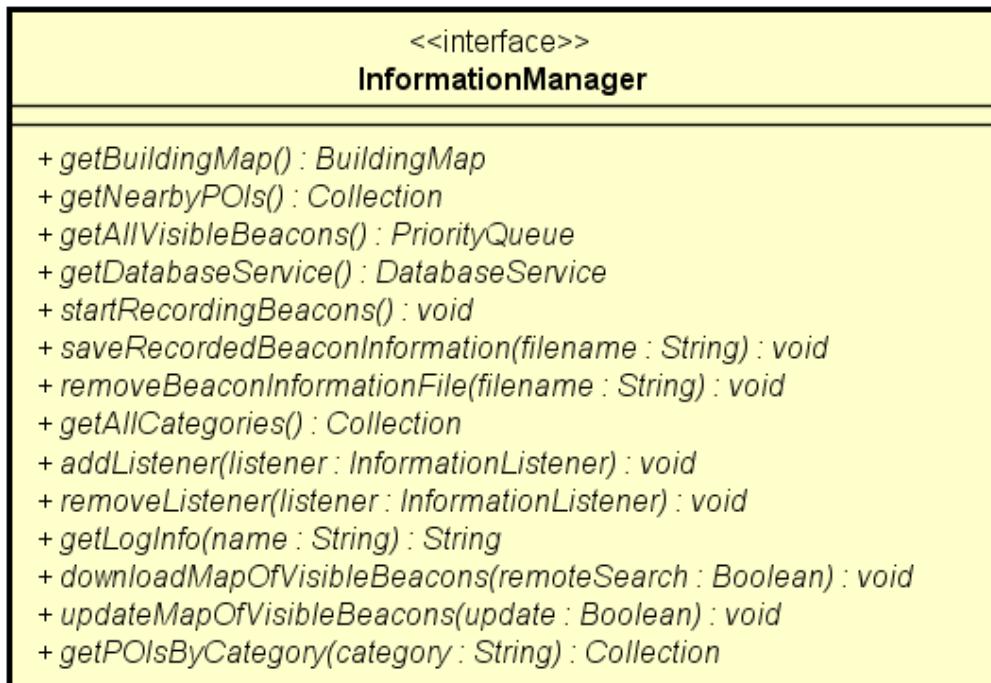


Figura 31: Interfaccia InformationManager

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:

- + *addListener(listener : InformationListener) : void*
Metodo che permette di registrare un listener

Argomenti:

- *listener : InformationListener*
Listener che deve essere aggiunto alla lista di Information-Listener

- + *downloadMapOfVisibleBeacons(remoteSearch : Boolean) : void*

Metodo che permette di scaricare la mappa associata ai beacon visibili

Argomenti:

- *remoteSearch : Boolean*

Booleano che indica se scaricare la mappa associata ai beacon visibili oppure no

- + *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

- + *getPOIsByCategory(category : String) : Collection<PointOfInterest>*

Metodo che ritorna tutti i PointOfInterest appartenenti ad una certa categoria

Argomenti:

- **category** : `String`

Nome della categoria di cui si vogliono recuperare tutti i PointOfInterest

- + `removeBeaconInformationFile(filename : String) : void`
Metodo che permette di rimuovere un log delle informazioni dei beacon visibili

Argomenti:

- **filename** : `String`

Nome del file da rimuovere

- + `removeListener(listener : InformationListener) : void`
Metodo che permette di rimuovere un listener

Argomenti:

- **listener** : `InformationListener`

Listener che deve essere rimosso dalla lista di InformationListener

- + `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

- + `updateMapOfVisibleBeacons(update : Boolean) : void`
Metodo che permette di aggiornare la mappa associata ai beacon visibili

Argomenti:

- **update** : `Boolean`

Booleano che indica se aggiornare la mappa associata ai beacon visibili oppure no

4.4.2.4 model::InformationManagerImp

InformationManagerImp	
- map : BuildingMap	
- lastBeaconsSeen : PriorityQueue	
- activeLog : Logger	
- dbService : DatabaseService	
- shouldLog : boolean	
+ InformationManagerImp(dbService : DatabaseService, context : Context)	
+ getBuildingMap() : BuildingMap	
+ getNearbyPOIs() : Collection	
+ getAllVisibleBeacons() : PriorityQueue	
+ getDatabaseService() : DatabaseService	
+ startRecordingBeacons() : void	
+ saveRecordedBeaconInformation(filename : String) : void	
+ removeBeaconInformationFile(filename : String) : void	
+ onReceive(context : Context, intent : Intent) : void	
- setVisibleBeacon(beacons : PriorityQueue) : void	
- loadMap() : void	
- isDeveloper() : boolean	
+ getAllCategories() : Collection	
+ getLoginInfo(name : String) : String	
+ addListener(listener : InformationListener) : void	
+ removeListener(listener : InformationListener) : void	
+ downloadMapOfVisibleBeacons(remoteSearch : Boolean) : void	
+ updateMapOfVisibleBeacons(update : Boolean) : void	
+ getPOIsByCategory(category : String) : Collection	

Figura 32: 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
- - `proc` : `boolean`
Variabile che rappresenta se è già in download una mappa oppure no
- - `shouldLog` : `boolean`
Variabile booleana che indica se debba essere effettuato o meno il log dei beacon rilevati

Metodi:

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

- + `addListener(listener : InformationListener) : void`
Override Metodo che permette di registrare un listener

Argomenti:

- **listener** : `InformationListener`
Listener che deve essere aggiunto alla lista di Information-Listener
- + `downloadMapOfVisibleBeacons(remoteSearch : Boolean) : void`

Override Metodo che permette di scaricare la mappa associata ai beacon visibili

Argomenti:

- `remoteSearch : Boolean`
Booleano che indica se scaricare la mappa associata ai beacon visibili oppure no
- + `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

- + `getPOIsByCategory(category : String) : Collection<PointOfInterest>`
Override Metodo che ritorna tutti i PointOfInterest appartenenti ad una certa categoria

Argomenti:

- `category : String`

Nome della categoria di cui si vogliono recuperare tutti i PointOfInterest

- - `isDeveloper() : boolean`

Metodo che permette di verificare se un utente è sviluppatore oppure no

- - `loadMap() : void`

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

- + `onReceive(context : Context, intent : Intent) : 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

Argomenti:

- `context : Context`

Contesto dell'applicazione

- `intent : Intent`

Intent contenente le informazioni del messaggio

- + `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

- + `removeListener(listener : InformationListener) : void`

Override Metodo che permette di rimuovere un listener

Argomenti:

- `listener : InformationListener`

Listener che deve essere rimosso dalla lista di InformationListener

- + 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

- + updateMapOfVisibleBeacons(update : Boolean) : void

Override Metodo che permette di aggiornare la mappa associata ai beacon visibili

Argomenti:

- update : Boolean

Booleano che indica se aggiornare la mappa associata ai beacon visibili oppure no

4.4.2.5 model::Listener



Figura 33: Interfaccia Listener

Nome: Listener;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata come interfaccia base per la gerarchia di listener ed inoltre può essere utilizzata per identificare che una classe è listener di altre ;

Descrizione: Interfaccia base per la gerarchia di listener presente nel package model;

4.4.2.6 model::MessageSendType



Figura 34: 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;

4.4.2.7 model::NavigationListener

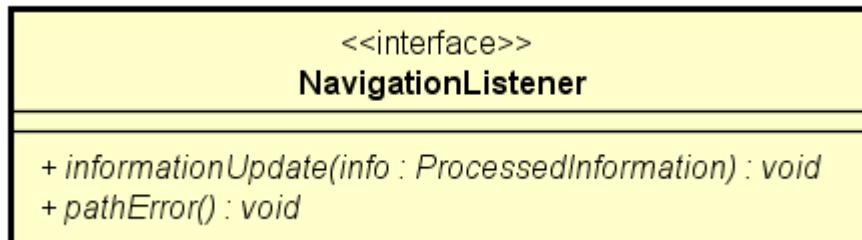


Figura 35: Interfaccia NavigationListener

Nome: NavigationListener;

Tipo: Interfaccia;

Implementa:

- Listener.

Visibilità: public;

Utilizzo: È utilizzata per esporre un insieme di metodi che devono essere implementati dalle classi che vogliono registrarsi come listener delle classi che implementano l'interfaccia NavigationManager, per un corretto funzionamento della navigazione ;

Descrizione: Interfaccia che deve essere implementata da chi vuole usufruire della navigazione;

Metodi:

- `+ informationUpdate(info : ProcessedInformation) : void`
Metodo che viene invocato quando c'è un aggiornamento nelle informazioni di navigazione

Argomenti:

- info : ProcessedInformation
Informazioni aggiornate

- `+ pathError() : void`

Metodo che viene invocato in caso di errori di navigazione

4.4.2.8 model::NavigationManager

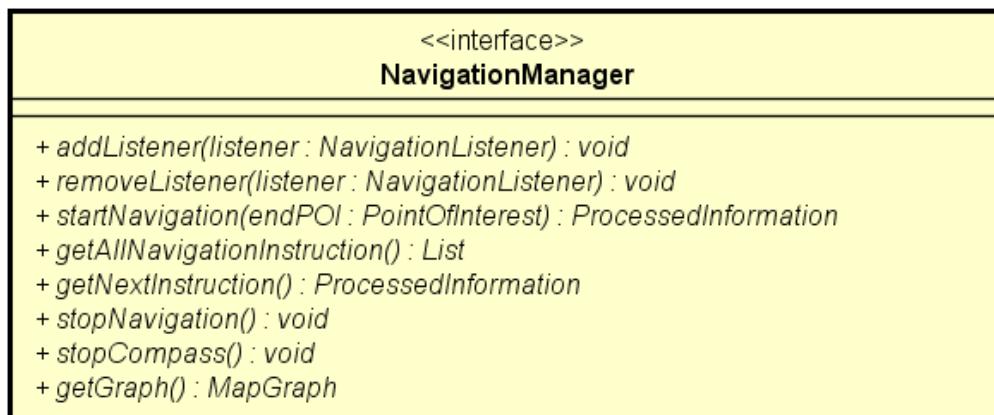


Figura 36: Interfaccia NavigationManager

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:

- + `addListener(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

- + *getGraph() : MapGraph*
Metodo che permette di accedere al grafo che rappresenta l'edificio
- + *getNextInstruction() : ProcessedInformation*
Metodo che permette di recuperare tutte le istruzioni di navigazione per un percorso calcolato in base al beacon più potente recuperato da una PriorityQueue<MyBeacon>. Viene lanciata una eccezione di tipo NoNavigationInformationException nel caso in cui venga richiamato questo metodo senza aver prima avviato la navigazione.
- + *removeListener(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

4.4.2.9 model::NavigationManagerImp

NavigationManagerImp	
- navigator : Navigator	
- graph : MapGraph	
~ compass : Compass	
+ NavigationManagerImp(graph : MapGraph, context : Context)	
+ startNavigation(endPOI : PointOfInterest) : ProcessedInformation	
+ getAllNavigationInstruction() : List	
+ getNextInstruction() : ProcessedInformation	
+ stopNavigation() : void	
+ startCompass() : void	
+ stopCompass() : void	
- setVisibleBeacon(beacons : PriorityQueue) : void	
+ addListener(listener : NavigationListener) : void	
+ removeListener(listener : NavigationListener) : void	
+ onReceive(context : Context, intent : Intent) : void	
+ getGraph() : MapGraph	

Figura 37: 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
Oggetto che permette di recuperare I dati della bussola

- - `graph : MapGraph {readOnly}`
Grafo rappresentante la mappa dell'edificio
- - `navigator : Navigator`
Oggetto per la navigazione

Metodi:

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

Argomenti:

- `graph : MapGraph`
Grafo dell'edificio in cui si desidera navigare
- `context : Context`
Contesto dell'applicazione

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

Argomenti:

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

- + `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

- + `getGraph() : MapGraph`
Override Metodo che permette di accedere al grafo che rappresenta l'edificio

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

- + `onReceive(context : Context, intent : Intent) : void`
Override Metodo che si occupa di settare il campo dati lastBea-

consSeen con la PriorityQueue<MyBeacon> contenente gli ultimi beacon rilevati e di aggiornare tutti I listeners con le ultime istruzioni di navigazione

Argomenti:

- context : Context
Contesto dell'applicazione
- intent : Intent
Intent contenente le informazioni del messaggio

- + removeListener(listener : NavigationListener) : void
Override Metodo che permette di rimuovere un listener

Argomenti:

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

- - setVisibleBeacon(beacons : PriorityQueue<MyBeacon>) : void
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 attivare il rilevamento dei dati dalla bussola
- + startNavigation() : ProcessedInformation
Override Metodo che permette di avviare la navigazione verso uno specifico POI
- + stopCompass() : void
Override Metodo che permette di fermare il rilevamento dei dati ottenuti dalla bussola
- + stopNavigation() : void
Override Metodo che permette di fermare la navigazione

4.4.2.10 model::NoBeaconSeenException

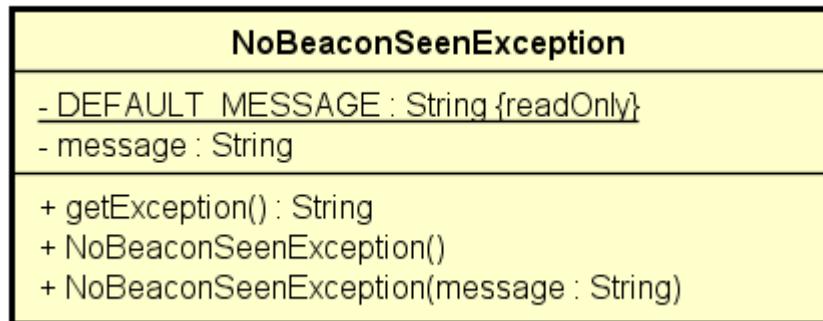


Figura 38: Classe NoBeaconSeenException

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

- + `getException() : String`

Metodo per ottenere un messaggio di spiegazione dell'eccezione

4.4.2.11 model::ServiceConnectionImp

ServiceConnectionImp
+ <code>onServiceConnected(className : ComponentName, service : IBinder) : void</code>
+ <code>onServiceDisconnected(arg0 : ComponentName) : void</code>

Figura 39: 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

4.4.2.12 model::beacon::BeaconManagerAdapter

BeaconManagerAdapter	
- locBinder : IBinder	
- beaconManager : BeaconManager	
- region : Region	
- <u>beaconLayout : String</u>	
- isBackground : boolean	
- periods : Map	
+ onCreate() : void	
+ onDestroy() : void	
+ setBackgroundMode(mode : boolean) : void	
- startRanging() : void	
+ modifyScanPeriod(p : long, type : PeriodType) : void	
+ setRegionExitPeriod(p : long) : void	
+ onBind(intent : int) : IBinder	
- setMonitorNotifier(notifier : MonitorNotifier) : void	
+ didEnterRegion(region : Region) : void	
+ didExitRegion(region : Region) : void	
+ didDeterminateStateForRegion(i : int, region : Region) : void	
+ isBackground() : boolean	
+ getPeriod(type : PeriodType) : long	

Figura 40: 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. Tale campo dati è fissato come statico poichè necessario a tutte le istanze della classe per decifrare in modo eguale le informazioni del segnale dei beacon da riconoscere

- - beaconManager : BeaconManager

Riferimento alla classe che permette di gestire il rilevamento dei beacon

- - isBackground : boolean

Indica se il BeaconManager è in background o meno

- - locBinder : IBinder

Riferimento al LocalBinder che detiene il collegamento con il Service

- - periods : Map<PeriodType, Long>

Insieme dei periodi di scan del BeaconManager

- - 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

- + getPeriod(type : PeriodType) : long

Override Metodo per ottenere uno tra i periodi di scan del BeaconManager in base alla richiesta

Argomenti:

- type : PeriodType
 Tipo di periodo di scan desiderato

- + isBackground() : boolean

Override Metodo che ritorna un valore booleano che indica se il BeaconManager è in background

- + modifyScanPeriod(p : long, type : periodType) : void
 Metodo che serve a modificare il periodo di scansione per il rilevamento dei beacon

Argomenti:

- p : long
 Periodo di scansione

- **type : periodType**

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

4.4.2.13 model::beacon::BeaconRanger

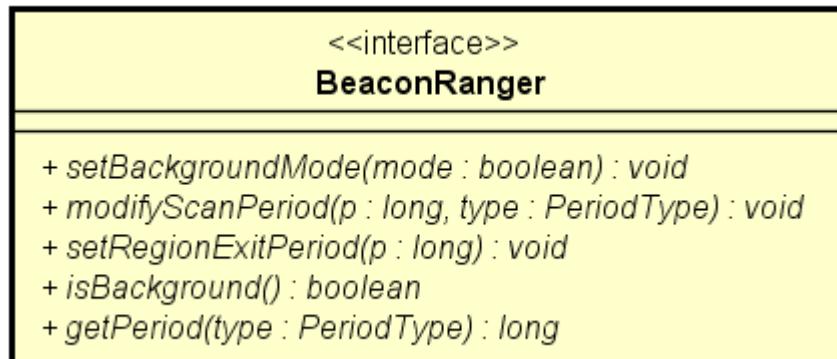


Figura 41: Interfaccia BeaconRanger

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:

- `+ getPeriod(type : PeriodType) : long`

Metodo per ottenere uno tra i periodi di scan del BeaconManager in base alla richiesta

Argomenti:

- `type : PeriodType`

Tipo di periodo di scan desiderato

- `+ isBackground() : boolean`

Metodo che ritorna un valore booleano che indica se il BeaconManager è in background

- `+ modifyScanPeriod(p : long, type : periodType) : void`

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

Argomenti:

- p : long
Periodo di scansione
- type : periodType
Parametro per decidere se cambiare il periodo di scansione in Foreground o in Background, 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

4.4.2.14 model::beacon::LocalBinder



Figura 42: 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

4.4.2.15 model::beacon::Logger

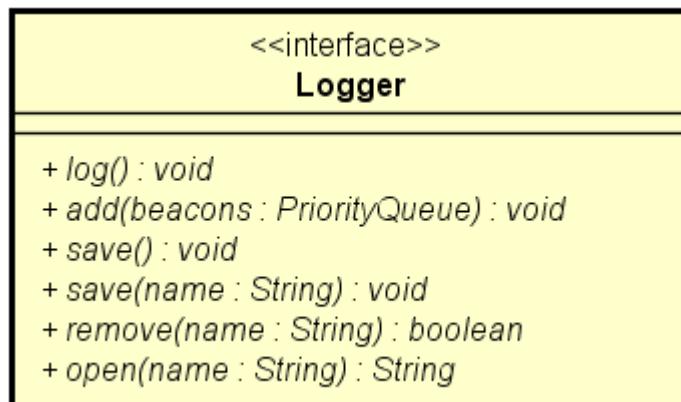


Figura 43: Interfaccia Logger

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) : boolean*

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.

4.4.2.16 model::beacon::LoggerImp

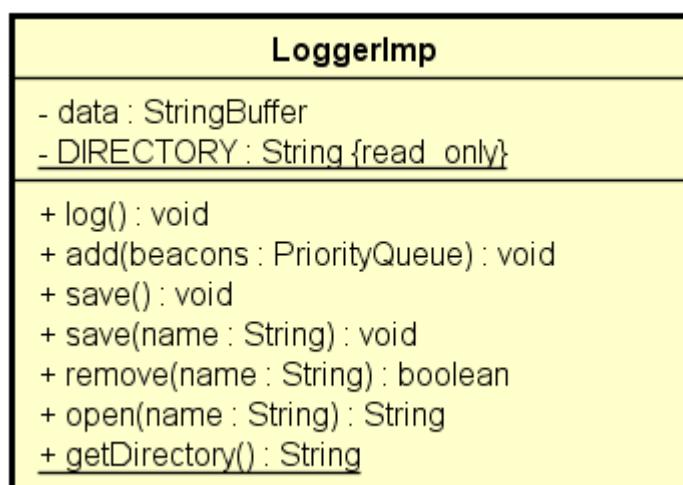


Figura 44: Classe LoggerImp

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:

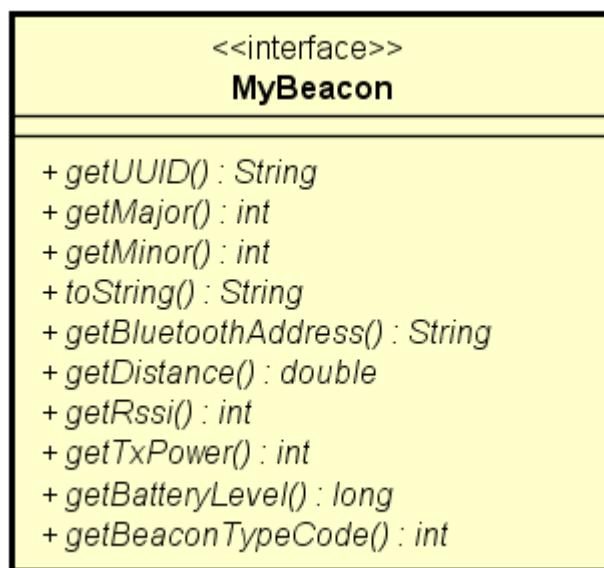
- name : String
Nome del log da cui reperire le informazioni
- + remove(name : String) : boolean
Override Metodo per la rimozione di un log precedentemente salvato

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

4.4.2.17 model::beacon::MyBeacon**Figura 45:** 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'esterno 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

4.4.2.18 model::beacon::MyBeaconImp

MyBeaconImp	
<u>+ CREATOR : Parcelable.Creator {readOnly}</u>	
+ 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 46: 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 : Parcelable.Creator<MyBeacon> {readOnly}
Attributo richiesto per rendere la classe Parcelable

Metodi:

- + MyBeaconImp(beacon : Beacon)
Costruttore della classe

Argomenti:

- beacon : Beacon
Questo parametro richiede un beacon

- + 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
- + 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

4.4.2.19 model::beacon::MyDistanceCalculator

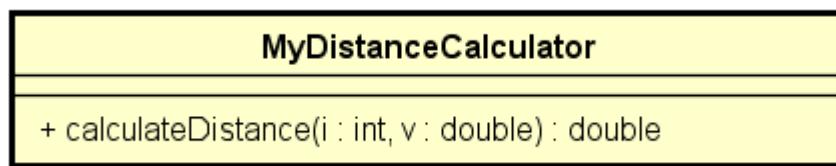


Figura 47: Classe MyDistanceCalculator

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

4.4.2.20 model::beacon::PeriodType

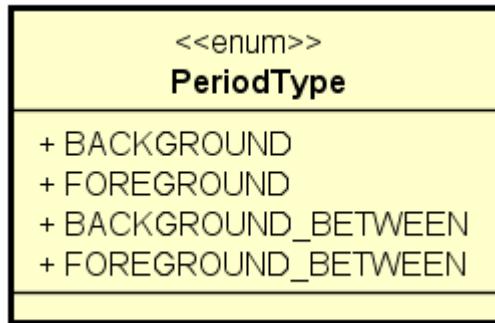


Figura 48: Classe PeriodType

Nome: PeriodType;

Tipo: Classe;

Visibilità: public;

Utilizzo: Utilizzato per il indicare il tipo di periodo di scan da modificare;

Descrizione: Classe che rappresenta il tipo di periodo di scan che deve essere modificato da BeaconManagerAdapter;

4.4.2.21 model::compass::Compass

Compass
<ul style="list-style-type: none">- sensorManager : SensorManager- accelerometer : Sensor- magnetometer : Sensor- lastAccelerometerSet : boolean- lastMagnetometerSet : boolean- rotationMatrix : float[0..*]- orientation : float[0..*]- lastAccelerometerData : float[0..*]- lastMagnetometerData : float[0..*]- listeners : Collection
<ul style="list-style-type: none">+ Compass(sensorManager : SensorManager)+ onSensorChanged(event : SensorEvent) : void+ onAccuracyChanged(sensor : Sensor, accuracy : int) : void+ unregisterListener() : void+ registerListener() : void+ getLastCoordinate() : float+ addListener(listener : CompassListener) : void+ removeListener(listener : CompassListener) : void

Figura 49: 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
- - `lastAccelerometerData` : `float[]`
array che contiene gli ultimi dati ricevuti dall'accelerometro
- - `lastAccelerometerSet` : `boolean`
variabile di guardia per accertarsi il reperimento di almeno un dato dall'accelerometro
- - `lastMagnetometerData` : `float[]`
array che contiene gli ultimi dati ricevuti dal magnetometro
- - `lastMagnetometerSet` : `boolean`
variabile di guardia per accertarsi il reperimento di almeno un dato dal magnetometro
- - `listeners` : `Collection<CompassListener>`
Collezione dei listener dei valori della bussola
- - `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(sensorManager : SensorManager)`
Costruttore della classe Compass

Argomenti:

– `sensorManager` : `SensorManager`
Classe Android che permette di ottenere i riferimenti dei sensori del device

- + `addListener(listener : CompassListener) : void`
Metodo che permette di aggiungere un listener sui valori della bussola

Argomenti:

- **listener** : `CompassListener`
Listener per i valori della bussola
- + **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'accuratezza 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
- + **removeListener(listener : CompassListener) : void**
Metodo che permette di rimuovere un listener sui valori della bussola

Argomenti:

- **listener** : `CompassListener`
Listener da rimuovere dalla collezione di listener dei valori della bussola
- + **unregisterListener() : void**
Metodo che permette all'oggetto Compass di smettere di ricevere dati dai sensori e quindi spegnerli

4.4.2.22 model::compass::CompassListener

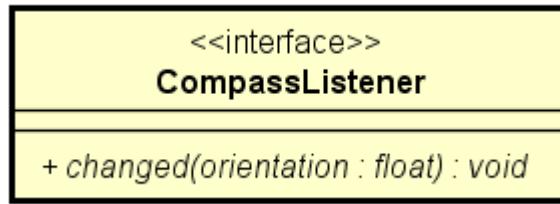


Figura 50: Interfaccia CompassListener

Nome: *CompassListener*;

Tipo: Interfaccia;

Estende:

- Listener.

Visibilità: public;

Utilizzo: Interfaccia utilizzata dalle classi per poter ricevere le informazioni relative alla bussola;

Descrizione: Interfaccia base che deve essere implementata dalle classi che vogliono leggere i cambiamenti della bussola;

Metodi:

- + *changed(orientation : float) : void*

Metodo che viene invocato ad ogni cambiamento della bussola

Argomenti:

- orientation : float

Gradazione rilevata dalla bussola

4.4.2.23 model::dataaccess::dao::BuildingContract

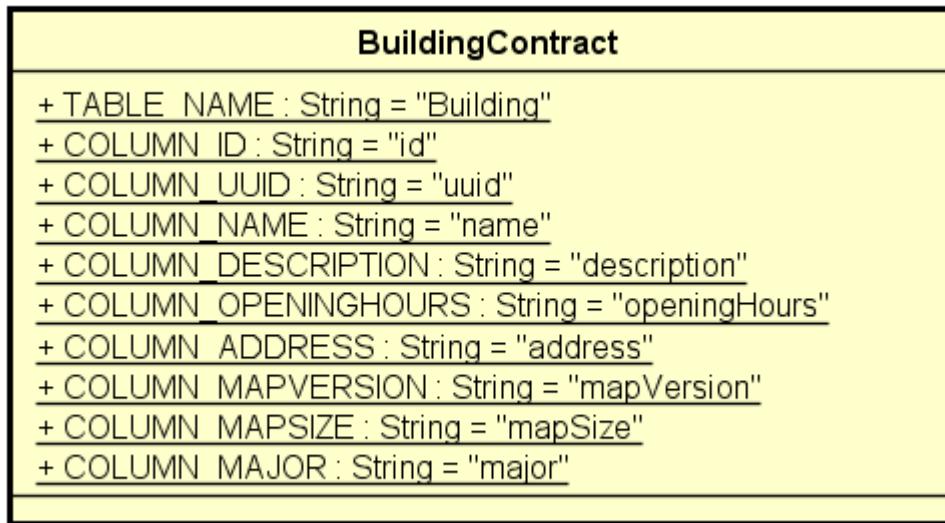


Figura 51: Classe BuildingContract

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:

- + 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_MAJOR : String {readOnly}
Valore della colonna major. Valore di default "major"
- + COLUMN_MAPSIZE : String {readOnly}
Valore della colonna mapSize. Valore di default "size"

- + 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"

4.4.2.24 model::dataaccess::dao::BuildingDao

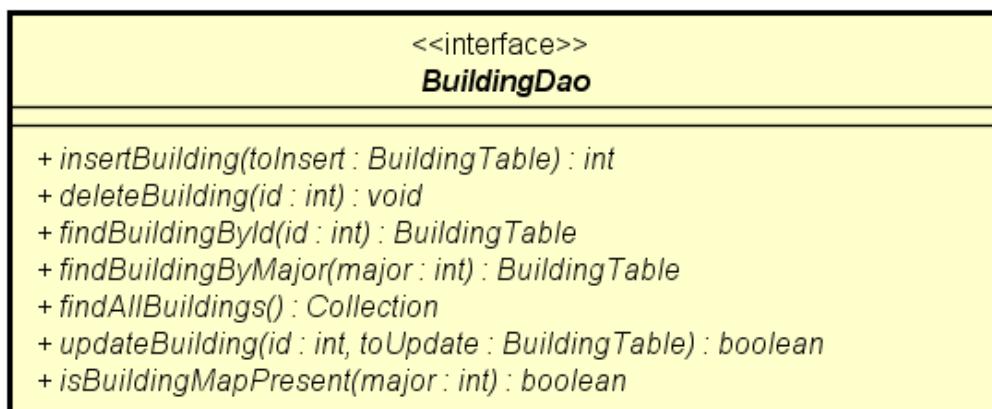


Figura 52: Interfaccia BuildingDao

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

4.4.2.25 model::dataaccess::dao::BuildingTable

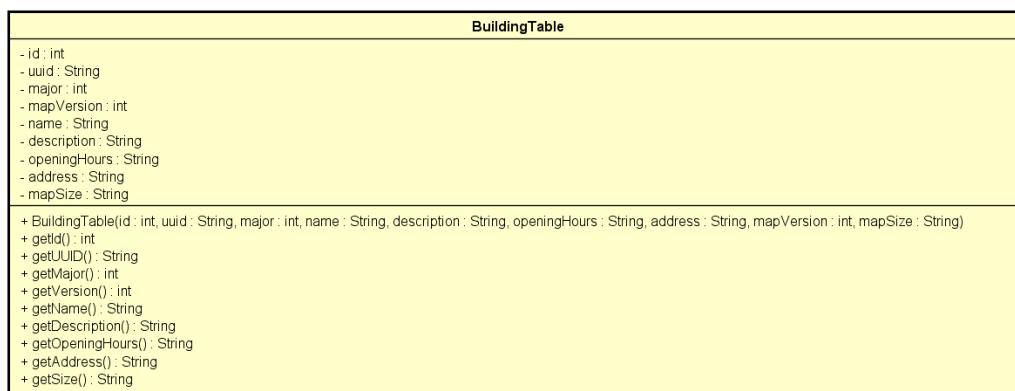


Figura 53: Classe BuildingTable

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

4.4.2.26 model::dataaccess::dao::CategoryContract

CategoryContract
+ TABLE_NAME : String = "Category"
+ COLUMN_ID : String = "id"
+ COLUMN_DESCRIPTION : String = "description"

Figura 54: 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:

- + 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"

4.4.2.27 model::dataaccess::dao::CategoryDao

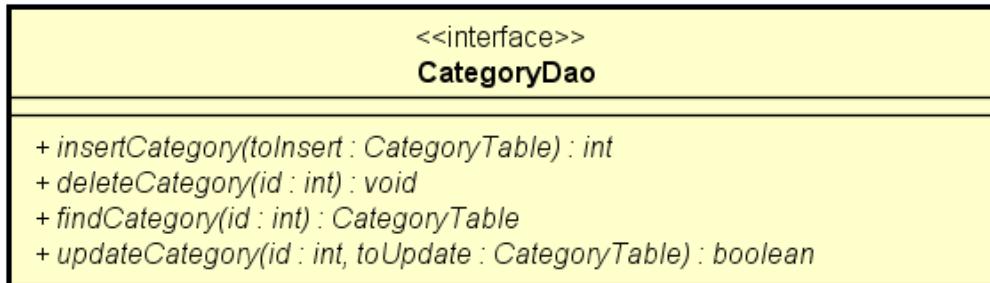


Figura 55: Interfaccia CategoryDao

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

4.4.2.28 model::dataaccess::dao::CategoryTable

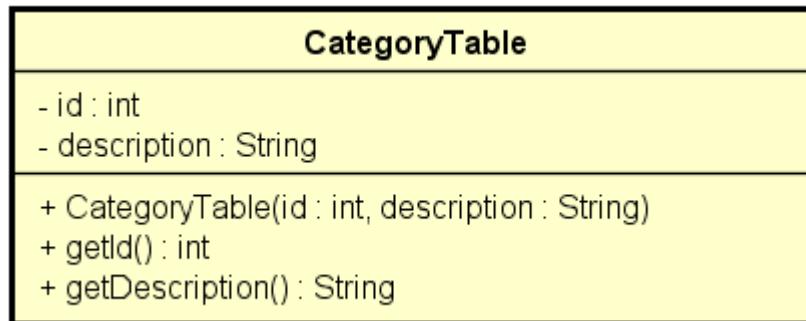


Figura 56: Classe CategoryTable

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

4.4.2.29 model::dataaccess::dao::CursorConverter

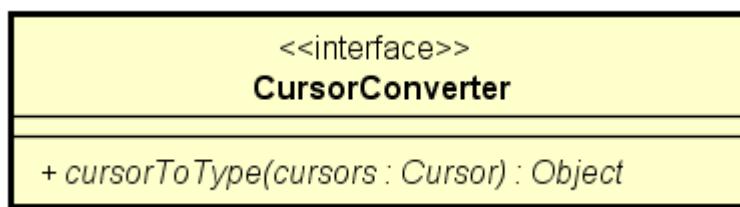


Figura 57: Interfaccia CursorConverter

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

4.4.2.30 model::dataaccess::dao::DaoFactoryHelper

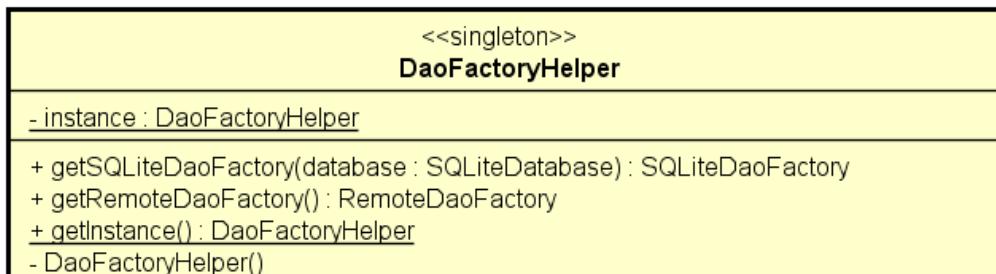


Figura 58: Classe DaoFactoryHelper

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);

Attributi:

- - instance : DaoFactoryHelper
Istanza di DaoFactoryHelper salvata per poter essere condivisa

Metodi:

- - DaoFactoryHelper()
Costruttore della classe DaoFactoryHelper
- + getInstance() : DaoFactoryHelper
Metodo che ritorna un'istanza di DaoFactoryHelper
- + 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

4.4.2.31 model::dataaccess::dao::EdgeContract

EdgeContract	
+ 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"	

Figura 59: 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"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "Edge"

4.4.2.32 model::dataaccess::dao::EdgeDao

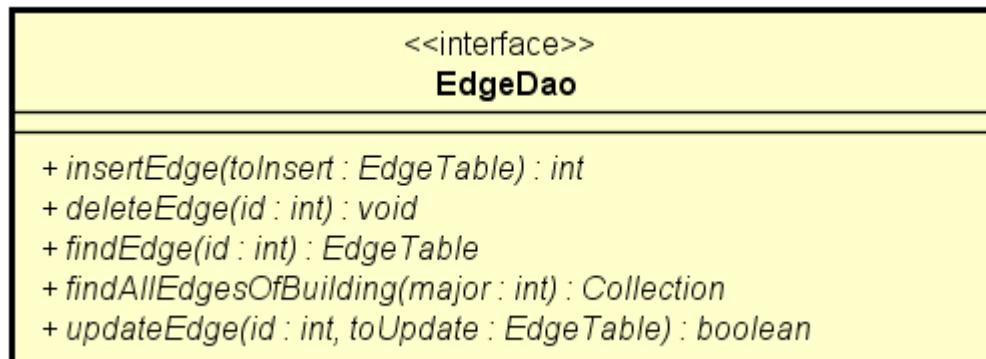


Figura 60: Interfaccia EdgeDao

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

4.4.2.33 model::dataaccess::dao::EdgeTable

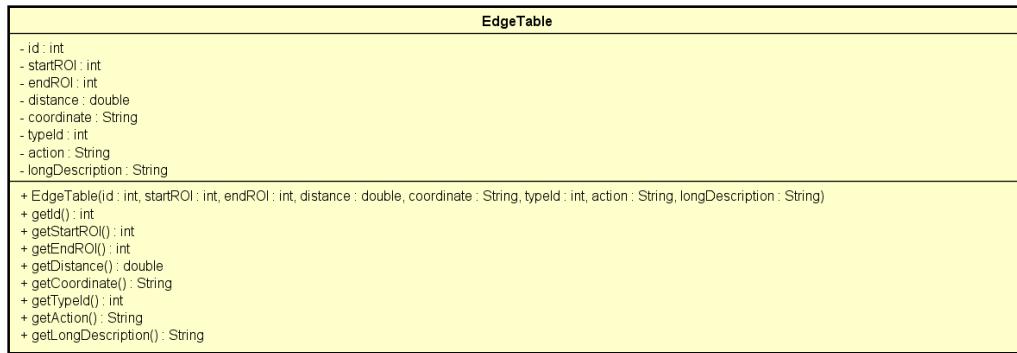


Figura 61: Classe EdgeTable

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

4.4.2.34 model::dataaccess::dao::EdgeTypeContract

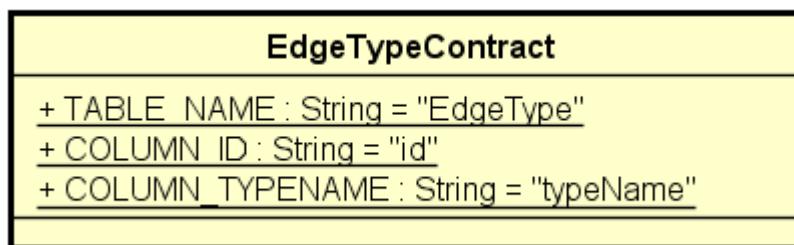


Figura 62: Classe EdgeTypeContract

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"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "EdgeType"

4.4.2.35 model::dataaccess::dao::EdgeTypeDao

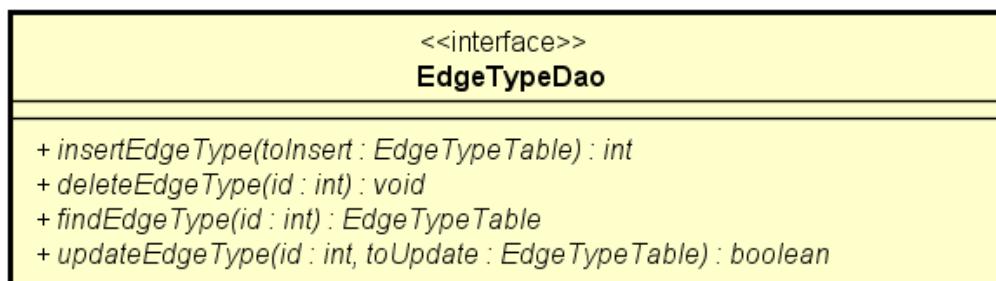


Figura 63: Interfaccia EdgeTypeDao

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

4.4.2.36 model::dataaccess::dao::EdgeTypeTable

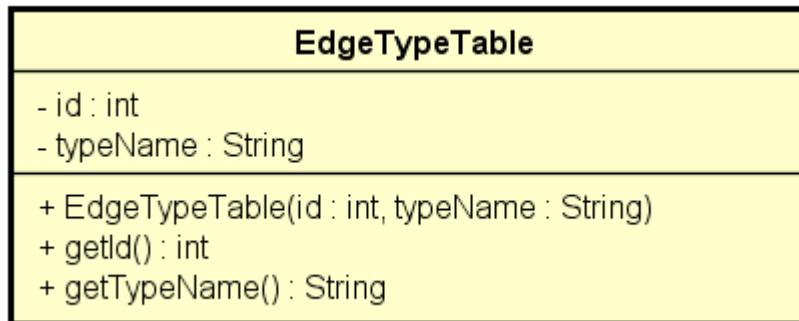


Figura 64: Classe EdgeTypeTable

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
- + `getTypeName()` : String
Metodo che restituisce l'identificativo numerico che permette di identificare il tipo di Edge

4.4.2.37 model::dataaccess::dao::MapsDbHelper

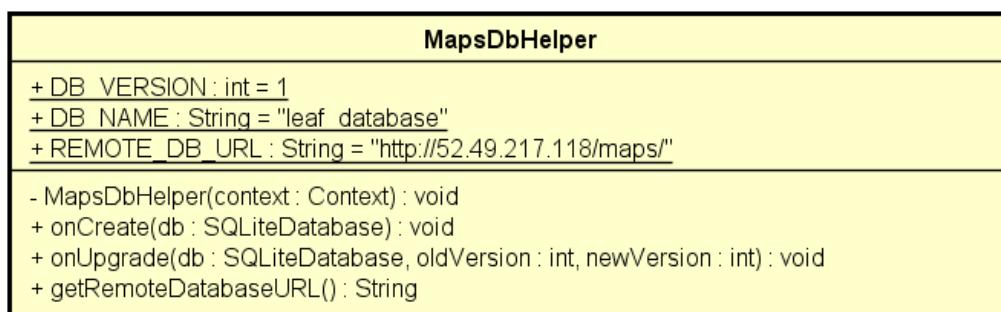


Figura 65: Classe MapsDbHelper

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"

- + REMOTE_DB_URL : String {readOnly}
URL del database remoto

Metodi:

- - MapsDbHelper()
Costruttore della classe MapsDbHelper
- + getRemoteDatabaseURL() : String
Metodo che ritorna l'URL del database remoto
- + 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

4.4.2.38 model::dataaccess::dao::PhotoContract

PhotoContract
+ TABLE_NAME : String = "Photo"
+ COLUMN_ID : String = "id"
+ COLUMN_URL : String = "URL"
+ COLUMN_EDGEID : String = "edgeld"

Figura 66: 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"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "Photo"

4.4.2.39 model::dataaccess::dao::PhotoDao

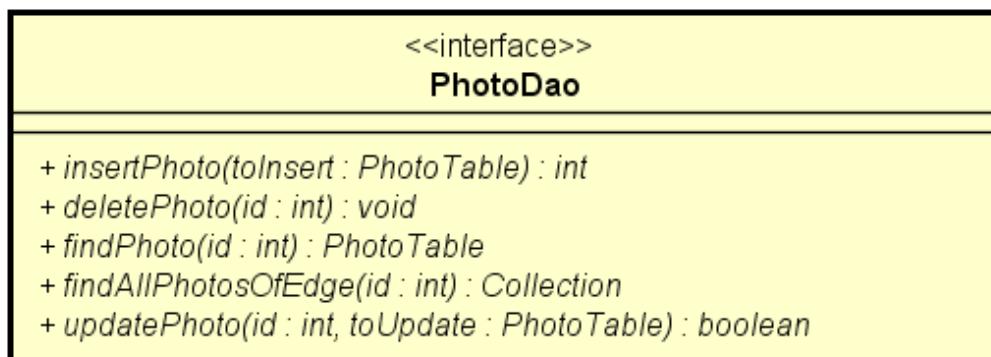


Figura 67: Interfaccia PhotoDao

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

4.4.2.40 model::dataaccess::dao::PhotoTable

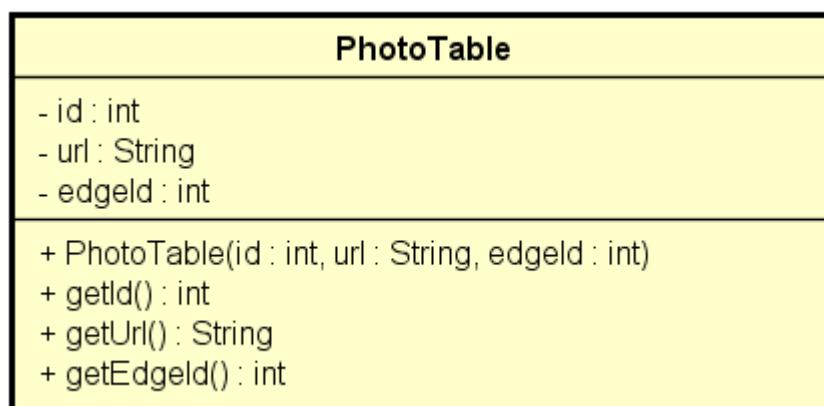


Figura 68: Classe PhotoTable

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

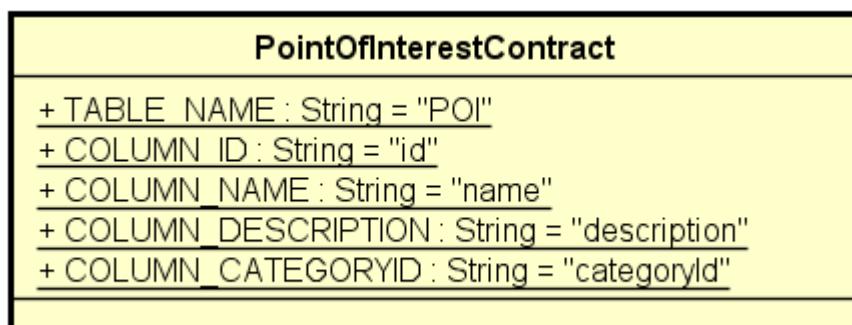
Metodi:

- + `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

- + `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

4.4.2.41 model::dataaccess::dao::PointOfInterestContract**Figura 69:** 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"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "POI"

4.4.2.42 model::dataaccess::dao::PointOfInterestDao

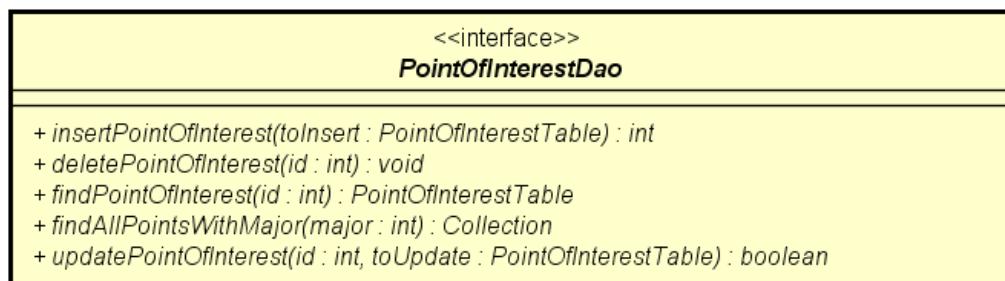


Figura 70: Interfaccia PointOfInterestDao

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

4.4.2.43 model::dataaccess::dao::PointOfInterestTable

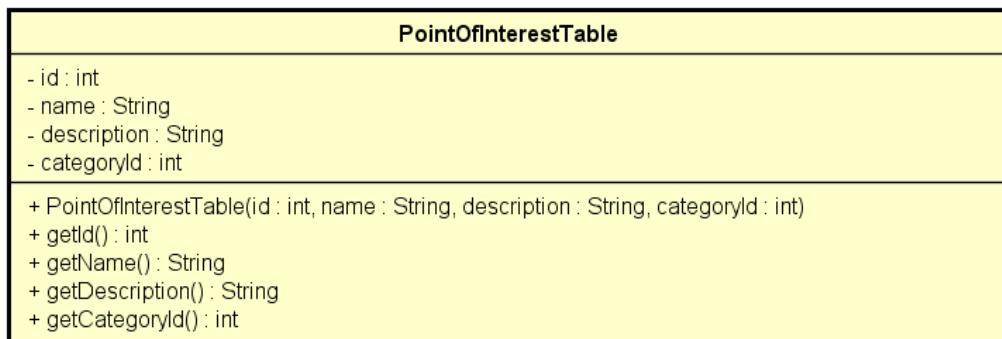


Figura 71: Classe PointOfInterestTable

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:

- + `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

- + `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

4.4.2.44 model::dataaccess::dao::RegionOfInterestContract

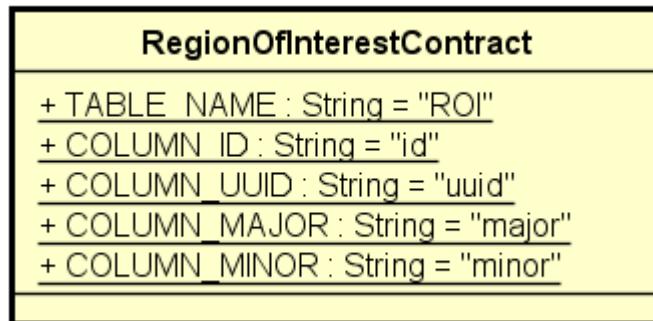


Figura 72: Classe RegionOfInterestContract

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"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "ROI"

4.4.2.45 model::dataaccess::dao::RegionOfInterestDao



Figura 73: 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

4.4.2.46 model::dataaccess::dao::RegionOfInterestTable

RegionOfInterestTable	
- id : int	
- uuid : String	
- major : int	
- minor : int	
+ RegionOfInterestTable(id : int, uuid : String, major : int, minor : int)	
+ getId() : int	
+ getUUID() : String	
+ getMajor() : int	
+ getMinor() : int	

Figura 74: 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:

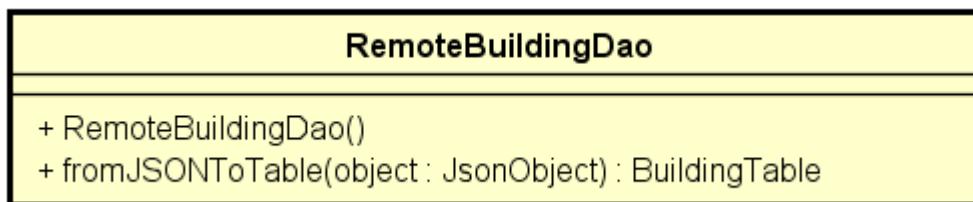
- - id : int
Identificativo della RegionOfInterest
- - major : int
Major dell'edificio
- - minor : int
Identificativo del beacon associato alla ROI
- - uuid : String
UUID dell'applicazione

Metodi:

- + 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
- + 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

4.4.2.47 model::dataaccess::dao::RemoteBuildingDao**Figura 75:** 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 (Gson).

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:

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

4.4.2.48 model::dataaccess::dao::RemoteCategoryDao

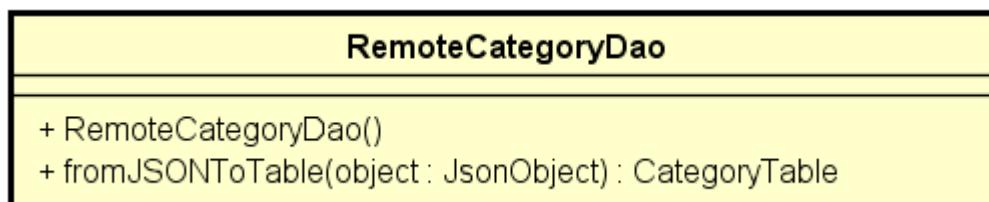


Figura 76: Classe RemoteCategoryDao

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 (Gson).

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:

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

4.4.2.49 model::dataaccess::dao::RemoteDaoFactory

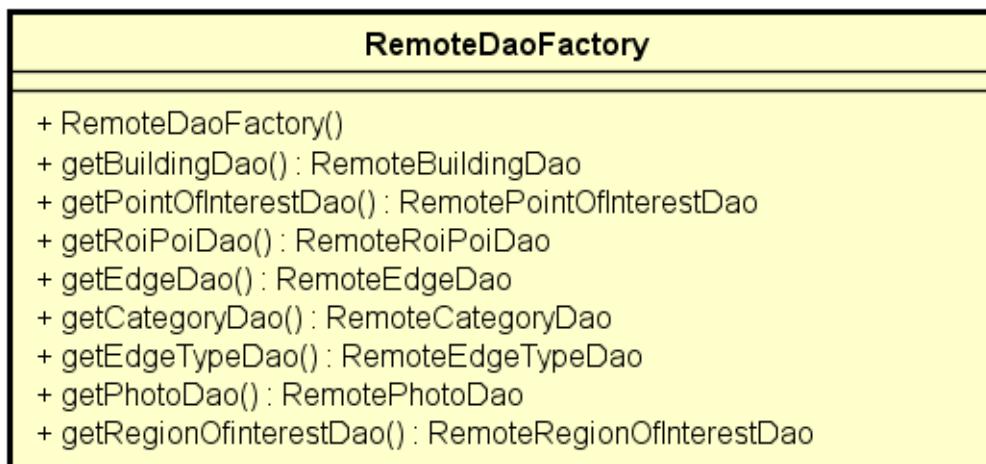


Figura 77: Classe RemoteDaoFactory

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:

- + `RemoteDaoFactory()`
Costruttore di default per la classe RemoteDaoFactory
- + `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

4.4.2.50 model::dataaccess::dao::RemoteEdgeDao

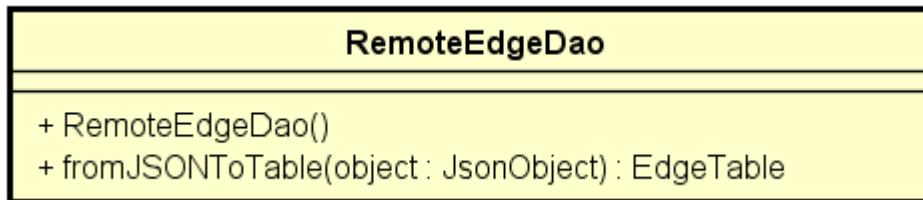


Figura 78: Classe RemoteEdgeDao

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 (Gson).

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:

- + **RemoteEdgeDao()**
Costruttore di default per la classe RemoteEdgeDao
- + **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

4.4.2.51 model::dataaccess::dao::RemoteEdgeTypeDao

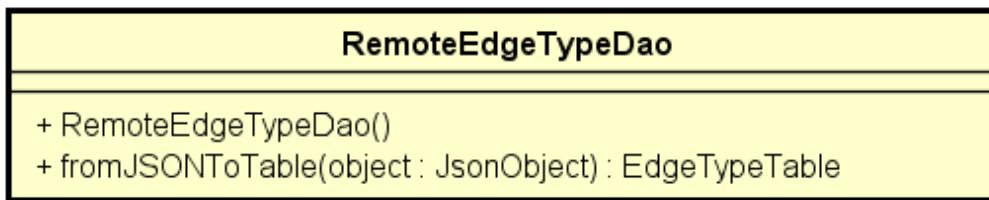


Figura 79: Classe RemoteEdgeTypeDao

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 (Gson).

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:

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

4.4.2.52 model::dataaccess::dao::RemotePhotoDao

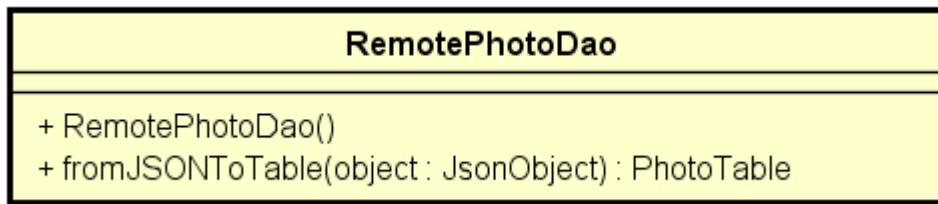


Figura 80: Classe RemotePhotoDao

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 (Gson).

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:

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

4.4.2.53 model::dataaccess::dao::RemotePointOfInterestDao

RemotePointOfInterestDao
+ RemotePointOfInterestDao()
+ fromJSONToTable(object : JsonObject) : PointOfInterestTable

Figura 81: 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 (Gson).

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:

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

4.4.2.54 model::dataaccess::dao::RemoteRegionOfInterestDao

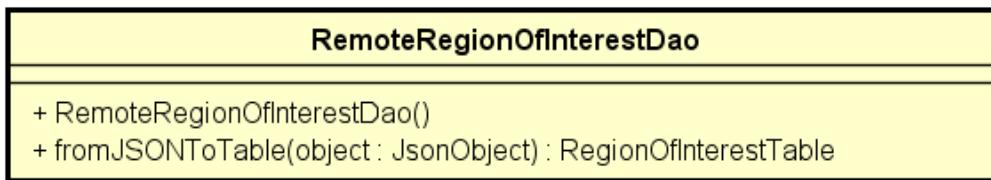


Figura 82: Classe RemoteRegionOfInterestDao

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 (Gson).

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:

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

4.4.2.55 model::dataaccess::dao::RemoteRoiPoiDao

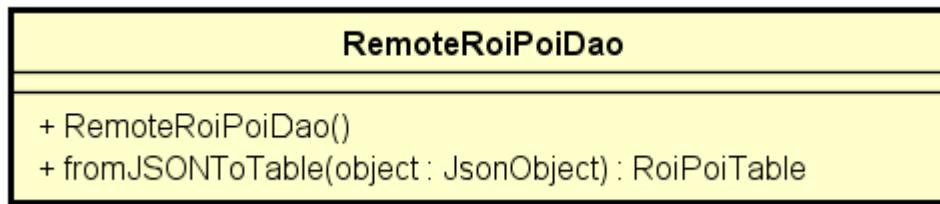


Figura 83: Classe RemoteRoiPoiDao

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 (Gson).

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:

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

4.4.2.56 model::dataaccess::dao::RoiPoiContract



Figura 84: 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"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "ROIPOI"

4.4.2.57 model::dataaccess::dao::RoiPoiDao

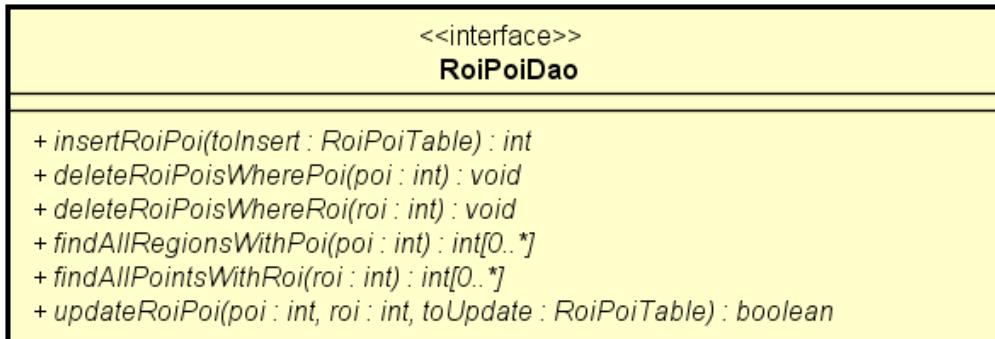


Figura 85: Interfaccia RoiPoiDao

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

4.4.2.58 model::dataaccess::dao::RoiPoiTable

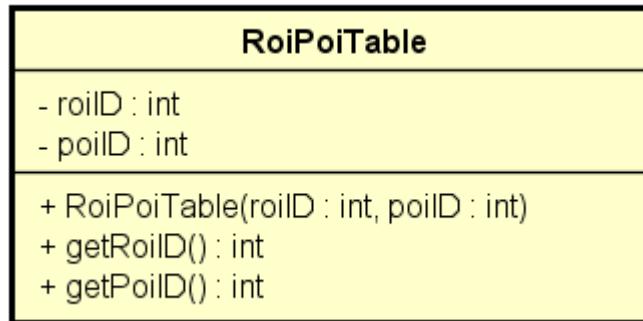


Figura 86: Classe RoiPoiTable

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:

- `+ RoiPoiTable(roiID : int, poiID : int)`
Costruttore della classe RoiPoiTable

Argomenti:

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

- + `getPoiID() : int`
Metodo che restituisce l'identificativo del POI
- + `getRoiID() : int`
Metodo che restituisce l'identificativo del ROI

4.4.2.59 model::dataaccess::dao::SQLDao

SQLDao
<pre>- database : SQLiteDatabase + SQLDao(database : SQLiteDatabase) : void + insert(tableName : String, values : ContentValues) : long + delete(tableName : String, where : String, whereArgs : String[0 ..]) : int + query(distinct : boolean, tableName : String, columns : String[0 ..], where : String, whereArgs : String[0 ..], groupBy : String, having : String, orderBy : String, limit : String) : Cursor + update(tableName : String, values : ContentValues, where : String, whereArgs : String[0 ..]) : int + rawQuery(sqlQuery : String, whereArgs : String[0 ..]) : Cursor</pre>

Figura 87: 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:

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

Argomenti:

- `database : SQLiteDatabase`
Database locale dell'applicazione

- + `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
- + **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

4.4.2.60 model::dataaccess::dao::SQLiteBuildingDao

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

Figura 88: Classe SQLiteBuildingDao

Nome: SQLiteBuildingDao;

Tipo: Classe;

Implementa:

- BuildingDao;
- CursorConverter.

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:

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

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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

- • `+ 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

4.4.2.61 model::dataaccess::dao::SQLiteCategoryDao

SQLiteCategoryDao
- <code>sqlDao</code> : <code>SQLDao</code>
+ <code>SQLiteCategoryDao(database : SQLiteDatabase)</code>
+ <code>insertCategory(toInsert : CategoryTable) : int</code>
+ <code>deleteCategory(id : int) : void</code>
+ <code>findCategory(id : int) : CategoryTable</code>
+ <code>updateCategory(id : int, toUpdate : CategoryTable) : boolean</code>
+ <code>cursorToType(cursor : Cursor) : CategoryTable</code>

Figura 89: Classe SQLiteCategoryDao

Nome: `SQLiteCategoryDao`;

Tipo: Classe;

Implementa:

- CategoryDao;
- CursorConverter.

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:

- + SQLiteCategoryDao(database : SQLiteDatabase)
Costruttore della classe SQLiteCategoryDao

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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
- + **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

4.4.2.62 model::dataaccess::dao::SQLiteDaoFactory

SQLiteDaoFactory	
-	database : SQLiteDatabase
+	SQLiteDaoFactory(database : SQLiteDatabase)
+	getBuildingDao() : BuildingDao
+	getPointOfInterestDao() : PointOfInterestDao
+	getRegionOfInterestDao() : RegionOfInterestDao
+	getRoiPoiDao() : RoiPoiDao
+	getEdgeDao() : EdgeDao
+	getCategoryDao() : CategoryDao
+	getEdgeTypeDao() : EdgeTypeDao
+	getPhotoDao() : PhotoDao

Figura 90: 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:

- + SQLiteDaoFactory(database : SQLiteDatabase)
Costruttore della classe SQLiteDaoFactory

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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

4.4.2.63 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 91: Classe SQLiteEdgeDao

Nome: SQLiteEdgeDao;

Tipo: Classe;

Implementa:

- EdgeDao;
- CursorConverter.

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:

- + SQLiteEdgeDao(database : SQLiteDatabase)
Costruttore della classe SQLiteEdgeDao

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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

- + `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

4.4.2.64 model::dataaccess::dao::SQLiteEdgeTypeDao

SQLiteEdgeTypeDao
- sqlDao : SQLDao
+ SQLiteEdgeTypeDao(database : SQLiteDatabase)
+ insertEdgeType(toInsert : EdgeTypeTable) : int
+ deleteEdgeType(id : int) : void
+ findEdgeType(id : int) : EdgeTypeTable
+ updateEdgeType(id : int, toUpdate : EdgeTypeTable) : boolean
+ cursorToType(cursor : Cursor) : EdgeTypeTable

Figura 92: Classe SQLiteEdgeTypeDao

Nome: SQLiteEdgeTypeDao;

Tipo: Classe;

Implementa:

- EdgeTypeDao;
- CursorConverter.

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:

- + SQLiteEdgeTypeDao(database : SQLiteDatabase)
Costruttore della classe SQLiteEdgeTypeDao

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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

- • **+ 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

4.4.2.65 model::dataaccess::dao::SQLitePhotoDao

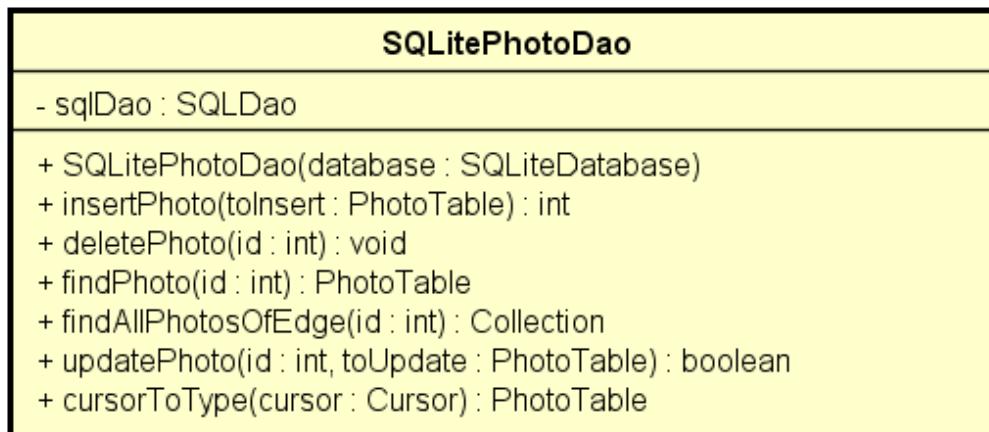


Figura 93: Classe SQLitePhotoDao

Nome: SQLitePhotoDao;

Tipo: Classe;

Implementa:

- PhotoDao;
- CursorConverter.

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:

- + SQLitePhotoDao(database : SQLiteDatabase)
Costruttore della classe SQLitePhotoDao

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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

- + 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

4.4.2.66 model::dataaccess::dao::SQLitePointOfInterestDao

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

Figura 94: 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:

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

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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
- + `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

4.4.2.67 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 95: Classe SQLiteRegionOfInterestDao**Nome:** `SQLiteRegionOfInterestDao;`**Tipo:** Classe;**Implementa:**

- `RegionOfInterestDao;`
- `CursorConverter.`

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:

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

Argomenti:

- `database : SQLiteDatabase`
Il database locale

- + `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
- + **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

4.4.2.68 model::dataaccess::dao::SQLiteRoiPoiDao

SQLiteRoiPoiDao
- sqlDao : SQLDao
+ SQLiteRoiPoiDao(database : SQLiteDatabase)
+ insertRoiPoi(toInsert : RoiPoiTable) : int
+ deleteRoiPoisWherePoi(poi : int) : void
+ deleteRoiPoisWhereRoi(roi : int) : void
+ findAllRegionsWithPoi(poi : int) : int[0..*]
+ findAllPointsWithRoi(roi : int) : int[0..*]
+ updateRoiPoi(poi : int, roi : int, toUpdate : RoiPoiTable) : boolean
+ cursorToType(cursor : Cursor) : RoiPoiTable

Figura 96: Classe SQLiteRoiPoiDao

Nome: SQLiteRoiPoiDao;

Tipo: Classe;

Implementa:

- RoiPoiDao;
- CursorConverter.

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:

- + SQLiteRoiPoiDao(database : SQLiteDatabase)
Costruttore della classe SQLiteRoiPoiDao

Argomenti:

- database : SQLiteDatabase
Il database locale

- + 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

- + `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

4.4.2.69 model::dataaccess::service::BuildingService

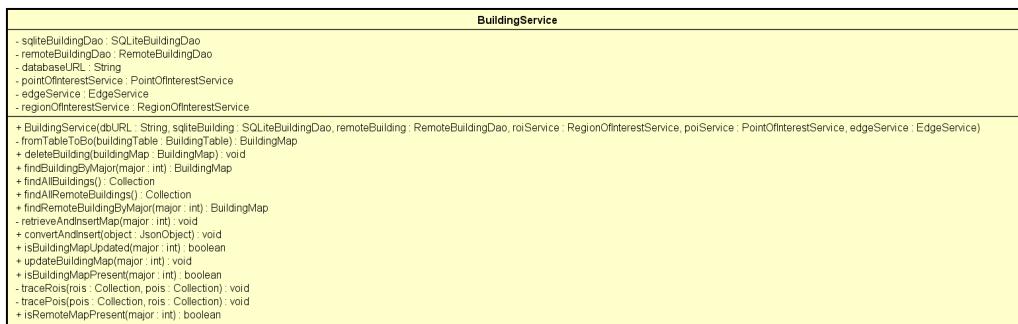


Figura 97: Classe BuildingService

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 (Gson).

Visibilità: public;

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

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

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 : SQLiteBuildingDao, remoteBuilding : RemoteBuildingDao, roiService : RegionOfInterestService, poiService : PointOfInterestService, edgeService : EdgeService)
Costruttore della classe BuildingService

Argomenti:

- **dbURL** : String
URL del database remoto
- **sqliteBuilding** : SQLiteBuildingDao
Oggetto che rappresenta un DAO per la tabella "Building" del database locale
- **remoteBuilding** : 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
- **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(buildingMap : BuildingMap) : void`

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

Argomenti:

– `buildingMap : BuildingMap`

Mappa 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

- + `isRemoteMapPresent() : boolean`

Override Metodo per verificare la presenza sul database remoto della mappa di un 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

- - `tracePois(pois : Collection<PointOfInterest>, rois : Collection<RegionOfInterest>) : void`

Metodo che permette di associare ad ogni PointOfInterest dell'edificio le ROI vicine

Argomenti:

– `pois : Collection<PointOfInterest>`
I PointOfInterest da tracciare

– `rois : Collection<RegionOfInterest>`
Le RegionOfInterest dell'edificio

- - `traceRois(rois : Collection<RegionOfInterest>, pois : Collection<PointOfInterest>) : void`

Metodo che permette di associare ad ogni RegionOfInterest di un edificio i POI vicini

Argomenti:

- rois : Collection<RegionOfInterest>
Le RegionOfInterest da tracciare
 - pois : Collection<PointOfInterest>
I PointOfInterest 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

4.4.2.70 model::dataaccess::service::DatabaseService

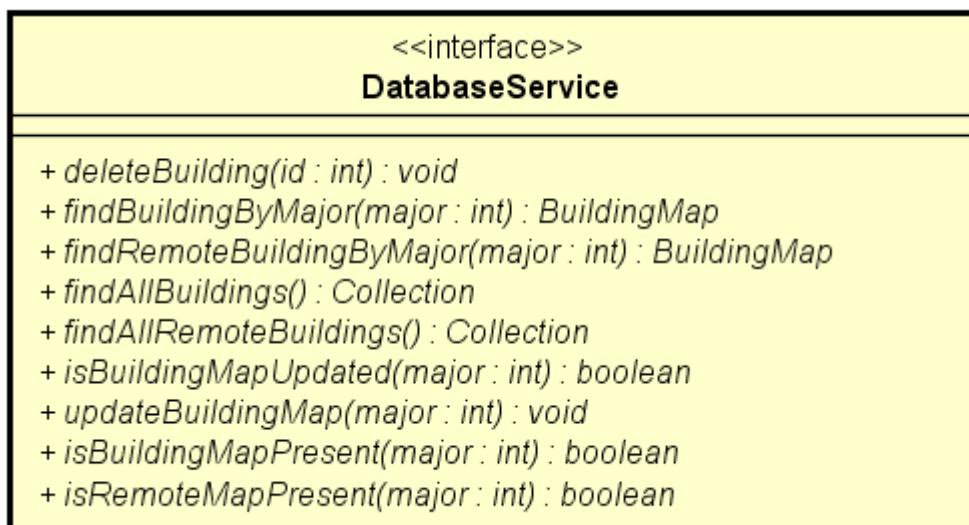


Figura 98: Interfaccia DatabaseService

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 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
- + ***isRemoteMapPresent() : boolean***
Metodo per verificare la presenza sul database remoto della mappa di un edificio
- + ***updateBuildingMap(major : int) : void***
Metodo per aggiornare la mappa di un edificio all'ultima versione disponibile

Argomenti:

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

4.4.2.71 model::dataaccess::service::EdgeService

EdgeService
<pre>- sqliteEdgeDao: SQLiteEdgeDao - remoteEdgeDao: RemoteEdgeDao - sqliteEdgeTypeDao: SQLiteEdgeTypeDao - remoteEdgeTypeDao: RemoteEdgeTypeDao - regionOfInterestService: RegionOfInterestService - photoService: PhotoService + EdgeService(splitEdge: SQLiteEdgeDao, remoteEdge: RemoteEdgeDao, sqliteEdgeType: SQLiteEdgeTypeDao, remoteEdgeType: RemoteEdgeTypeDao, photoService: PhotoService, roiService: RegionOfInterestService) - fromTableToEdge(edgeTable: EdgeTable): EnrichedEdge + deleteEdge(edge: EnrichedEdge): void + findEdge(id: int): EnrichedEdge + findAllEdgesOfBuilding(major: int): Collection<EnrichedEdge> + convertAndInsert(object: JsonObject): void + convertAndInsertEdgeType(object: JsonObject): void + convertAndInsertPhoto(object: JsonObject): void</pre>

Figura 99: 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 (Gson).

Visibilità: public;

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

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

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:

- + `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
- + **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

- + **convertAndInsertPhoto(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 un tipo di Edge

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

Argomenti:

- **edge** : `EnrichedEdge`
L'Edge da rimuovere

- + `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

4.4.2.72 model::dataaccess::service::PhotoService

PhotoService
- <code>sqlitePhotoDao : SQLitePhotoDao</code>
- <code>remotePhotoDao : RemotePhotoDao</code>
+ <code>PhotoService(sqlitePhoto : SQLitePhotoDao, remotePhoto : RemotePhotoDao)</code>
- <code>fromTableToBo(photoTable : PhotoTable) : PhotoRef</code>
+ <code>deletePhoto(id : int) : void</code>
+ <code>findPhoto(id : int) : PhotoRef</code>
+ <code>findAllPhotosOfEdge(id : int) : Collection</code>
+ <code>convertAndInsert(object : JsonObject) : void</code>

Figura 100: 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 (Gson).

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:

- + 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

- + 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

4.4.2.73 model::dataaccess::service::PointOfInterestService

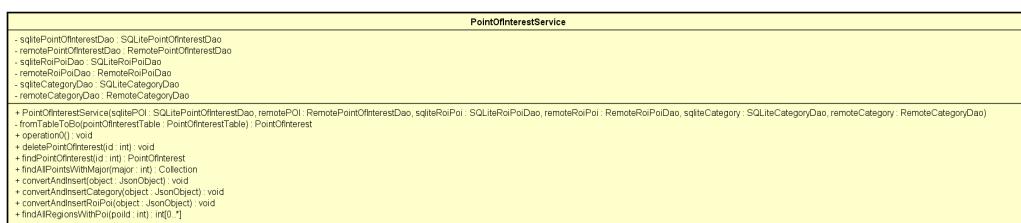


Figura 101: Classe PointOfInterestService

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 (Gson).

Visibilità: public;

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

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

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:

- + 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 RoiPoiTable
 - **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
- + **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

- • + findAllRegionsWithPoi(poiId : int) : int[]

Metodo per recuperare gli identificativi di tutte le RegionOfInterest associate ad uno specifico PointOfInterest

Argomenti:

- poiId : int

Identificativo numerico del PointOfInterest

- • + 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

4.4.2.74 model::dataaccess::service::RegionOfInterestService

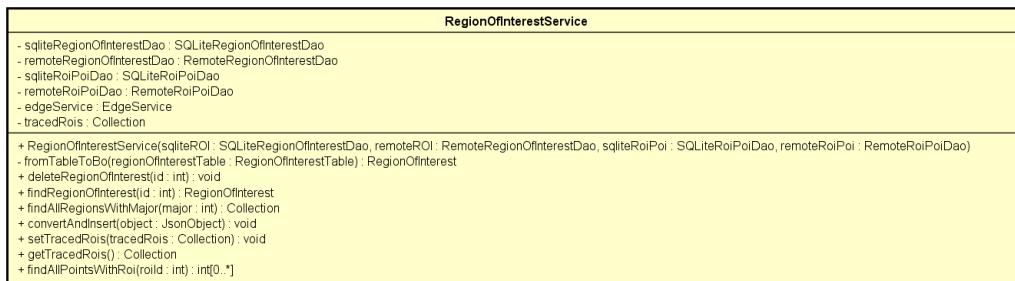


Figura 102: Classe RegionOfInterestService

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 (Gson).

Visibilità: public;

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

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

- - **tracedRois** : Collection<RegionOfInterest>
Lista delle RegionOfInterest della mappa dell'edificio che si sta costruendo che sono già state associate ai PointOfInterest vicini

Metodi:

- + **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

- + **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

- + **findAllPointsWithRoi(roiId : int) : int[]**
Metodo per recuperare gli identificativi di tutti i PointOfInterest associati ad una specifica RegionOfInterest

Argomenti:

- roiId : int

Identificativo della RegionOfInterest

- + 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() : RegionOfInterest
- Metodo per la costruzione di oggetto RegionOfInterest a partire da un RegionOfInterestTable
- + getTracedRois() : Collection<RegionOfInterest>
- Metodo per recuperare le RegionOfInterest che sono già state associate ai POI vicini
- + setTracedRois(tracedRois : Collection<RegionOfInterest>) : void
- Metodo per impostare le RegionOfInterest che sono già state associate i PointOfInterest vicini

Argomenti:

- tracedRois : Collection<RegionOfInterest>

Le RegionOfInterest che sono già state associate ai POI vicini

4.4.2.75 model::dataaccess::service::ServiceHelper

Figura 103: 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)`
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

4.4.2.76 model::navigator::BuildingInformation

BuildingInformation
<pre>- 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</pre>

Figura 104: 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

4.4.2.77 model::navigator::BuildingMap

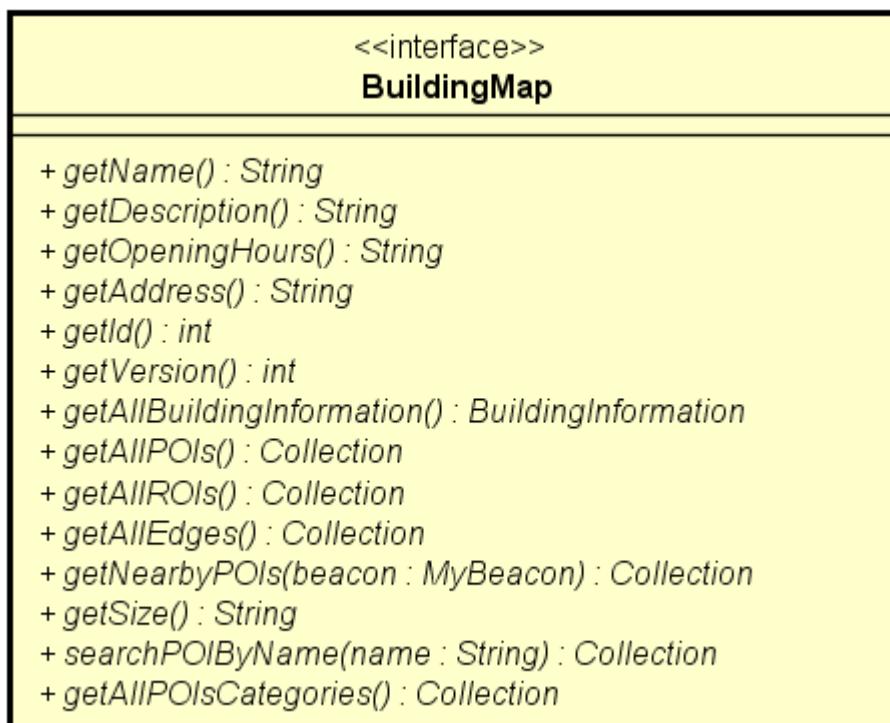


Figura 105: Interfaccia BuildingMap

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

4.4.2.78 model::navigator::BuildingMapImp

BuildingMapImp
<pre> - id : int {readOnly} - version : int {readOnly} - pois : Collection {readOnly} - rois : Collection {readOnly} - edges : Collection {readOnly} - buildingInformation : BuildingInformation {readOnly} - size : String + BuildingMapImp(edges : Collection, id : int, version : int, pois : Collection, rois : Collection, buildingInfo : BuildingInformation, size : String) + getName() : String + getDescription() : String + getOpeningHours() : String + getAddress() : String + getId() : int + getVersion() : int + getSize() : String + getAllBuildingInformation() : BuildingInformation + getAllPOIs() : Collection + getAllROIs() : Collection + getAllEdges() : Collection + getNearbyPOIs(beacon : MyBeacon) : Collection + searchPOIByName(name : String) : Collection + getAllPOIsCategories() : Collection </pre>

Figura 106: 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(edges : Collection<EnrichedEdge>, id : int, version : int, pois : Collection<PointOfInterest>, rois : Collection<RegionOfInterest>, buildingInfo : BuildingInformation, size : String)

Costruttore della classe BuildingMapImp

Argomenti:

- edges : Collection<EnrichedEdge>
Insieme di archi che indicano i possibili percorsi tra due ROI
- 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

- + 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
- + `getNearbyPOIs(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

- + `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 il 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

4.4.2.79 model::navigator::NavigationDirection

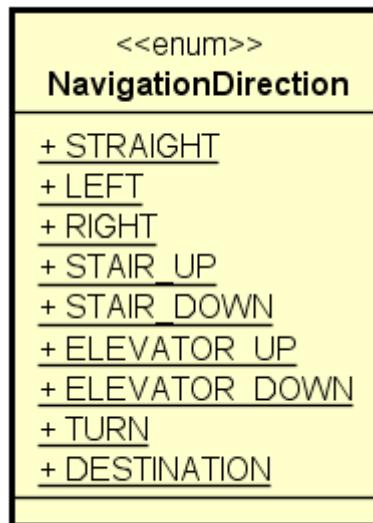


Figura 107: Classe NavigationDirection

Nome: NavigationDirection;

Tipo: Classe;

Visibilità: public;

Utilizzo: Classe utilizzata dalla funzionalità di navigazione per poter rappresentare le varie direzioni;

Descrizione: Classe per rappresentare le direzioni di navigazione;

Attributi:

- - DESTINATION : int
Rappresenta l'arrivo ad una destinazione
- + ELEVATOR_DOWN : int
Rappresenta la direzione che indica di scendere un piano con l'ascensore
- + ELEVATOR_UP : int
Rappresenta la direzione che indica di salire un piano con l'ascensore
- + LEFT : int
Rappresenta la direzione che indica di girare a sinistra

- + RIGHT : int
Rappresenta la direzione che indica di girare a destra
- + STAIR_DOWN : int
Rappresenta la direzione che indica di scendere un piano di scale
- + STAIR_UP : int
Rappresenta la direzione che indica di salire un piano di scale
- - STRAIGHT : int
Rappresenta la direzione che indica di proseguire dritti
- + TURN : int
Rappresenta l'azione di tornare indietro

4.4.2.80 model::navigator::NavigationExceptions

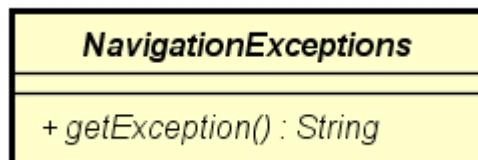


Figura 108: Classe astratta NavigationExceptions

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

4.4.2.81 model::navigator::Navigator

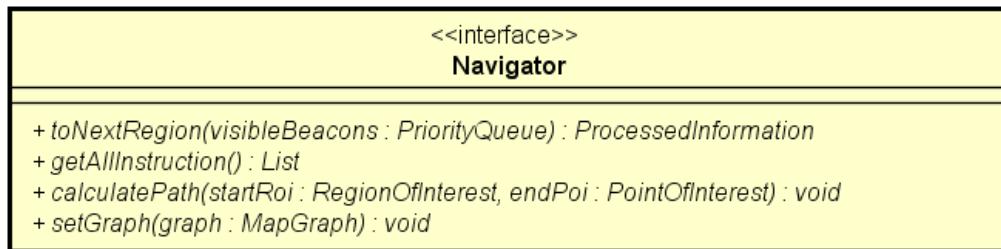


Figura 109: Interfaccia Navigator

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, endPoi : PointOfInterest) : 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. Vierne lanciata un'eccezione di tipo NoGraphSetException nel caso in cui non sia settato alcun grafo.

Argomenti:

- **startRoi : RegionOfInterest**
Punto di partenza del percorso.
- **endPoi : PointOfInterest**
Punto di arrivo del percorso.

- **+ getAllInstructions() : List<ProcessedInformation>**

Metodo che ritorna la lista completa delle ProcessedInstruction da seguire per percorrere un percorso calcolato.

- + *hasFinishedPath() : boolean*

Metodo che ritorna un booleano false se il percorso è concluso

- + *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 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.

4.4.2.82 model::navigator::NavigatorImp

NavigatorImp	
- path : List	
- progress : Iterator	
- buildingGraph : MapGraph	
- pathFinder : PathFinder	
- compass : Compass	
- setting : Setting	
+ NavigatorImp(compass : Compass, setting : Setting)	
+ 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	
- initIterator() : void	
+ hasFinishedPath() : boolean	

Figura 110: 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<EnrichedEdge>`
Iteratore rappresentante il punto che è stato raggiunto durante la navigazione
- - `setting` : `Setting`
Oggetto che rappresenta le impostazioni dell'applicazione

Metodi:

- + `NavigatorImp(compass : Compass)`
Costruttore della classe NavigatorImp

Argomenti:

- `compass` : `Compass`
Sensore di tipo Compass utilizzato durante la navigazione

- + `calculatePath(startRoi : RegionOfInterest, endPoi : PointOfInterest : 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
- `endPoi` : `PointOfInterest`
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
- • - getMostPowerfulBeacon(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
- • + hasFinishedPath() : boolean
Metodo che ritorna un booleano false se il percorso è concluso
- • - initIterator() : void
Metodo che inizializza l'iteratore del percorso calcolato
- • - 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

- • + 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

4.4.2.83 model::navigator::NoGraphSetException

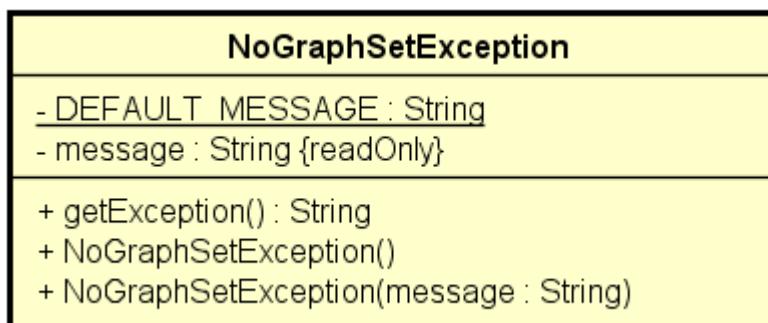


Figura 111: Classe NoGraphSetException

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 avvia la navigazione senza aver settato il grafo
- - message : String {readOnly}
Stringa che rappresenta l'eccezione

Metodi:

- + NoGraphSetException()
Costruttore di default della classe NoGraphSetException
- + NoGraphSetException(message : String)
Costruttore della classe NoGraphSetException

Argomenti:

- message : String
Messaggio che rappresenta l'eccezione
- + getException() : String
Override Metodo che ritorna il messaggio che rappresenta l'eccezione

4.4.2.84 model::navigator::NoNavigationInformationException

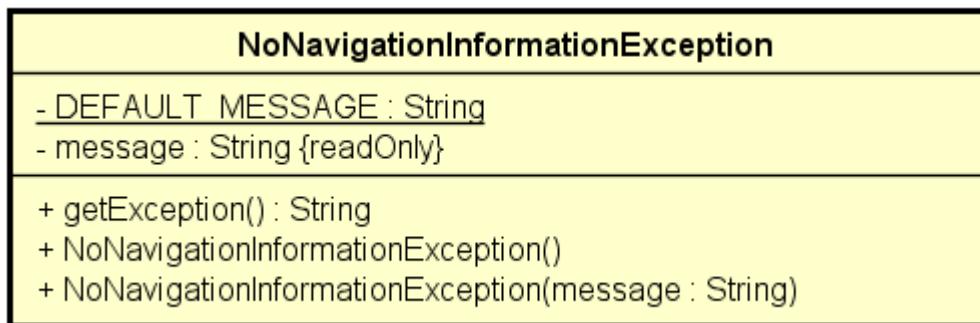


Figura 112: Classe NoNavigationInformationException

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 non si sia ancora calcolato un percorso
- - **message : String {readOnly}**
Stringa che rappresenta l'eccezione

Metodi:

- + NoNavigationInformationException()

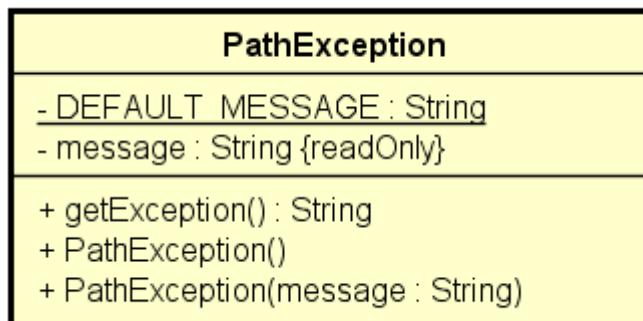
Costruttore di default della classe NoNavigationInformationException
- + NoNavigationInformationException(message : String)

Costruttore della classe NoNavigationInformationException

Argomenti:

- message : String
Messaggio che rappresenta l'eccezione

- + getException() : String
Override Metodo che ritorna il messaggio che rappresenta l'eccezione

4.4.2.85 model::navigator::PathException**Figura 113:** 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 il beacon più potente nella lista di ingresso sia associa
- - message : String {readOnly}
Stringa che rappresenta l'eccezione

Metodi:

- + PathException()
Costruttore di default della classe PathException
- + PathException(message : String)
Costruttore della classe PathException

Argomenti:

- message : String
Messaggio che rappresenta l'eccezione
- + getException() : String
Override Metodo che ritorna il messaggio che rappresenta l'eccezione

4.4.2.86 model::navigator::ProcessedInformation

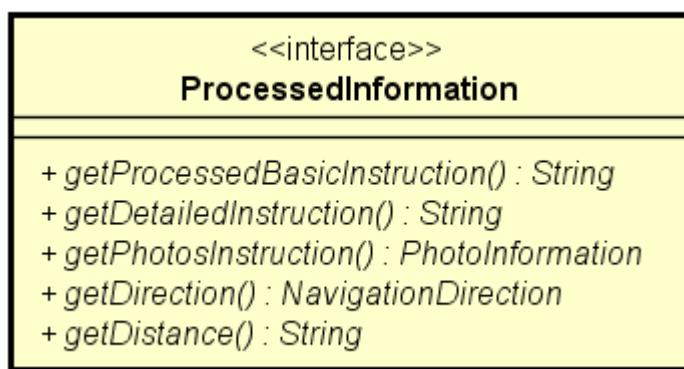


Figura 114: Interfaccia ProcessedInformation

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.
- + *getDirection()* : *NavigationDirection*
Metodo che ritorna la direzione verso cui dirigersi
- + *getDistance()* : *String*
Metodo che ritorna la distanza da percorrere nell'arco in cui ci si trova
- + *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.

4.4.2.87 model::navigator::ProcessedInformationImp

ProcessedInformationImp	
- basic : String	
- detailed : String	
- photos : PhotoInformation	
- direction : NavigationDirection	
- distance : String	
+ ProcessedInformationImp()	
+ ProcessedInformationImp(edge : EnrichedEdge)	
+ ProcessedInformationImp(edge : EnrichedEdge, starterInformation : int)	
+ getProcessedBasicInstruction() : String	
+ getDetailedInstruction() : String	
+ getPhotosInstruction() : PhotoInformation	
+ getDirection() : NavigationDirection	
+ getDistance() : String	
+ equals(o : Object) : boolean	

Figura 115: 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
- - `direction` : `NavigationDirection`
La prossima direzione verso cui dirigersi
- - `distance` : `String`
La distanza da percorrere in un certo Edge
- - `photos` : `List<PhotoRef>`
Lista di foto di un Edge

Metodi:

- + `ProcessedInformationImp()`
Costruttore della classe ProcessedInformationImp
- + `ProcessedInformationImp(edge : EnrichedEdge)`
Costruttore della classe ProcessedInformationImp

Argomenti:

- `edge` : `EnrichedEdge`
Edge da cui devono essere estratte le informazioni

- + `ProcessedInformationImp(edge : EnrichedEdge, starterInformation : int)`
Costruttore della classe ProcessedInformationImp

Argomenti:

- `edge` : `EnrichedEdge`
Edge da cui devono essere estratte le informazioni
- `starterInformation` : `int`
Informazioni aggiuntive per costruire le informazioni associate ad un arco del percorso per superarlo

- + `equals(o : Object) : boolean`

Override Metodo che permette di comparare un oggetto con una istanza della classe ProcessedInformationImp

Argomenti:

- `o` : `Object`
Oggetto da comparare

- + `getDetailedInstruction() : String`

Override Metodo che ritorna le istruzioni dettagliate per superare un certo arco nel percorso calcolato

- + `getDirection() : NavigationDirection`

Override Metodo che ritorna la direzione verso cui dirigersi

- + `getDistance() : String`
Override Metodo che ritorna la distanza da percorrere nell'arco in cui ci si trova
- + `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

4.4.2.88 model::navigator::algorithm::DijkstraPathFinder

DijkstraPathFinder
+ DijkstraPathFinder()
+ calculatePath(MapGraph, startRoi : RegionOfInterest, endRoi : RegionOfInterest) : List

Figura 116: 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>.

Descrizione: Classe che rappresenta un algoritmo per il calcolo del percorso di navigazione;

Metodi:

- + `DijkstraPathFinder()`
Costruttore di default della classe DijkstraPathFinder

- + calculatePath(graph : MapGraph, startROI : RegionOfInterest, endROI : 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 parametron.
- endROI : RegionOfInterest
RegionOfInterest di arrivo del percorso. Deve appartenere al grafo passato come parametron.

4.4.2.89 model::navigator::algorithm::PathFinder



Figura 117: 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.

4.4.2.90 model::navigator::graph::MapGraph

MapGraph
<code>- graph : SimpleDirectedWeightedGraph</code>
<code>+ MapGraph()</code> <code>+ addRegionOfInterest(roi : RegionOfInterest) : void</code> <code>+ addAllRegions(regions : Collection) : void</code> <code>+ addEdge(edge : EnrichedEdge) : void</code> <code>+ addAllEdges(edges : Collection) : void</code> <code>+ getGraph() : SimpleDirectedWeightedGraph</code> <code>+ setSettingAllEdge(setting : Setting) : void</code>

Figura 118: 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.SimpleDirectedWeighted

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:

- + MapGraph()
Costruttore della classe
- + 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

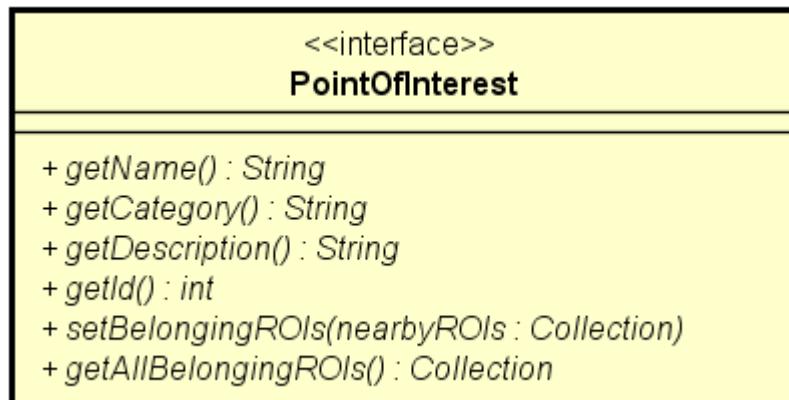
Argomenti:

- roi : RegionOfInterest
RegionOfInterest da aggiungere al grafo che rappresenta l'edificio

- + `getGraph()` : `SimpleDirectedWeightedGraph<RegionOfInterest,EnrichedEdge>`
Metodo che permette di restituire il grafo che rappresenta la distribuzione degli oggetti RegionOfInterest ed EnrichedEdge
- + `setSettingAllEdge(setting : Setting) : void`
Metodo che permette di impostare le setting passate come parametro a tutti gli edge all'interno del graph

Argomenti:

- `setting : Setting`
Impostazioni di preferenza dell'applicazione

4.4.2.91 model::navigator::graph::area::PointOfInterest**Figura 119:** 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:**

- + *getAllBelongingROIs() : 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
- + *setBelongingROIs(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

4.4.2.92 model::navigator::graph::area::PointOfInterestImp

PointOfInterestImp
<ul style="list-style-type: none"> - <i>id : int {readOnly}</i> - <i>info : PointOfInterestInformation {readOnly}</i> - <i>rois : Collection</i> <ul style="list-style-type: none"> + <i>PointOfInterestImp(id : int, description : PointOfInterestInformation)</i> + <i>getName() : String</i> + <i>getDescription() : String</i> + <i>getCategory() : String</i> + <i>getId() : int</i> + <i>getAllBelongingROIs() : Collection</i> + <i>setBelongingROIs(rois : Collection)</i>

Figura 120: 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:

- + PointOfInterestImp(id : int, info : PointOfInterestInformation)
Costruttore della classe PointOfInterestImp

Argomenti:

- id : int
Identificativo numerico della classe PointOfInterestImp.
 - info : PointOfInterestInformation
Informazioni relative ad un POI
- + 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 appartenza 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
- + `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.

4.4.2.93 model::navigator::graph::area::PointOfInterestInformation

PointOfInterestInformation
<ul style="list-style-type: none"> - <code>description : String {readOnly}</code> - <code>name : String {readOnly}</code> - <code>category : String {readOnly}</code>
<ul style="list-style-type: none"> + <code>PointOfInterestInformation(name : String, description : String, category : String)</code> + <code>getName() : String</code> + <code>getDescription() : String</code> + <code>getCategory() : String</code>

Figura 121: 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:

- + `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

- + `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

4.4.2.94 model::navigator::graph::area::RegionOfInterest

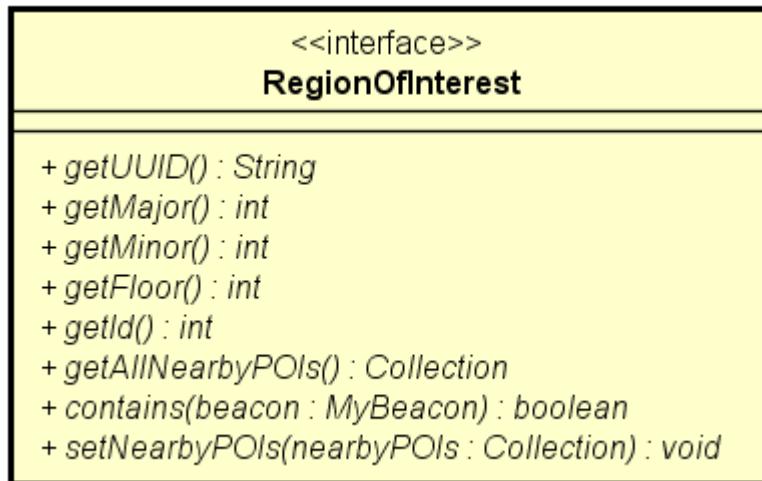


Figura 122: Interfaccia RegionOfInterest

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

4.4.2.95 model::navigator::graph::area::RegionOfInterestImp

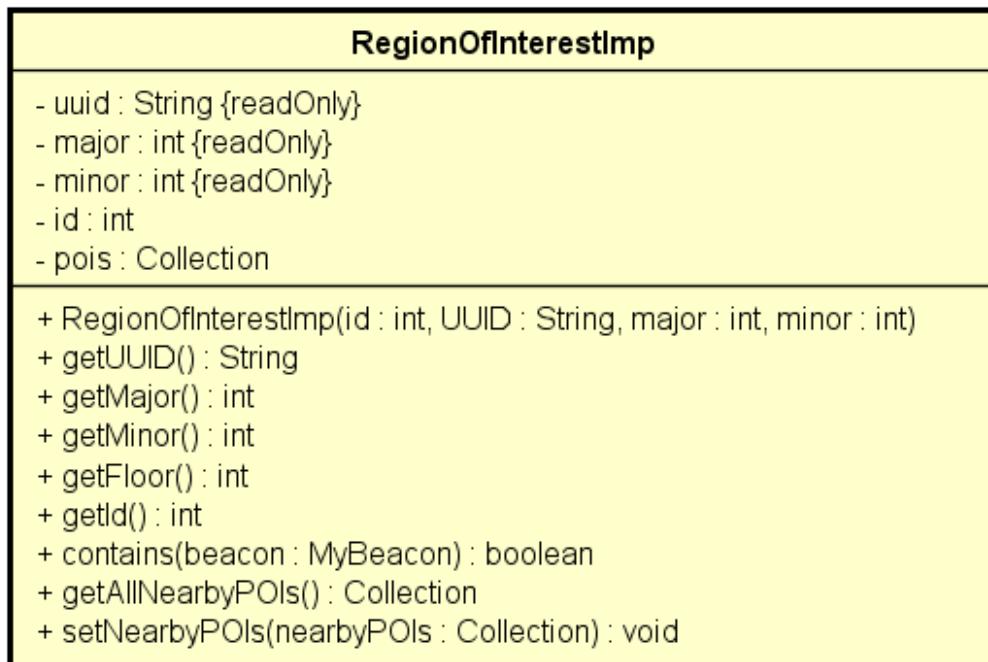


Figura 123: Classe RegionOfInterestImp

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:

- + `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

- + `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
- + `setNearbyPOIs(Collection<PointOfInterest> : nearbyPOIs) : void`
Override Metodo utilizzato per settare l'insieme di PointOfInterest associato a tale RegionOfInterestImp

Argomenti:

- `Collection<PointOfInterest> : nearbyPOIs`
Insieme di nearby POIs

4.4.2.96 model::navigator::graph::edge::AbsEnrichedEdge

<i>AbsEnrichedEdge</i>
<pre> - startROI : RegionOfInterest - endROI : RegionOfInterest - coordinate : int - distance : double # userElevatorPreference : int # userStairPreference : int - id : int - navInfo : NavigationInformation # maxDistance : double + AbsEnrichedEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation) + getStarterPoint() : RegionOfInterest + getEndPoint() : RegionOfInterest + getWeight() : double + getId() : int + getBasicInformation() : String + getDetailedInformation() : String + getPhotoInformation() : PhotoInformation + getDistance() : double + getCoordinate() : int + setUserPreference(userPref : Setting) : void # getNavigationInformation() : NavigationInformation </pre>

Figura 124: 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
- # *maxDistance* : *double*
Distanza massima tra gli Edge istanziati. Valore di default: 0
- - *navInfo* : *NavigationInformation {readOnly}*
Informazioni associate ad un arco per il superamento dello stesso
- - *startROI* : *RegionOfInterest {readOnly}*
Punto di partenza dell'arco
- # *userElevatorPreference* : *int*
Preferenze dell'utente rispetto agli archi che prevedono un ascensore
- # *userStairPreference* : *int*
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`
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 un oggetto PhotoInformation contenente un riferimento alle fotografie associate all'arco
- + `getStarterPoint() : RegionOfInterest`
Override Metodo che ritorna la RegionOfInterest di partenza dell'arco
- + `getWeight() : double`
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

4.4.2.97 model::navigator::graph::edge::DefaultEdge

DefaultEdge
+ DefaultEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation) + getWeight() : double + getBasicInformation() : String + getDetailedInformation() : String

Figura 125: 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() : double
Override Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

4.4.2.98 model::navigator::graph::edge::Edge

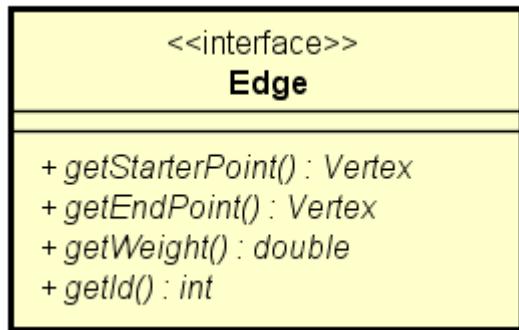


Figura 126: Interfaccia Edge

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() : double*
Metodo che ritorna il peso dell'arco

4.4.2.99 model::navigator::graph::edge::ElevatorEdge

ElevatorEdge
<u>- NO_ELEVATOR_FACTOR : int {readOnly}</u>
+ ElevatorEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)
+ getWeight() : double
+ getBasicInformation() : String
+ getDetailedInformation() : String

Figura 127: 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 uguale per tutti gli oggetti ElevatorEdge

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()** : double
Override Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

4.4.2.100 model::navigator::graph::edge::EnrichedEdge

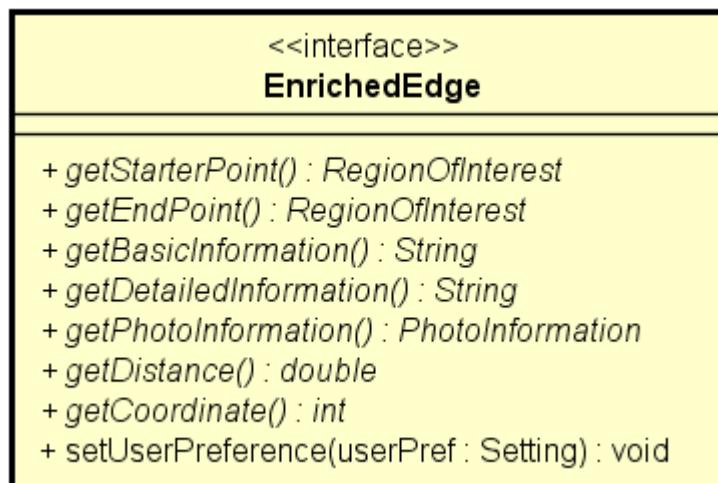


Figura 128: Interfaccia EnrichedEdge

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
- *+ getCoordinate() : int*
Metodo che ritorna le coordinate di un 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
- *+ setUserPreference(setting : Setting) : void*
Metodo che permette di impostare le preferenze di un utente per il calcolo del peso dell'arco

Argomenti:

- *setting : Setting*
setting Preferenze da impostare

4.4.2.101 model::navigator::graph::edge::StairEdge

StairEdge
<u>- NO_STAIR_FACTOR : int {readOnly}</u>
+ StairEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)
+ getWeight() : double
+ getBasicInformation() : String
+ getDetailedInformation() : String

Figura 129: 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 uguale per tutti gli oggetti StairEdge

Metodi:

- + 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
- • `+ 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() : double`
Override Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

4.4.2.102 model::navigator::graph::navigationinformation::BasicInformation

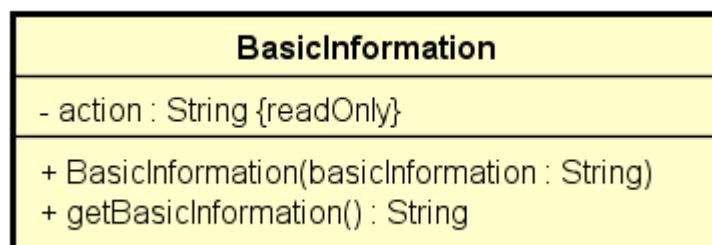


Figura 130: Classe BasicInformation

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

4.4.2.103 model::navigator::graph::navigationinformation::DetailedInformation

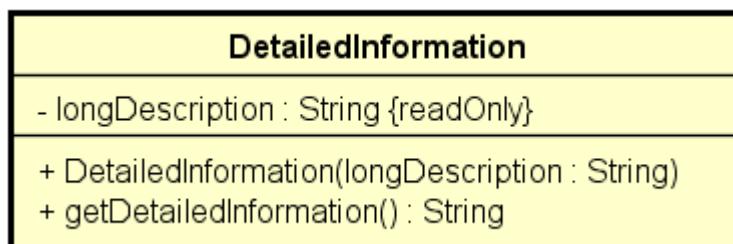


Figura 131: Classe DetailedInformation

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

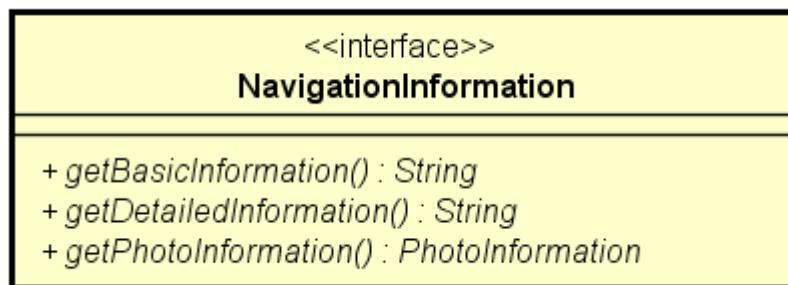
4.4.2.104 model::navigator::graph::navigationinformation::NavigationInformation

Figura 132: Interfaccia NavigationInformation

Nome: NavigationInformation;

Tipo: Interfaccia;

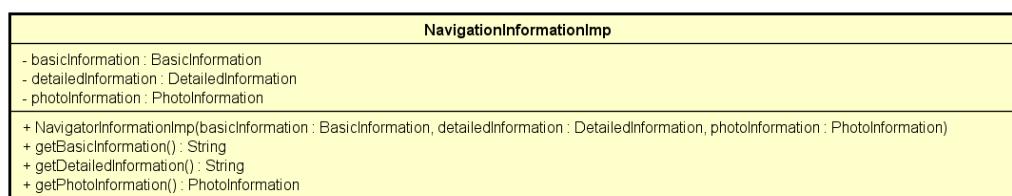
Visibilità: public;

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

4.4.2.105 model::navigator::graph::navigationinformation::NavigationInformationImp**Figura 133:** 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:

- + `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

- + `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

4.4.2.106 model::navigator::graph::navigationinformation::PhotoInformation

PhotoInformation
- photos : Collection
+ PhotoInformation(photoURLs : Collection)
+ getPhotoInformation() : Collection

Figura 134: 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:

- + PhotoInformation()
Costruttore della classe PhotoInformation
- + getPhotoInformation() : Collection<PhotoRef>
Metodo che restituisce una collezione di oggetti PhotoRef rappresentanti le fotografie dell'arco a cui l'oggetto è associato

4.4.2.107 model::navigator::graph::navigationinformation::PhotoRef

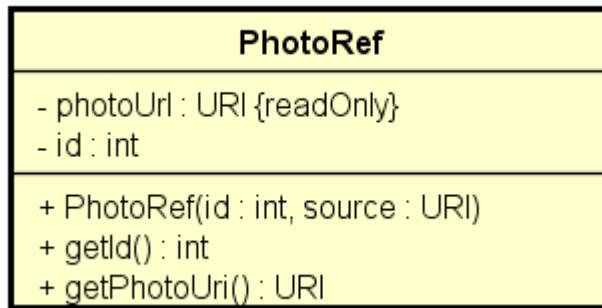


Figura 135: Classe PhotoRef

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:

- **- id : int**
Identificativo della fotografia
- **- photoUrl : URI {readOnly}**
URL di una fotografia

Metodi:

- **+ PhotoRef(id : int, source : URI)**
Costruttore della classe PhotoRef

Argomenti:

- **id : int**
Identificativo della fotografia
- **source : URI**
URL di una fotografia

- **+ getId() : int**
Metodo per recuperare l'identificativo della foto

- + `getPhotoUri() : URI`
Metodo che restituisce l'URL per accedere alla fotografia che l'oggetto rappresenta

4.4.2.108 model::navigator::graph::vertex::Vertex

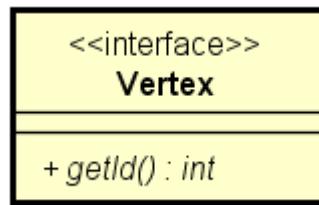


Figura 136: Interfaccia Vertex

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

4.4.2.109 model::navigator::graph::vertex::VertexImp

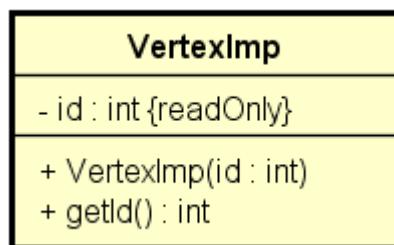


Figura 137: Classe VertexImp

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:

- + VertexImp(id : int)
Costruttore della classe VertexImp

Argomenti:

- id : int
Identificativo numerico di un oggetto VertexImp

- + getId() : int
Override Metodo che ritorna l'identificativo numerico associato all'oggetto VertexImp

4.4.2.110 model::usersetting::DeveloperCodeManager

DeveloperCodeManager	
- DEVELOPER_CODE : String {readOnly}	- sharedPreferences : SharedPreference
+ isValid(code : String) : boolean	+ DeveloperCodeManager(s : SharedPreference)

Figura 138: 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;

Attributi:

- - `DEVELOPER_CODE : String {readOnly}`
Stringa rappresentante la chiave per recuperare le Shared Preferences riguardanti il codice sviluppatore
- - `sharedPreferences : SharedPreferences`
SharedPreferences relative all'applicazione

Metodi:

- + `DeveloperCodeManager(sharedPreferences : SharedPreferences)`
Costruttore della classe DeveloperCodeManager

Argomenti:

- `sharedPreferences : SharedPreferences`
SharedPreferences relative all'applicazione

- + `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

4.4.2.111 model::usersetting::InstructionPreference

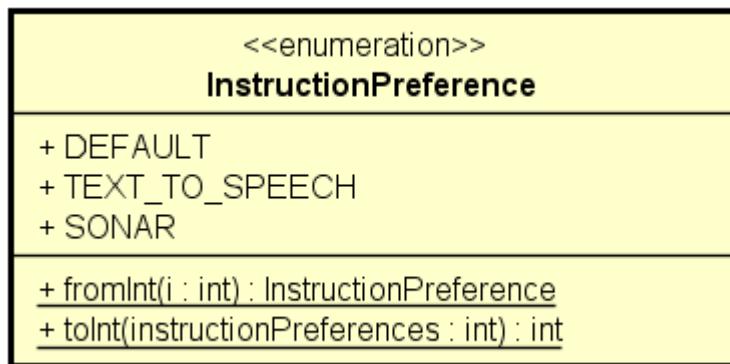


Figura 139: Classe InstructionPreference

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;

Metodi:

- + fromInt(i : int) : InstructionPreference

Metodo per convertire un intero ad una InstructionPreference. Tale metodo è utilizzato per poter recuperare il numero salvato nella memoria del dispositivo che rappresenta la preferenza dell'utente

Argomenti:

- i : int

Intero da convertire ad una InstructionPreference

- + toInt(instructionPreference : InstructionPreference) : int

Metodo per convertire una InstructionPreference in un intero. Tale metodo è utilizzato per rappresentare una preferenze utente in intero per poter essere salvata sulla memoria del dispositivo

Argomenti:

- instructionPreference : InstructionPreference

InstructionPreference da convertire ad intero

4.4.2.112 model::usersetting::PathPreference

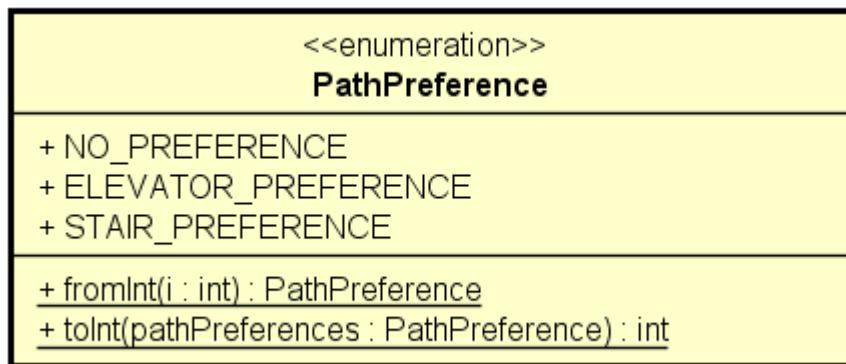


Figura 140: Classe PathPreference

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;

Metodi:

- + fromInt(i : int) : PathPreference

Metodo per convertire un intero ad una PathPreference. Tale metodo è utilizzato per poter recuperare il numero salvato nella memoria del dispositivo che rappresenta la preferenza dell'utente

Argomenti:

- i : int

Intero da convertire a PathPreference

- + toInt(pathPreference : PathPreference) : int

Metodo per convertire una PathPreference ad un intero. Tale metodo è utilizzato per rappresentare una preferenza utente in intero per poter essere salvata sulla memoria del dispositivo

Argomenti:

- pathPreference : PathPreference
PathPreference da convertire ad un intero

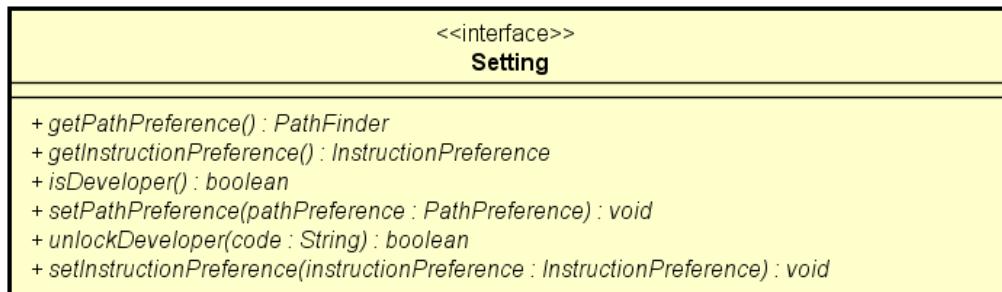
4.4.2.113 model::usersetting::Setting

Figura 141: 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(code : String) : boolean*

Metodo che passato una stringa in ingresso controlla se tale stringa rappresenta un codice sviluppatore valido

Argomenti:

- *code : String*
Stringa rappresentante il codice sviluppatore

4.4.2.114 model::usersetting::SettingImp

SettingImp
<ul style="list-style-type: none"> - <i>pathPreference : PathPreference</i> - <i>instructionPreference : InstructionPreference</i> - <i>PATH_PREFERENCES : String {readOnly}</i> - <i>USER_PREFERENCES : String {readOnly}</i> - <i>DEVELOPER_CODE : String {readOnly}</i> - <i>INSTRUCTION_PREFERENCES : String {readOnly}</i>
<ul style="list-style-type: none"> + <i>getPathPreference() : PathPreference</i> + <i>getInstructionPreference() : InstructionPreference</i> + <i>isDeveloper() : boolean</i> + <i>unlockDeveloper(code : String) : boolean</i> + <i>setPathPreference(pathPreference : PathPreference) : void</i> + <i>setInstructionPreference(instructionPreference : InstructionPreference) : void</i> + <i>SettingImp(context : Context)</i>

Figura 142: 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:

- - `DEVELOPER_CODE : String {readOnly}`
Chiave per accedere al codice sviluppatore contenuto nelle `SharedPreferences`
- - `instructionPreference : InstructionPreference`
Preferenze dell’utente riguardanti le modalità di fruizione delle informazioni per la navigazione
- - `INSTRUCTION_PREFERENCE : String {readOnly}`
Chiave per accedere alle preferenze di fruizione dell’informazione dell’utente contenute nelle `SharedPreferences`
- - `pathPreference : PathPreference`
Preferenze di percorso dell’utente
- - `PATH_PREFERENCES : String {readOnly}`
Chiave per accedere alle preferenze di percorso dell’utente contenute nelle `SharedPreferences`
- - `preferencesEditor : SharedPreferences.Editor`
Editor per modificare le `SharedPreferences` dell’applicazione

- - `sharedPreferences : SharedPreferences`
SharedPreferences relative all'applicazione
- - `USER_PREFERENCES : String {readOnly}`
Chiave per accedere alle preferenze dell'utente contenute nelle SharedPreferences

Metodi:

- + `SettingImp(context : Context)`
Costruttore della classe SettingImp

Argomenti:

- `context : Context`
Context dell'applicazione

- + `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

Stringa rappresentante il codice sviluppatore

4.4.3 presenter

4.4.3.1 presenter::BeaconPos

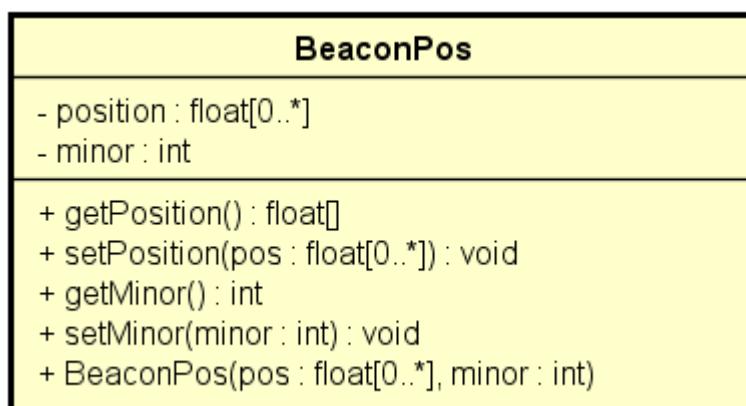


Figura 143: Classe BeaconPos

Nome: BeaconPos;

Tipo: Classe;

Visibilità: public;

Utilizzo: Classe usata per rappresentare la posizione di un Beacon;

Descrizione: Classe che permette di rappresentare la posizione di un Beacon in base al minor;

Attributi:

- - minor : int

Rappresenta il minor di un Beacon

- - position : float[]

Rappresenta la posizione di un Beacon

Metodi:

- **BeaconPos(pos : float[], minor : int)**

Costruttore della classe

Argomenti:

- **pos : float[]**
posizione del Beacon
- **minor : int**
minor del Beacon

- **+ getMinor() : int**

Metodo che ritorna il minor del Beacon

- **+ getPosition() : float[]**

Metodo che ritorna la posizione del Beacon

- **+ setMinor(minor : int) : void**

Metodo che permette di impostare il minor desiderato

Argomenti:

- **minor : int**
minor del Beacon che deve essere impostato

- **+ setPosition(position : float[]) : void**

Metodo che permette di settare la posizione del Beacon

Argomenti:

- **position : float[]**
posizione del Beacon che deve essere impostata

4.4.3.2 presenter::BeaconPowerAreaActivity

BeaconPowerAreaActivity
- view : BeaconPowerArea - map : Collection - scan : boolean - infoManager : InformationManager

onCreate(savedInstanceState : Bundle) : void
+ startScan() : void
+ stopScan() : void
+ isScanning() : boolean
+ getAllVisibleBeacons(visibleBeacon : PriorityQueue) : void
+ buildMap() : void
+ onPause() : void
+ onDatabaseLoaded() : void
+ onLocalMapNotFound() : boolean
+ onRemoteMapNotFound() : void
+ cannotRetrieveRemoteMapDetails() : void
+ noLastMapVersion() : void

Figura 144: Classe BeaconPowerAreaActivity

Nome: BeaconPowerAreaActivity;

Tipo: Classe;

Implementa:

- InformationListener.

Visibilità: public;

Utilizzo: Classe utilizzata per visualizzare l'area coperta dai beacon in base al segnale rss;

Descrizione: Classe che permette di creare un'Activity che è in grado di rilevare il segnale emesso dai Beacon e visualizzarlo a video. Il segnale preso in considerazione è l'RSSI;

Attributi:

- **infoManager : InformationManager**
Oggetto del Model per la gestione delle informazioni
- **- map : Collection<BeaconPowerPos>**
Mappa dei Beacon con relativo segnale e posizione
- **- scan : boolean**
Rappresenta lo stato di scansione. Il valore true indica che la scansione è attiva, false non è attiva
- **- view : BeaconPowerAreaView**
View associata a tale Activity

Metodi:

- **+ buildMap() : void**
Metodo che permette di creare la mappa dei Beacon
- **+ cannotRetrieveRemoteMapDetails() : void**
Override Metodo che viene invocato nel caso in cui non si possono recuperare le informazioni di una mappa in remoto. Non implementato
- **+ getAllVisibleBeacons(visibleBeacons : PriorityQueue<MyBeacon>) : void**
Override Metodo che permette di ottenere tutti i beacon rilevati e di creare una mappa che contiene i Beacon rilevati e i relativi segnali

Argomenti:

- **visibleBeacons : PriorityQueue<MyBeacon>**
lista di beacon rilevati
- **+ isScanning() : boolean**
Metodo che permette di controllare lo stato della scansione. Se true indica che la scansione è attiva, false indica che la scansione non è attiva
- **+ noLastMapVersion() : false**
Override Metodo che viene invocato nel caso in cui la mappa presente in locale non sia aggiornata. Non implementato
- **# onCreate(savedInstanceState : Bundle) : void**
Override Metodo che viene invocato alla creazione dell'oggetto

Argomenti:

- **savedInstanceState : Bundle**
Metodo che viene invocato alla creazione dell'oggetto

- + `onDatabaseLoaded() : void`
Override Metodo che viene invocato al caricamento del database.
Non implementato
- + `onLocalMapNotFound() : boolean`
Override Metodo che viene invocato nel caso in cui la mappa non venga trovata in locale. Non implementato
- + `onPause() : void`
Override Metodo che permette di fermare la scansione ogni volta che l'Activity entra nello stato onPause(). Questo metodo viene chiamato ogni qualvolta viene richiamata un'altra Activity o viene premuto il tasto di blocco del dispositivo
- + `onRemoteMapNotFound() : void`
Metodo che viene invocato nel caso in cui la mappa non venga trovata nel database remoto. Non implementato
- + `startScan() : void`
Metodo che permette di avviare la scansione del segnale dei Beacon
- + `stopScan() : void`
Metodo che permette di terminare la scansione del segnale dei Beacon

4.4.3.3 presenter::BeaconPowerPos

BeaconPowerPos	
-	<code>rssi : int</code>
+	<code>BeaconPowerPos(pos : float[0..*], minor : int)</code>
+	<code>getRssi() : int</code>
+	<code>setRssi(rssi : int) : void</code>

Figura 145: Classe BeaconPowerPos

Nome: BeaconPowerPos;

Tipo: Classe;

Estende:

- BeaconPos.

Visibilità: public;

Utilizzo: Classe usata per rappresentare la posizione e la potenza di un Beacon;

Descrizione: Classe che permette di rappresentare un Beacon con posizione, minor e valore RSSI associato;

Attributi:

- - **rssi** : int
Valore RSSI associato. Valore di default 0

Metodi:

- + **BeaconPowerPos(pos : float, minor : int)**
Costruttore della classe

Argomenti:

- **pos** : float
Posizione del Beacon
- **minor** : int
Minor del Beacon

- + **getRssi() : int**

Metodo che permette di ottenere il valore RSSI del Beacon

- + **setRssi(rssi : int) : void**

Metodo che permette di impostare il valore RSSI associato ad un Beacon

Argomenti:

- **rssi** : int
Valore RSSI da impostare

4.4.3.4 presenter::BlankHomeFragment

BlankHomeFragment
+ BlankHomeFragment()
+ onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View

Figura 146: Classe BlankHomeFragment

Nome: BlankHomeFragment;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.app.Fragment (Android).

Visibilità: public;

Utilizzo: Viene utilizzata nel caso di: Bluetooth non attivo, geolocalizzazione non attiva, nessun beacon conosciuto rilevato, beacon conosciuto rilevato ma nessuna connessione internet presente (e conseguente impossibilità di scaricare la mappa dell'edificio rilevato);

Descrizione: Un BlankHomeFragment viene mostrato dalla HomeActivity in caso di errori nel processo di recupero delle informazioni dell'edificio;

Metodi:

- + BlankHomeFragment()
Costruttore di default
- + onCreateView(inflater : LayoutInflator, container : ViewGroup, savedInstanceState : Bundle) : View
Override Metodo che viene chiamato per fare in modo che il Fragment istanzi la propria UI. Si occupa dell'inflazione

Argomenti:

- inflater : LayoutInflator
oggetto usato per eseguire l'inflate di qualsiasi view nel fragment.
- container : ViewGroup
Parent view a cui è associato il fragment
- savedInstanceState : Bundle
Eventuale stato precedentemente salvato del fragment

4.4.3.5 presenter::CompleteHomeFragment

CompleteHomeFragment	
- searchView : SearchView	
- buildingAddress : TextView	
- buildingName : TextView	
- buildingDescription : TextView	
- buildingOpeningHours : TextView	
- poiCategories : ListView	
+ CompleteHomeFragment()	
+ onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View	
+ setBuildingName(name : String) : void	
+ setBuildingDescription(description : String) : void	
+ setBuildingOpeningHours(hours : String) : void	
+ setBuildingAddress(address : String) : void	
+ setPoiCategoryListAdapter(list : List) : void	

Figura 147: Classe CompleteHomeFragment

Nome: CompleteHomeFragment;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.app.Fragment (Android).

Visibilità: public;

Utilizzo: È utilizzata dalla HomeActivity per mostrare delle informazioni su di un edificio nel caso in cui si riesca a recuperarne le informazioni;

Descrizione: Un CompleteHomeFragment viene mostrato dalla HomeActivity se il processo di recupero delle informazioni dell'edificio è andato a buon fine;

Attributi:

- - buildingAddress : TextView
Riferimento al widget che mostra l'indirizzo dell'edificio rilevato
- - buildingDescription : TextView
Riferimento al widget che mostra gli orari di apertura al pubblico dell'edificio rilevato
- - buildingName : TextView
Riferimento al widget che mostra il nome dell'edificio rilevato
- - buildingOpeningHours : TextView
Riferimento al widget che mostra gli orari di apertura al pubblico dell'edificio rilevato

- - **poiCategories** : `ListView`
Riferimento al widget che mostra le categorie di POI presenti nell'edificio rilevato
- - **searchView** : `SearchView`
Riferimento al widget adibito a search box

Metodi:

- + `CompleteHomeFragment()`
Costruttore di default
- + `onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View`
Override Metodo che viene chiamato per fare in modo che il Fragment istanzi la propria UI. Si occupa dell'inflazione

Argomenti:

- `inflater : LayoutInflater`
Oggetto usato per eseguire linflate di qualsiasi view nel fragment
- `container : ViewGroup`
Parent view a cui è associato il fragment
- `savedInstanceState : Bundle`
L'eventuale stato precedentemente salvato del fragment

- + `setBuildingAddress(address : String) : void`

Metodo che imposta l'indirizzo dell'edificio nel relativo widget

Argomenti:

- `address : String`
Indirizzo dell'edificio

- + `setBuildingDescription(description : String) : void`

Metodo che imposta la descrizione dell'edificio nel relativo widget

Argomenti:

- `description : String`
descrizione dell'edificio

- + `setBuildingName(name : String) : void`

Metodo che imposta il nome dell'edificio nel relativo widget

Argomenti:

- `name : String`
nome dell'edificio

- + `setBuildingOpeningHours(hours : String) : void`
Metodo che imposta le ore di apertura al pubblico dell'edificio, nel relativo widget

Argomenti:

- `hours : String`
orario di apertura

- + `setPoiCategoryListAdapter(list : List<String>) : void`
Metodo che si occupa del binding tra la List, contenente i nomi delle categorie di POI, ed il relativo ArrayAdapter

Argomenti:

- `list : List<String>`
Lista contenente i nomi delle categorie di POI presenti nell'edificio

4.4.3.6 presenter::DetailedInformationActivity

DetailedInformationActivity
- <code>view : DetailedInformationView</code> - <code>processedInfo : ProcessedInformation</code> - <code>imgListFragment : ImageListFragment</code> ~ <code>informationManager : InformationManager</code> ~ <code>navigationmanager : NavigationManager</code>
+ <code>onCreate(bundle : Bundle) : void</code> + <code>updatePhoto() : void</code> + <code>updateDetailedDescription() : void</code> - <code>checkConnection() : boolean</code> + <code>onSupportNavigateUp() : boolean</code>

Figura 148: 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
- - `informationManager` : `InformationManager`
Riferimento utilizzato per accedere alle informazioni trattate dal Model
- - `navigationManager` : `NavigationManager`
Riferimento utilizzato per accedere alle istruzioni di navigazione trattate dal Model
- - `processedInfo` : `ProcessedInformation`
ProcessedInformation associata all'Edge corrente
- - `view` : `DetailedInformationView`
View associata a tale Activity

Metodi:

- + `checkConnection()` : `boolean`
Controlla che sia disponibile una connessione ad Internet
- + `onCreate(bundle : Bundle)` : `void`
Override Metodo che inizializza la DetailedInformationView

Argomenti:

- `bundle` : `Bundle`
Componente per salvare lo stato dell'applicazione
- + `onSupportNavigateUp()` : `boolean`
Override Metodo che viene invocato alla pressione del tasto indietro in alto
- + `updateDetailedDescription()` : `void`
Metodo che aggiorna le informazioni testuali estese visualizzate sulla View

- + updatePhoto() : void
Metodo che aggiorna la foto visualizzata sulla View

4.4.3.7 presenter::DeveloperUnlockerActivity

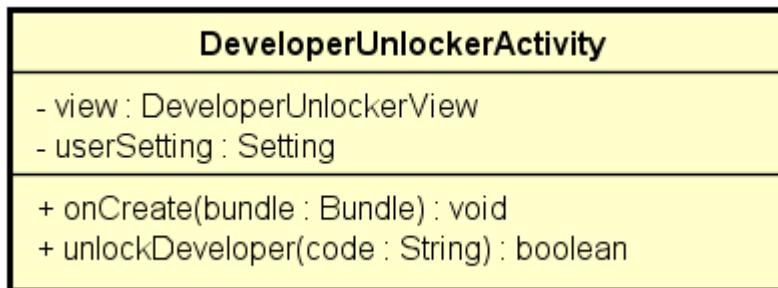


Figura 149: Classe DeveloperUnlockerActivity

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

4.4.3.8 presenter::HelpActivity

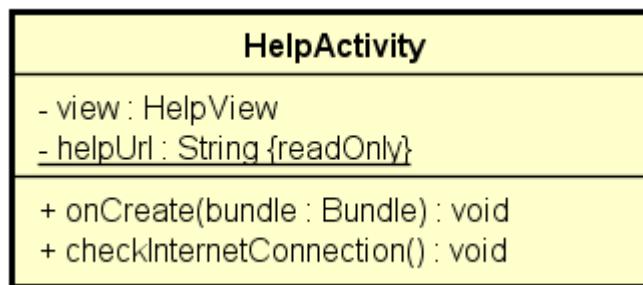


Figura 150: Classe HelpActivity

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:

- - helpUrl : String {readOnly}

Stringa che rappresenta l'URL a cui è possibile recuperare la guida

- - `view : HelpView`
View associata a tale Activity

Metodi:

- - `checkInternetConnection() : void`
Metodo che permette di controllare se la connessione internet è attiva. In caso affermativo interroga la view per mostrare la guida, altrimenti viene mostrato un Alert che informa che la connessione internet non è attiva
- + `onCreate(bundle : Bundle) : void`
Override Metodo che inizializza la HelpView

Argomenti:

- `bundle : Bundle`
Componente per salvare lo stato dell'applicazione

4.4.3.9 presenter::HomeActivity

HomeActivity	
- view : HomeView	
- informationManager : InformationManager	
- noInternetConnection : NoInternetAlert	
+ onCreate(bundle : Bundle) : void	
+ onResume() : void	
+ onBackPressed() : void	
+ onDestroy() : void	
+ updateBuildingName() : void	
+ updateBuildingDescription() : void	
+ updateBuildingOpeningHours() : void	
+ updatePoiCategoryList() : void	
+ updateBuildingAddress() : void	
+ showPoisCategory(categoryName : String) : void	
+ showExplorer() : void	
+ showLocalMaps() : void	
+ showPreferences() : void	
+ showHelp() : void	
+ startNavigation(poiPosition : int) : void	
+ onDatabaseLoaded() : void	
+ onLocalMapNotFound() : boolean	
+ onRemoteMapNotFound() : void	
+ cannotRetrieveMapsDetails() : void	
+ noLastMapVersion() : boolean	
+ getAllVisibleBeacons(visibleBeacons : PriorityQueue) : void	
# onPostResume() : void	
# onStart() : void	
+ checkBluetoothConnection() : void	
+ checkLocationService() : void	
+ checkStoragePermissions() : void	
+ getAllVisibleBeacons(visibleBeacons : PriorityQueue) : void	
+ onDestroy() : void	
+ showAllPois() : void	
+ showDeveloper() : void	

Figura 151: Classe HomeActivity

Nome: HomeActivity;

Tipo: Classe;

Implementa:

- `InformationListener`.

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 AppCompcatActivity per la gestione dell'interazione tra HomeView ed il model;

Attributi:

- - `informationManager` : `InformationManager`
Riferimento utilizzato per accedere alle informazioni trattate dal model
- - `noInternetConnection` : `AlertDialog`
Alert da mostrare nel caso in cui non sia presente connessione internet
- - `view` : `HomeView`
View associata a tale Activity

Metodi:

- + `cannotRetrieveMapsDetails()` : `void`
Override Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui non sia possibile recuperare le informazioni riguardanti le mappe dal database remoto
- + `checkBluetoothConnectio()` : `void`
Controlla che la connettività Bluetooth sia attiva. In caso negativo chiede il permesso di attivarla.
- + `checkLocationService()` : `void`
Controlla che i servizi di Localizzazione siano attivi. In caso negativo chiede all'utente di attivarli.

- + `checkStoragePermissions() : void`
Controlla che i servizi di Localizzazione siano attivi. In caso negativo chiede all'utente di attivarli
- + `getAllVisibleBeacons(visibleBeacons : PriorityQueue<MyBeacon> : void)`
Override Metodo che viene invocato ogni volta che vengono rilevati dei beacon, passando la coda di beacon trovati

Argomenti:

- `visibleBeacons : PriorityQueue<MyBeacon>`
Coda dei beacon rilevati ordinata per potenza

- + `noLastMapVersion() : boolean`
Override Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui la versione della mappa locale differisca dalla versione della mappa in remoto
- + `onBackPressed() : void`
Override Metodo che gestisce il tap dell'utente sul tasto Back. Implementato in modo che chiuda il Drawer se questo è aperto
- + `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

- + `onDatabaseLoaded() : void`
Override Metodo che viene invocato quando il database è stato caricato dalla classe in cui l'oggetto si è registrato come listener
- + `onDestroy() : void`
Override Metodo che viene invocato alla distruzione dell'oggetto
- + `onLocalMapNotFound() : boolean`
Override Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui non sia stata trovata una mappa nel database locale
- + `onRemoteMapNotFound() : void`
Override Metodo che viene invocato dalla classe in cui l'oggetto si è registrato come listener nel caso in cui non sia stata trovata un'acerta mappa nel database remoto

- + `onResume() : void`
Override Recupera le informazioni dell'edificio dal database ed utilizza la View associata per mostrarle all'utente
- # `onResumeFragments() : void`
Override Recupera le informazioni dell'edificio dal database ed utilizza la View associata per mostrarle all'utente.
- # `onStart() : void`
Override Si occupa di controllare che Bluetooth e servizi di Localizzazione siano attivati sul dispositivo
- + `showAllPois() : void`
Avvia l'Activity deputata a mostrare al lista di tutti i POI dell'edificio
- + `showDeveloper() : void`
Avvia l'Activity deputata alla gestione delle funzionalità sviluppatore
- + `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

4.4.3.10 presenter::ImageAdapter

ImageAdapter	
- mContext : Context {readOnly}	
+ ImageAdapter(mContext : Context)	
+ getCount() : int	
+ getItem(position : int) : Object	
+ getItemId(position : int) : long	
+ getView(position : int, convertView : View, parent : ViewGroup) : View	

Figura 152: 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:

- - mContext : Context {readOnly}
Contesto dell'applicativo

Metodi:

- + ImageAdapter(mContext : Context)
Costruttore della classe ImageAdapter

Argomenti:

- mContext : Context
Contesto di esecuzione dell'applicazione

- + 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

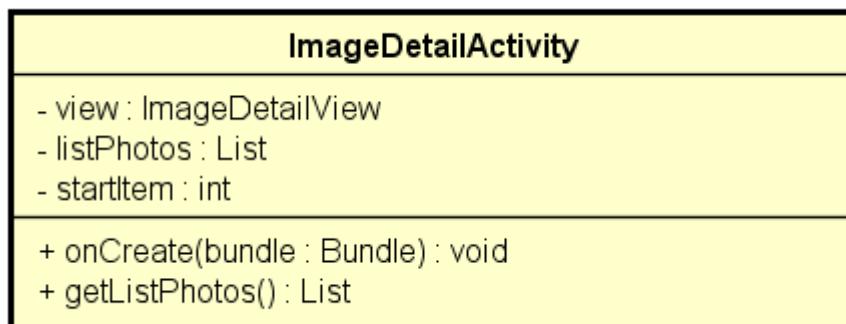
4.4.3.11 presenter::ImageDetailActivity

Figura 153: 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 AppCompcatActivity per la gestione dell’interazione tra ImageDetailView ed il model;

Attributi:

- - `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:

- + `getListPhotos() : List<String>`
Metodo per recuperare la lista di foto
- + `onCreate(bundle : Bundle) : void`
Override Metodo che inizializza ImageDetailView

Argomenti:

- `bundle : Bundle`
Componente per salvare lo stato dell'applicazione

4.4.3.12 presenter::ImageDetailFragment

ImageDetailFragment	
-	<code>mImageNum : int</code>
-	<code>mImageView : ImageView</code>
-	<code>photoUris : List</code>
+	<code>newInstance(photoUris : ArrayList, imageNum : int) : ImageDetailFragment</code>
+	<code>ImageDetailFragment() : void</code>
+	<code>onCreate(savedInstanceState : Bundle) : void</code>
+	<code>onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View</code>
+	<code>onActivityCreated(savedInstanceState : Bundle) : void</code>

Figura 154: Classe ImageDetailFragment

Nome: ImageDetailFragment;

Tipo: Classe;

Componenti delle librerie utilizzate:

- `android.app.Fragment` (Android).

Visibilità: public;

Utilizzo: È utilizzato per fornire le immagini relative ad un certo arco;

Descrizione: Una sottoclasse di Fragment che gestisce e mostra un'immagine tra quelle relative ad una certa istruzione di navigazione;

Attributi:

- - `mImageView : mImageView`
Riferimento al widget usato per mostrare l'immagine
- - `nImageNum : int`
Numero dell'immagine, tra quelle associate ad una particolare istruzione di navigazione, da mostrare
- - `photoUris : List<String>`
Riferimento alla lista di URI delle immagini associate all'istruzione scelta

Metodi:

- + `ImageDetailFragment()`
Costruttore di default della classe ImageDetailFragment
- + `newInstance(photoUris : ArrayList<String>, imageNum : int) : ImageDetailFragment`
Usato per costruire un Fragment attraverso il passaggio di parametri. Best practice consigliata dalla documentazione Android

Argomenti:

- `photoUris : ArrayList<String>`
URI delle immagini associate all'istruzione scelta
 - `imageNum : int`
Numero dell'immagine, tra quelle associate ad una particolare istruzione di navigazione, da mostrare
- + `onActivityCreated(savedInstanceState : Bundle) : void`
Override Metodo che viene invocato quando l'Activity è stata creata e il Fragment è stato istanziato. È utilizzato per la creazione delle anteprime delle fotografie

Argomenti:

- `savedInstanceState : Bundle`
Eventuale stato precedentemente salvato del fragment
- + `onCreate(savedInstanceState : Bundle) : void`
Override Metodo che viene invocato alla creazione dell'oggetto

Argomenti:

- `savedInstanceState : Bundle`
se l'Activity viene re-inizializzata dopo essere stata chiusa, allora questo Bundle contiene i dati più recenti forniti al metodo

- + onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
Override Metodo che viene invocato alla creazione della view corrispondente

Argomenti:

- inflater : LayoutInflater
oggetto LayoutInflater usato per eseguire linflate di qualsiasi view nel fragment
- container : ViewGroup
La parent view a cui è associato il fragment
- savedInstanceState : Bundle
L'eventuale stato precedentemente salvato del fragment

4.4.3.13 presenter::ImageListFragment

ImageListFragment
- imgAdapter : ImageAdapter
- photosUrls : List
- ImageListFragment()
+ newInstance(photosUrls : List) : ImageListFragment
+ onCreate(bundle : Bundle) : void
+ onCreateView(lay : LayoutInflater, viewGr : ViewGroup, bundle : Bundle) : View
+ onItemClick(parent : AdapterView, view : View, position : int, id : long) : void

Figura 155: Classe ImageListFragment

Nome: ImageListFragment;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.app.Fragment (Android);
- android.widget.AdapterView.OnItemClickListener (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 : LayoutInflator, viewGr : ViewGroup, bundle : Bundle) : View`
Override Metodo che crea il layout di un fragment

Argomenti:

- `lay : LayoutInflator`
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(AdapterView<?>, View, position

: int, id : long) : void

Override Metodo che viene invocato alla pressione di un bottone

Argomenti:

– parent : AdapterView<?>

View contenitore della lista di thumbnail

– view : View

View relativa alla lista di thumbnail

– position : int

Posizione nella lista della thumbnail scelta

– id : long

ID della thumbnail scelta

4.4.3.14 presenter::LocalMapActivity

LocalMapActivity	
- view : LocalMapManagerView	
- databaseService : DatabaseService	
# onCreate(bundle : Bundle) : void	
+ updateMap(major : int) : void	
+ deleteMap(major : int) : void	
- LoadMaps() : void	
+ DownloadNewMap() : void	
# onResume() : void	

Figura 156: 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:

- - `databaseService` : `DatabaseService`
Riferimento al model per poter accedere alla gestione delle mappe
- - `view` : `LocalMapManagerView`
View associata a tale Activity

Metodi:

- + `DownloadNewMap()` : `void`
Metodo utilizzato per poter avviare l'activity che si occupa della gestione delle mappe remote
- - `LoadMaps()` : `void`
Metodo utilizzato per leggere le mappe dal database e aggiornare la view
- + `deleteMap(major : int)` : `void`
Metodo che permette di rimuovere una mappa del database locale

Argomenti:

- `major` : `int`
Major della 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

- # `onResume()` : `void`
Override Override del metodo onResume per ricaricare le mappe dopo che l'aktivy è passata in background
- + `updateMap(major : int)` : `void`
Metodo che permette di aggiornare una mappa del database locale

Argomenti:

- **major** : int
Major della mappa da aggiornare

4.4.3.15 presenter::LocalMapAdapter

LocalMapAdapter
<pre>- presenter : LocalMapActivity - collectionBuildingTable : Collection<BuildingTable> - mapVersionStatus : Boolean []</pre>
<pre>+ LocalMapAdapter(presenter : LocalMapActivity, collectionBuildingTable : Collection<BuildingTable>, mapVersionStatus : Boolean []) + getCount() : int + getItem(position : int) : Object + getItemId(position : int) : long + getView(position : int, convertView : View, parent : ViewGroup) : View - showNoUpdateDialog() : void - showUpdateDialog(major : int) : void</pre>

Figura 157: Classe LocalMapAdapter

Nome: LocalMapAdapter;

Tipo: Classe;

Visibilità: public;

Utilizzo: È utilizzata per la gestione dell’interazione tra le mappe installate in locale e le possibili azioni che è possibile effettuare su di esse;

Descrizione: Si occupa del binding tra le informazioni di navigazione fornite dal Model e la View deputata a mostrarle;

Attributi:

- - **collectionBuildingTable** : Collection<BuildingTable>
Collezione di BuildingTable contenente tutte le informazioni associate ad una certa mappa
- - **mapVersionStatus** : Boolean []
Array contenente lo stato di ogni mappa installata. Se vero allora la mappa è da aggiornare, se falso non lo è
- - **presenter** : LocalMapActivity
Riferimento all’activity che si occupa della sua gestione

Metodi:

- + **LocalMapAdapter(presenter : LocalMapActivity , collectionBuildingTable : Collection<BuildingTable>, mapsVersionStatus : boolean [])**
Costruttore della classe LocalMapAdapter

Argomenti:

- **presenter** : LocalMapActivity
Riferimento al model utile per avere anche il contesto dell'applicazione
 - **collectionBuildingTable** : Collection<BuildingTable>
Insieme di mappe installate sul dispositivo
 - **mapsVersionStatus** : boolean []
Array contenente lo stato delle mappe
- + **getCount()** : int
Override Restituisce il numero di mappe installate nel telefono
 - + **getItem(position : int)** : Object
Override Restituisce la mappa della collezione che si trova nella posizione fornita come parametro

Argomenti:

- **position** : int
Posizione della mappa
- + **getItemId()** : long
Override Restituisce l'id della mappa della collezione che si trova nella posizione fornita come parametro fornita come parametro
 - + **getView(position : int, convertView : View, parent : ViewGroup)** : View
Override Metodo che viene utilizzare per recuperare la view di una mappa in una certa posizione

Argomenti:

- **position** : int
Posizione delle view da recuperare
 - **convertView** : View
View dalla quale recuperare la mappa cercata
 - **parent** : ViewGroup
Layout della lista di mappe
- - **showNoUpdateDialog()** : void
Metodo utilizzato per mostrare il dialog che indica che la mappa è già aggiornata
 - - **showUpdateDialog()** : void
Metodo utilizzato per mostrare un dialog che chiede la conferma dell'utente per aggiornare la mappa

4.4.3.16 presenter::LoggingActivity

LoggingActivity	
- infoManager : InformationManager	
- view : LoggingView	
+ stopLogging() : void	
+ getAllVisibleBeacons(visibleBeacons : PriorityQueue) : void	
+ onBackPressed() : void	
+ onOptionsItemSelected(MenuItem item : int) : boolean	
+ onCreate(savedInstanceState : Bundle) : void	

Figura 158: Classe LoggingActivity

Nome: LoggingActivity;

Tipo: Classe;

Implementa:

- InformationListener.

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:

- + getAllVisibleBeacons(visibleBeacons : PriorityQueue<MyBeacon>)
: void

Override Metodo invocato ogni volta che vengono rilevati beacon che consente di avere la lista aggiornata dei Beacon rilevati

Argomenti:

- visibleBeacons : PriorityQueue<MyBeacon>
lista di beacon rilevati

- + onBackPressed() : void

Override Redefinizione del comportamento di default del tasto upBack per fermare la registrazione di un log

- + onCreate(savedInstanceState : Bundle)

Override Metodo che viene chiamato quando si sta avviando l'activity. Questo metodo si occupa di inizializzare i campi dati.

Argomenti:

- savedInstanceState : Bundle
Stato precedente dell'Activity

- + onOptionsItemSelected(item : MenuItem) : void

Override Redefinizione del comportamento di default del tasto upBack per fermare la registrazione di un log

Argomenti:

- item : MenuItem
Redefinizione del comportamento di default del tasto up-
Back per fermare la registrazione di un log

- + stopLogging() : void

Metodo che viene utilizzato per interrompere l'attività di log

4.4.3.17 presenter::LogInformationActivity

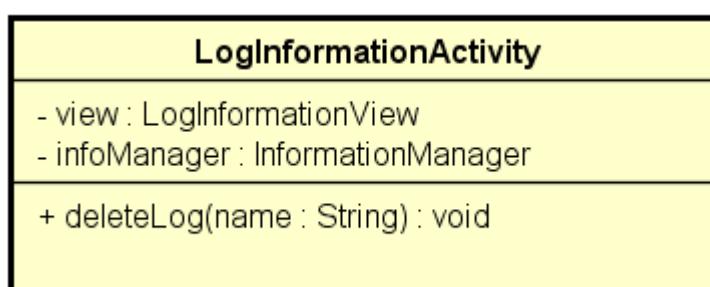


Figura 159: Classe LogInformationActivity

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

4.4.3.18 presenter::MainActivity

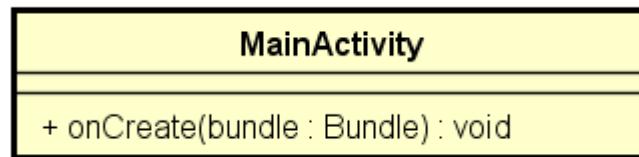


Figura 160: Classe MainActivity

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;

Metodi:

- + onCreate(bundle : Bundle) : void
Override Metodo che inizializza la HomeView

Argomenti:

- bundle : Bundle
Componente per salvare lo stato dell'applicazione

4.4.3.19 presenter::MainDeveloperActivity

MainDeveloperActivity
- view : MainDeveloperView - infoManager : InformationManager
+ onCreate(bundle : Bundle) : void + showDetailedLog(logPosition : int) : void + startNewLog() : void + onBackPressed() : void

Figura 161: 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 : MainDeveloperView
View associata a tale Activity

Metodi:

- + onBackPressed() : void
Override Metodo utilizzato per modificare il comportamento di default del tasto back
- + 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

4.4.3.20 presenter::MainDeveloperPresenter

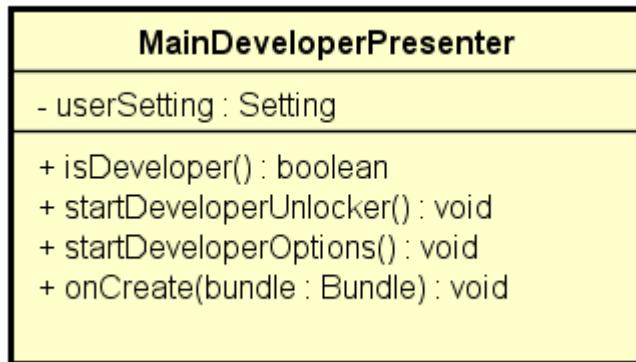


Figura 162: Classe MainDeveloperPresenter

Nome: MainDeveloperPresenter;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.support.v7.app.AppCompatActivity (Android).

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;

Attributi:

- - userSetting : Setting
Preferenze dell'utente

Metodi:

- + isDeveloper() : boolean
Metodo che permette di verificare se un utente è sviluppatore
- + onCreate(bundle : Bundle) : void
Override Metodo che viene invocato alla creazione dell'oggetto

Argomenti:

- bundle : Bundle
Stato dell'oggetto che è stato salvato

- + `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

4.4.3.21 presenter::MyApplication

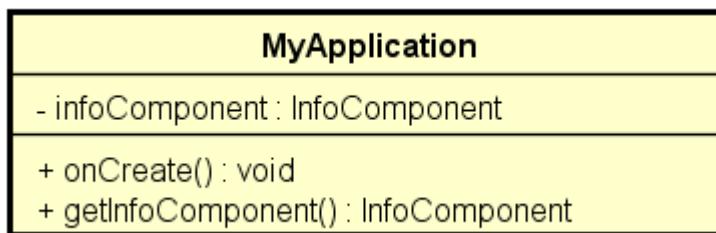


Figura 163: Classe MyApplication

Nome: MyApplication;

Tipo: Classe;

Componenti delle librerie utilizzate:

- `android.app.Application` (Android).

Visibilità: public;

Utilizzo: Viene utilizzata per gestire l'inizializzazione della dependency injection;

Descrizione: Classe derivata da Application. Essa permette di ridefinire il comportamento dell'applicazione al momento della creazione tramite il metodo onCreate;

Attributi:

- - `infoComponent : InfoComponent`

Riferimento all'oggetto di tipo InfoComponent che permette di risolvere le dipendenze tra i tipi presenti nell'applicazione

Metodi:

- + getInfoComponent() : InfoComponent
Metodo getter che permette di recuperare il riferimento a infoComponent. Questo metodo è utile per poter eseguire il metodo inject nella classi richieste
- + onCreate() : void
Override Metodo di callback che permette di ridefinire il comportamento dell'applicazione al momento della creazione. Il metodo viene ridefinito per poter gestire la creazione dell'oggetto di tipo InfoComponent utile alla gestione della dependency injection

4.4.3.22 presenter::NavigationActivity

NavigationActivity	
- view : NavigationView	
- navigationInstruction : List	
- informationManager : InformationManager	
- navigationManager : NavigationManager	
- poild : int	
- compass : Compass	
- dialogPathError : AlertDialog	
- builder : AlertDialog.Builder	
- noInternetConnection : NoInternetAlert	
+ onCreate(bundle : Bundle) : void	
+ pathError() : void	
+ informationUpdate(info : ProcessedInformation) : void	
+ showDetailedInformation(instructionPosition : int) : void	
+ stopNavigation() : void	
# onNewIntent(intent : Intent) : void	
# handleIntent(intent : Intent) : void	
# onSaveInstanceState(outState : Bundle) : void	
# onDestroy() : void	

Figura 164: 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:

- - `builder` : `AlertDialog.Builder`
Oggetto che si occupa della creazione dell'alert di avviso in caso di percorso errato
- - `dialogPathError` : `AlertDialog`
Alert che viene mostrato nel caso un cui l'utente sbagli percorso
- - `informationManager` : `InformationManager`
Interfaccia che permette l'accesso alle informazioni trattate nel package "model"
- - `navigationInstruction` : `List<ProcessedInformation>`
Riferimento alla lista di istruzioni di navigazione
- - `navigationManager` : `NavigationManager`
Interfaccia che si occupa di esporre tutti i metodi utili alla navigazione
- - `noInternetConnection` : `AlertDialog`
Alert da mostrare nel caso in cui non sia presente connessione internet
- - `poiId` : `int`
Id del POI che l'utente ha indicato come destinazione
- - `view` : `NavigationView`
View associata a tale Activity

Metodi:

- `# handleIntent(intent : Intent) : void`
Override Metodo che gestisce l'`Intent` lanciato dalla ricerca nominativa di un POI

Argomenti:

– intent : Intent

Intent attraverso il quale è stata creata la corrente istanza
di NavigationActivity

- + 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

- # onDestory()
Override Metodo che viene invocato alla distruzione dell'Activity
- # onNewIntent(intent : Intent) : void
Override Metodo che viene invocato con un nuovo Intent

Argomenti:

– intent : Intent

Intent con il quale è stata avviata al corrente Activity

- + onSaveInstanceState() : void
Override Salva lo stato corrente dell'Activity, in modo da poterlo
ripristinare in caso di interruzione coatta della stessa
- + 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

4.4.3.23 presenter::NavigationAdapter

NavigationAdapter	
- context : Context	
- navigationInformation : List	
+ getCount() : int	
+ getItem(position : int) : Object	
+ getItemId(position : int) : long	
+ getView(position : int, convertView : View, parent : ViewGroup) : View	
+ NavigationAdapter(context : Context, navigationInformation : List)	
+ setDirectionArrow(direction : NavigationDirection, image : ImageView) : void	
+ isEnabled(position : int) : boolean	

Figura 165: 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;

Attributi:

- - context : Context
Context dell'applicazione
- - navigationInformation : List<ProcessedInformation>
Lista di informazioni necessarie a raggiungere la destinazione

Metodi:

- + NavigationAdapter(context : Context, navigationInformation : List<ProcessedInformation>)
Costruttore della classe NavigationAdapter

Argomenti:

- `context : Context`
Context dell'applicazione
- `navigationInformation : List<ProcessedInformation>`
Lista di informazioni necessarie a raggiungere la destinazione
- + `NavigationAdapter()`
Costruttore della classe NavigationAdapter
- + `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
- + `isEnabled(position : int) : boolean`
Override Metodo che permette di verificare se un certo elemento della lista è attivo oppure no

Argomenti:

– position : int

Posizione dell'elemento

- - setDirectionArrow(direction : int, image : ImageView) : void

Metodo di utilità che associa la freccia corretta (Drawable) alla istruzione, in modo che ne indichi visivamente la direzione da seguire

Argomenti:

– direction : int

Indicatore della direzione

– image : ImageView

View da aggiornare con l'immagine

4.4.3.24 presenter::NearbyPoiActivity

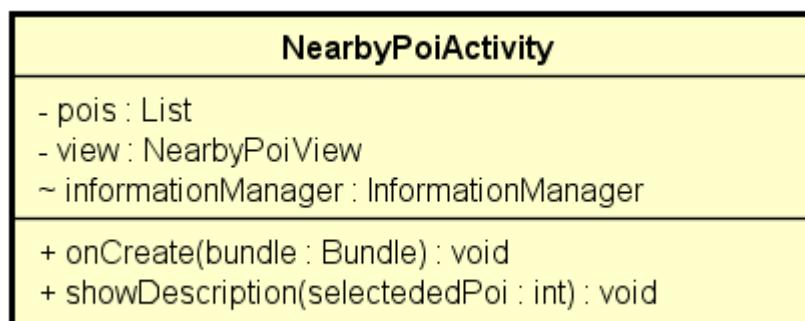


Figura 166: Classe NearbyPoiActivity

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:

- **informationManager : InformationManager**
Interfaccia che permette l'accesso alle informazioni trattate nel package "model"
- **- 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

- **+ showDescription(selectedPoi : int) : void**
Metodo che recupera l'id del POI scelto e lo passa a DescriptionPoiActivity, in modo che essa

Argomenti:

- **selectedPoi : int**
POI di cui visualizzare la descrizione

4.4.3.25 presenter::NoInternetAlert

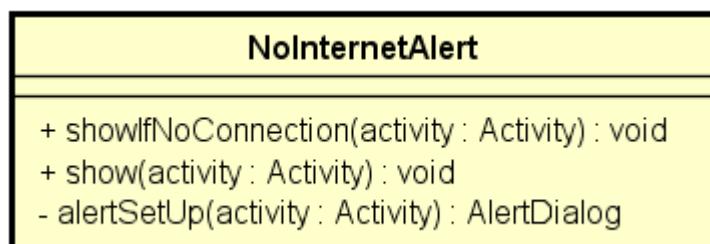


Figura 167: Classe NoInternetAlert

Nome: NoInternetAlert;

Tipo: Classe;

Visibilità: public;

Utilizzo: Utilizzata per visualizzare l'avviso di connessione mancante all'utente;

Descrizione: Classe per creare alert che avvisano che non è disponibile la connessione internet;

Metodi:

- - `alertSetUp(activity : Activity) : AlertDialog`
Metodo che si occupa della creazione dell'Alert da mostrare

Argomenti:

- `activity : Activity`
Contesto in cui deve essere mostrato l'Alert

- + `show(activity : Activity) : void`

Metodo che permette di mostrare l'Alert senza controllare se la connessione internet sia attiva

Argomenti:

- `activity : Activity`
Contesto in cui deve essere mostrato l'alert

- + `showIfNoConnection(activity : Activity) : void`

Metodo che controlla se presente la connessione internet. Se non presente viene mostrato un Alert

Argomenti:

- `activity : Activity`
Contesto in cui deve essere mostrato l'Alert

4.4.3.26 presenter::PoiActivity

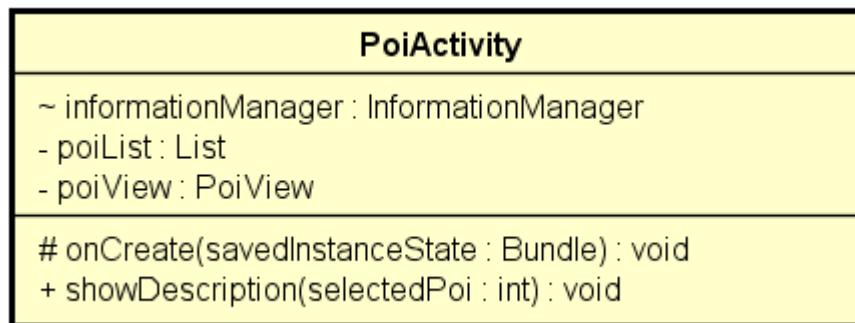


Figura 168: Classe PoiActivity

Nome: PoiActivity;

Tipo: Classe;

Visibilità: public;

Utilizzo: Utilizzata per la visualizzazione dei POI;

Descrizione: PoiCategoryActivity si occupa di gestire le categorie di POI presenti nell'edificio, in modo che l'utente possa effettuare la ricerca della destinazione per categoria;

Attributi:

- `informationManager : InformationManager`
Riferimento utilizzato per accedere alle informazioni trattate dal model
- `- poiList : List<PointOfInterest>`
Lista di POI associati ad una certa categoria
- `- view : PoiView`
View associata a tale Activity

Metodi:

- `# onCreate(savedInstanceState : Bundle)`
Override Chiamato quando si sta avviando l'activity. Questo metodo si occupa di inizializzare i campi dati.

Argomenti:

- `savedInstanceState : Bundle`
se l'Activity viene re-inizializzata dopo essere stata chiusa, allora questo Bundle contiene i dati più recenti forniti al metodo
- + `showDescription(selectedPoi : int) : void`
Metodo che recupera l'id del POI scelto e lo passa a DescriptionPoiActivity, in modo che essa possa visualizzare la descrizione del Poi scelto.

Argomenti:

- `selectedPoi : int`
POI di cui visualizzare la descrizione

4.4.3.27 presenter::PoiCategoryActivity

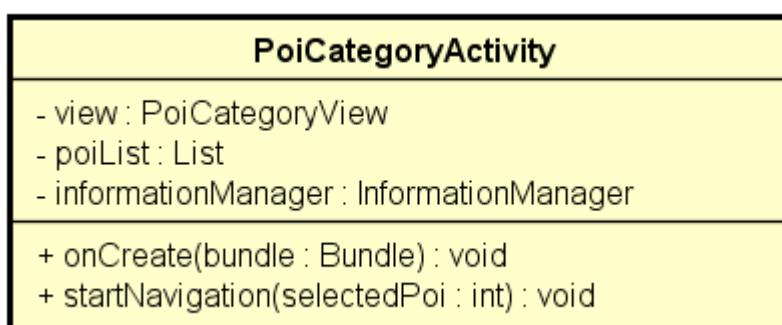


Figura 169: Classe PoiCategoryActivity

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:

- - informationManager : InformationManager
Interfaccia che permette l'accesso alle informazioni trattate nel package "model"
- - 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

4.4.3.28 presenter::PoiDescriptionActivity

PoiDescriptionActivity
~ informationManager : InformationManager - view : DescriptionView
onCreate(savedInstanceState : Bundle) : void + onNavigateUp() : boolean

Figura 170: Classe PoiDescriptionActivity

Nome: PoiDescriptionActivity;

Tipo: Classe;

Visibilità: public;

Utilizzo: Utilizzata per la gestione della descrizione dei POI circostanti;

Descrizione: PoiDescriptionActivity è la classe che si occupa di gestire la descrizione di un POI presente nell'edificio associato al beacon rilevato, in modo che vengano fornite all'utente le informazioni presenti nel database associate a quello specifico POI;

Attributi:

- - informationManager : InformationManager
Riferimento utilizzato per accedere alle informazioni trattate dal Mode
- - view : DescriptionView
Riferimento alla relativa View

Metodi:

- # onCreate(savedInstanceState : Bundle) : void
Override Chiamato quando si sta avviando l'activity. Questo metodo si occupa di inizializzare i campi dati

Argomenti:

- savedInstanceState : Bundle
Se l'Activity viene re-inizializzata dopo essere stata chiusa, allora questo Bundle contiene i dati più recenti forniti al metodo

- + onNavigateUp() : boolean
Override Metodo che ridefinisce il comportamento del bottone di navigazione

4.4.3.29 presenter::PreferencesActivity

PreferencesActivity
- view : PreferencesView
- setting : Setting
onCreate(savedInstanceState : Bundle) : void
+ savePreferences() : void
onPostCreate(savedInstanceState : Bundle) : void
+ onNavigateUp() : boolean

Figura 171: 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:

- **setting** : Setting
Riferimento alle preferenze utente
- - **view** : PreferencesView
View associata a tale Activity

Metodi:

- + **onCreate(bundle : Bundle) : void**
Override Metodo che inizializza la View associata

Argomenti:

- **bundle : Bundle**
Componete per salvare lo stato dell'applicazione

- `# onResume() : void`
Override Metodo che viene invocato ogni volta che si accede all'activity. Ha il compito di impostare le preferenze sulla view
- `+ savePreferences() : void`
Metodo che permette di salvare le preferenze utente

4.4.3.30 presenter::RemoteMapManagerActivity

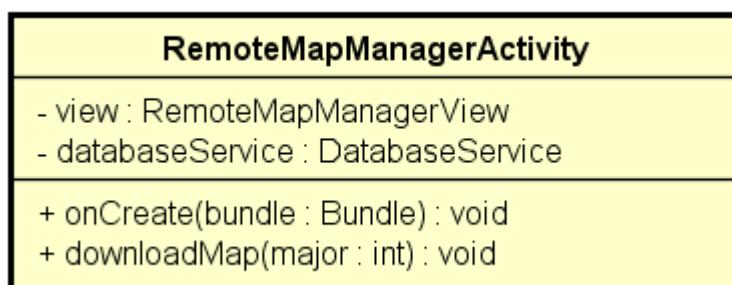


Figura 172: Classe RemoteMapManagerActivity

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:

- `- databaseService : DatabaseService`
Riferimento al model per poter accedere alla gestione delle mappe
- `- view : RemoteMapManagerView`
View associata a tale Activity

Metodi:

- + `downloadMap(major : int) : void`

Metodo che permette di eseguire il download di una mappa da un database remoto

Argomenti:

- `major : int`

Major 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

4.4.3.31 presenter::RemoteMapManagerAdapter

RemoteMapManagerAdapter
- presenter : RemoteMapManagerActivity - buildingTables : Collection<BuildingTable>
+ RemoteMapManagerAdapter(presenter : RemoteMapManagerActivity, collectionBuildingTable : Collection<BuildingTable>) + getCount() : int + getItem(position : int) : Object + getItemId(position : int) : long + getView(position : int, convertView : View, parent : ViewGroup) : View

Figura 173: Classe RemoteMapManagerAdapter

Nome: RemoteMapManagerAdapter;

Tipo: Classe;

Visibilità: public;

Utilizzo: È utilizzata per la gestione dell'interazione tra le mappe installate disponibili e le possibili azioni che è possibile effettuare su di esse;

Descrizione: Si occupa del binding tra le mappe fornite dal Model e la View deputata a mostrarle ;

Attributi:

- - `buildingTables : Collection<BuildingTable>`

Collezione di BuildingTable contenente tutte le informazioni associate ad una certa mappa

- - **presenter** : `RemoteMapManagerActivity`
Riferimento all'activity che si occupa della sua gestione

Metodi:

- + `RemoteMapManagerAdapter(presenter : RemoteMapManagerActivity, buildingTables : Collection<BuildingTable>)`
Costruttore della classe `RemoteMapAdapter`

Argomenti:

- **presenter** : `RemoteMapManagerActivity`
Riferimento al presenter utile per avere anche il contesto dell'applicazione
- **buildingTables** : `Collection<BuildingTable>`
Insieme di mappe disponibili

- + `getCount() : int`

Override Restituisce il numero di mappe disponibili

- + `getItem(position : int) : Object`

Override Restituisce la mappa della collezione che si trova nella posizione fornita come parametro

Argomenti:

- **position** : `int`
Posizione della mappa

- + `getItemId(position : int) : Object`

Override Restituisce l'id della mappa della collezione che si trova nella posizione fornita come parametro fornita come parametro

Argomenti:

- **position** : `int`
Posizione della mappa

- + `getView(position : int, convertView : View, parent : ViewGroup) : View`

Override Metodo che viene utilizzare per recuperare la view di una mappa in una certa posizione

Argomenti:

- **position** : `int`
Posizione delle view da recuperare
- **convertView** : `View`
View dalla quale recuperare la mappa cercata
- **parent** : `ViewGroup`
Layout della view cercata

4.4.3.32 presenter::SearchSuggestionsProvider

SearchSuggestionsProvider
<pre> - COLUMNS : String - informationManager : InformationManager - pois : Collection + query(Uri : Uri, projection : String[0..*], selection : String, selectionArgs : String[0..*], sortOrder : String) : Cursor + getType(Uri : Uri) : String + insert(Uri : Uri, ContentValues) : Uri + update(Uri : Uri, ContentValues, selection : String, selectionArgs : String[0..*]) : int + onCreate() : void </pre>

Figura 174: 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
- informationManager : InformationManager
Riferimento utilizzato per accedere alle informazioni trattate dal model
- - pois : Collection<PointOfInterest>
Insieme di tutti i PointOfInterest di un edificio

Metodi:

- + delete(Uri : Uri, String, String[]) : int

Override Metodo che serve per eliminare elementi dal Content Provider

Argomenti:

- **uri** : Uri
URI su cui fare la query
- **selection** : String
Criterio da applicare per filtrare le righe del content provider
- **selectionArgs** : String[]
Insieme di argomenti su cui fare la selezione

- + **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 : Uri, values : 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() : boolean**

Override Metodo che inizializza un oggetto di tipo SearchSuggestionProvider

- + **query(uri : Uri, projection : String[], selection : String, selectionArgs : String[], sortOrder : 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, values : ContentValues, selection : String, selectionArgs : String[]) : 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

4.4.4 view

4.4.4.1 view::BeaconPowerArea

BeaconPowerArea	
- <u>background : boolean</u>	
- <u>map : Collection</u>	
- <u>updateMap : Collection</u>	
+ <u>BeaconPowerArea(context : Context)</u>	
+ <u>BeaconPowerArea(context : Context, attrs : AttributeSet)</u>	
+ <u>onDraw(canvas : Canvas) : void</u>	
+ <u>drawBeacon(canvas : Canvas) : void</u>	
+ <u>drawBeaconPower(canvas : Canvas) : void</u>	
+ <u>setMap(map : Collection) : void</u>	
+ <u>setUpdateMap(updateMap : Collection) : void</u>	

Figura 175: Classe BeaconPowerArea

Nome: BeaconPowerArea;

Tipo: Classe;

Visibilità: public;

Utilizzo: Classe utilizzata per la visualizzazione dei segnali dei Beacon all'interno di una mappa;

Descrizione: Classe ausiliaria che permette definire una View in grado di aggiornarsi in tempo reale attraverso l'aiuto dei Canvas. La classe in questione permette di visualizzare una mappa dell'edificio sotto forma di immagine e i Beacon con i relativi segnali.;

Attributi:

- - background : boolean
Booleano che indica se la mappa dell'edificio è stata caricata
- - map : Collection<BeaconPowerPos>
Mappa dei Beacon
- - updateMap : Collection<BeaconPowerPos>
Mappa dei Beacon con valore del segnale rssi aggiornato

Metodi:

- + BeaconPowerArea(context : Context)
Costruttore della classe BeaconPowerArea

Argomenti:

- context : Context
Contesto dell'applicazione

- + BeaconPowerArea(context : Context, attrs : AttributeSet)
Costruttore della class BeaconPowerArea

Argomenti:

- context : Context
Contesto dell'applicazione
- attrs : AttributeSet
Attributi che permettono di configurare la View

- + drawBeacon(canvas : Canvas) : void
Metodo che permette di disegnare i Beacon sulla View

Argomenti:

- canvas : Canvas
Oggetto Canvas che fornisce le primitive per poter disegnare

- + **drawBeaconPower(canvas : Canvas) : void**
Metodo che permette di disegnare l'area del segnale rssи coperta da ogni Beacon

Argomenti:

- canvas : Canvas
Oggetto Canvas che fornisce le primitive per poter disegnare

- # **onDraw(canvas : Canvas) : void**

Override Metodo che permette di definire il comportamento dell'aggiornamento della View

Argomenti:

- canvas : Canvas
Oggetto Canvas fornito dall'Applicazione

- + **setMap(map : Collection<BeaconPowerPos>) : void**

Metodo che permette di settare la mappa dei Beacon

Argomenti:

- map : Collection<BeaconPowerPos>
mappa dei Beacon

- + **setUpDateMap(updateMap : Collection<BeaconPowerPos>) : void**

Metodo che permette di settare la mappa dei Beacon con il segnale rssи rilevato

Argomenti:

- updateMap : Collection<BeaconPowerPos>
Mappa dei Beacon rilevati con il relativo segnale rssи

4.4.4.2 view::BeaconPowerAreaView

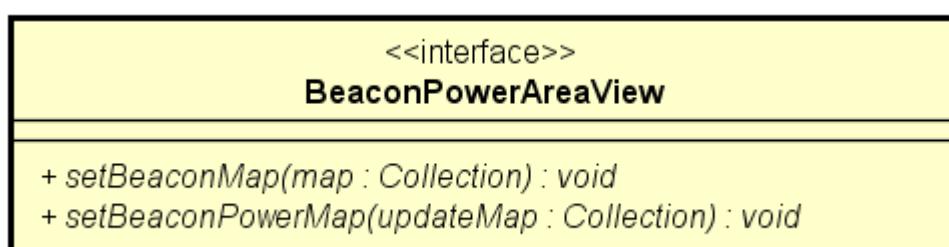


Figura 176: Interfaccia BeaconPowerAreaView

Nome: BeaconPowerAreaView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Interfaccia utilizzata per settare i dati della view a cui fa riferimento;

Descrizione: Interfaccia che espone i metodi per poter aggiornare la mappa dei Beacon con il segnale rssl rilevato;

Metodi:

- + setBeaconMap(map : Collection<BeaconPowerPos>) : void
Metodo che permette di impostare la mappa dei Beacon

Argomenti:

- map : Collection<BeaconPowerPos>
Mappa dei Beacon

- + setBeaconPowerMap(updateMap : Collection<BeaconPowerPos>) : void

Metodo che permette di impostare la mappa dei Beacon rilevati con il segnale rssl aggiornato

Argomenti:

- updateMap : Collection<BeaconPowerPos>
Mappa dei Beacon rilevati con valore rssl aggiornato

4.4.4.3 view::BeaconPowerAreaViewImp

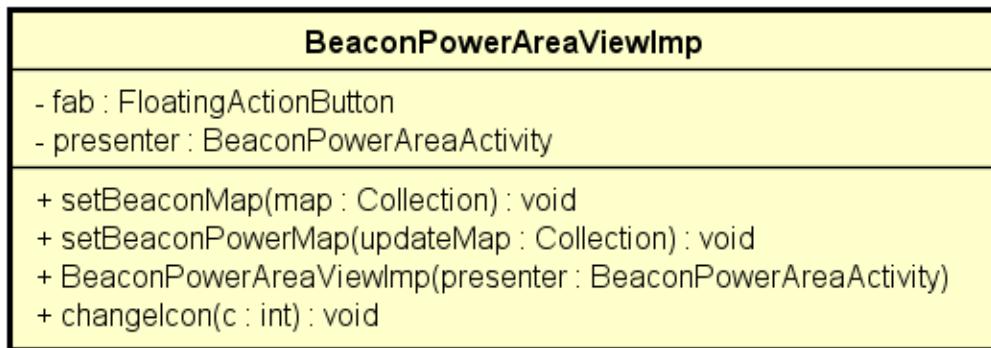


Figura 177: Classe BeaconPowerAreaViewImp

Nome: BeaconPowerAreaViewImp;

Tipo: Classe;

Implementa:

- BeaconPowerAreaView.

Visibilità: public;

Utilizzo: Classe utilizzata per la visualizzazione della mappa dei Beacon con relativa potenza;

Descrizione: Classe che implementa BeaconPowerView e fornisce una vista alla classe BeaconPowerActivity. Essa è in grado di rappresentare una mappa dell'edificio in cui si sta navigando e l'area coperta dal segnale dei beacon sotto forma di cerchio.;

Attributi:

- - fab : FloatingActionButton
Bottone per avviare o interrompere la scansione del segnale dei Beacon
- - presenter : BeaconPowerAreaActivity
Presenter della View

Metodi:

- + BeaconPowerAreaViewImp(presenter : BeaconPowerAreaActivity)
Costruttore della classe BeaconPowerAreaViewImp

Argomenti:

- presenter : BeaconPowerAreaActivity
Presenter della View che viene creata

- + changeIcon(c : int) : void
Metodo che permette di cambiare l'icona del pulsante che permette l'avvio o l'interruzione della scansione dei Beacon

Argomenti:

- c : int
Intero che rappresenta lo stato del bottone FAB

- + setBeaconMap(map : Collection<BeaconPowerPos>) : void
Override Metodo che permette di impostare la mappa dei Beacon

Argomenti:

- map : Collection<BeaconPowerPos>
Mappa dei Beacon

- + `setBeaconPowerMap(updateMap : Collection<BeaconPowerPos>)`

: void

Override Metodo che permette di impostare la mappa dei Beacon rilevati con il segnale rssi aggiornato

Argomenti:

- `updateMap : Collection<BeaconPowerPos>`

Mappa dei Beacon rilevati con valore rssi aggiornato

4.4.4.4 view::DescriptionView

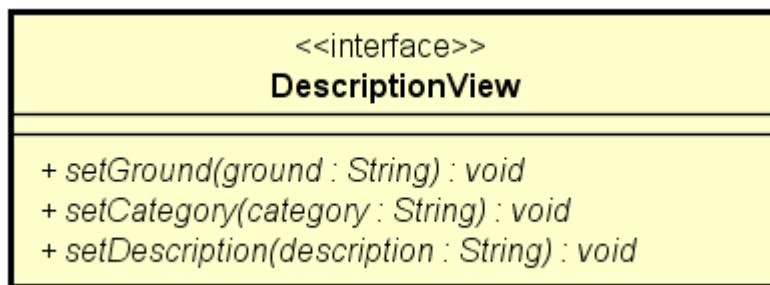


Figura 178: Interfaccia DescriptionView

Nome: `DescriptionView`;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Utilizzata per le informazioni riguardanti un singolo POI;

Descrizione: DescriptionView espone i metodi utili per aggiornare la UI con le informazioni riguardanti un singolo POI.;

Metodi:

- + `setCategory(category : String) : void`

Imposta la categoria del POI nel widget deputato a mostrarlo

Argomenti:

- `category : String`

Categoria del POI

- + `setDescription(description : String) : void`

Imposta la descrizione del POI nel widget deputato a mostrarla

Argomenti:

– description : String

 Descrizione del POI

- + setGround(ground : String) : void

 Imposta l'identificativo del piano nel widget deputato a mostrarlo

Argomenti:

– ground : String

 Identificativo del piano

4.4.4.5 view::DescriptionViewImp

DescriptionViewImp
- presenter : PoiDescriptionActivity
+ DescriptionViewImp(presenter : PoiDescriptionActivity)
+ setGround(ground : String) : void
+ setCategory(category : String) : void
+ setDescription(description : String) : void

Figura 179: Classe DescriptionViewImp

Nome: DescriptionViewImp;

Tipo: Classe;

Implementa:

- DescriptionView.

Visibilità: public;

Utilizzo: Classe utilizzata per la visualizzazione della descrizione dei POI;

Descrizione: DescriptionView espone i metodi utili per aggiornare la UI con le informazioni riguardanti un singolo POI;

Attributi:

- - presenter : PoiDescriptionActivity
 Riferimento al presenter

Metodi:

- + **DescriptionViewImp()**
Costruttore della classe DescriptionViewImp
- + **setCategory(category : String) : void**
Override Imposta la categoria del POI nel widget deputato a mostrarlo

Argomenti:

- **category : String**
Categoria del POI

- + **setDescription(description : String) : void**
Override Imposta la descrizione del POI nel widget deputato a mostrarlo

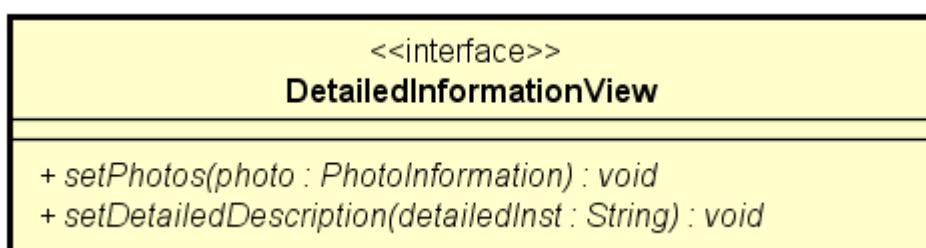
Argomenti:

- **description : String**
Descrizione del POI

- + **setGround(ground : String) : void**
Override Imposta l'identificativo del piano nel widget deputato a mostrarlo

Argomenti:

- **ground : String**
Identificativo del piano

4.4.4.6 view::DetailedInformationView**Figura 180:** 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 che imposta una stringa formattata in HTML come testo della TextView dedicata a mostrare l'istruzione dettagliata

Argomenti:

- detailedInst : String
Stringa contenente l'istruzione dettagliata formattata come HTML

- + *setPhoto(photo : PhotoInformation) : void*
Metodo utilizzato per visualizzare la lista delle anteprime delle foto relative ad un certo POI

Argomenti:

- photo : PhotoInformation
Oggetto che contiene la lista degli URI delle immagini relative ad un certo POI

4.4.4.7 view::DetailedInformationViewImp

DetailedInformationViewImp
- presenter : DetailedInformationActivity
+ setPhotos(urls : List) : void
+ setDetailedDescription(detailedInst : String) : void
+ DetailedInformationViewImp(presenter : DetailedInformationActivity)

Figura 181: 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: `DetailedInformationViewImp` si occupa di gestire direttamente i widget della UI deputati a mostrare le informazioni dettagliate rispetto ad una certa istruzione di navigazione. Tali informazioni comprendono le immagini del prossimo ROI da raggiungere ed i passi dettagliati da seguire;

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 che imposta una stringa formattata in HTML come testo della `TextView` dedicata a mostrare l'istruzione dettagliata

Argomenti:

- `detailedInst : String`
Stringa contenente l'istruzione dettagliata formattata come HTML

- + `setPhoto(photo : PhotoInformation) : void`

Override Metodo utilizzato per visualizzare la lista delle anteprime delle foto relative ad un certo POI

Argomenti:

- `photo : PhotoInformation`
Oggetto che contiene la lista degli URI delle immagini relative ad un certo POI

4.4.4.8 view::DeveloperUnlockerView

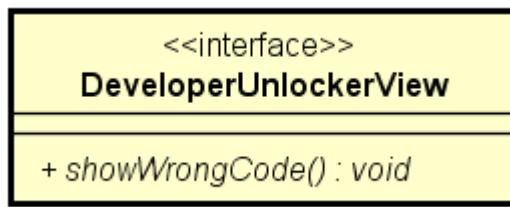


Figura 182: Interfaccia DeveloperUnlockerView

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

4.4.4.9 view::DeveloperUnlockerViewImp

DeveloperUnlockerViewImp	
- developerUnlockerActivity : DeveloperUnlockerActivity	
- developerCode : EditText	
- button : Button	
+ DeveloperUnlockerViewImp(presenter : DeveloperUnlockerActivity)	
+ showWrongCode() : void	

Figura 183: 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:

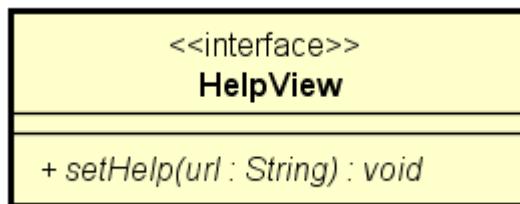
- - button : Button
Bottone per confermare l'inserimento del codice sviluppatore
- - developerCode : EditText
EditText in cui è possibile inserire il codice sviluppatore
- - developerUnlockerActivity : DeveloperUnlockerActivity
Presenter della View

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

4.4.4.10 view::HelpView**Figura 184:** 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

4.4.4.11 view::HelpViewImp

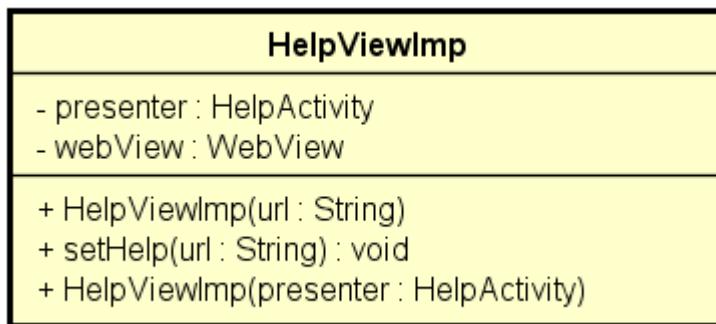


Figura 185: Classe HelpViewImp

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

4.4.4.12 view::HomeView

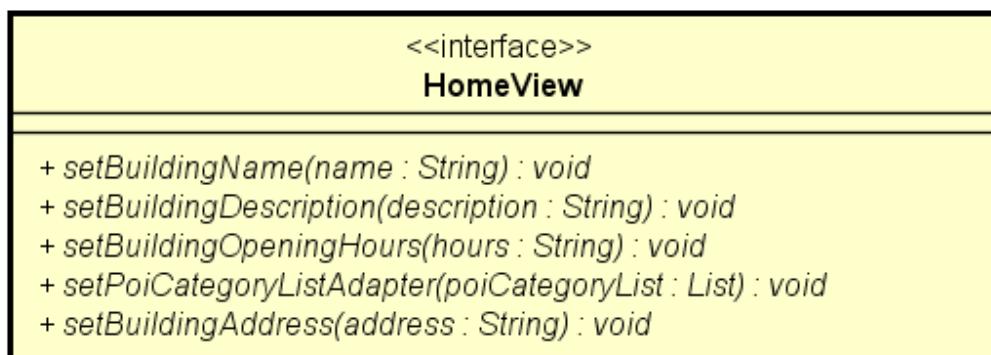


Figura 186: Interfaccia HomeView

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*
Imposta l'indirizzo dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- address : String
Indirizzo dell'edificio

- + *setBuildingDescription(description : String) : void*
Imposta la descrizione dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- description : String
Descrizione dell'edificio

- + *setBuildingName(name : String) : void*
Imposta il nome dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- name : String
Nome dell'edificio

- + *setBuildingOpeningHours(hours : String) : void*
Imposta la descrizione dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- hours : String
Orari di apertura

- + *setPoiCategoryListAdapter(poiCategoryList : List<String> : void)*
Imposta la lista delle categorie di POI dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- poiCategoryList : List<String>
Lista di categorie di POI dell'edificio

4.4.4.13 view::HomeViewImp

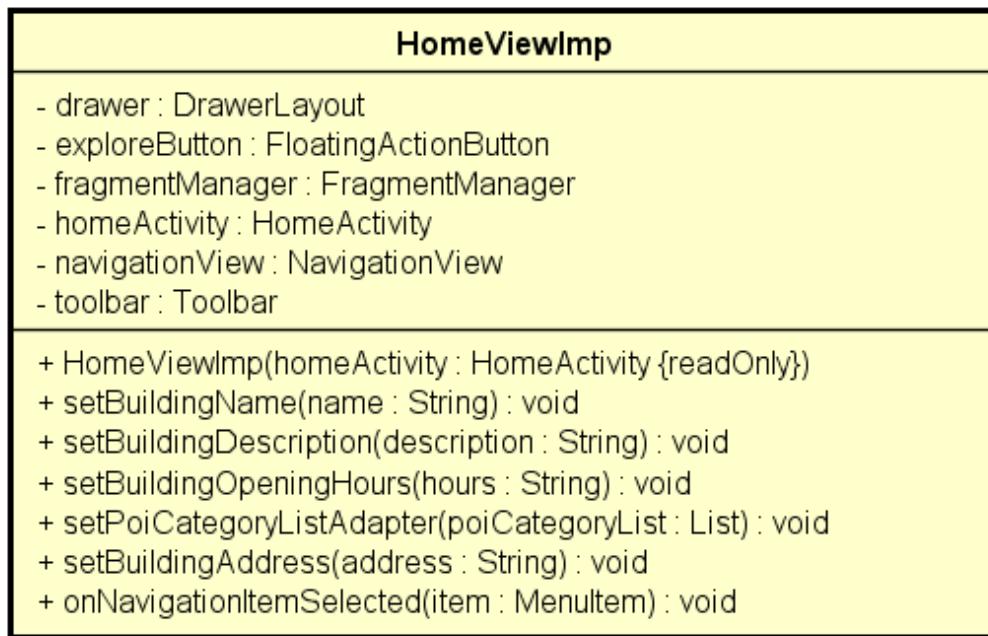


Figura 187: Classe HomeViewImp

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:

- - `drawer` : `DrawerLayout`
Permette di estrarre delle view dagli angoli dello schermo
- - `exploreButton` : `FloatingActionButton`
Bottone che permette di trovare i POI intorno all'utente
- - `fragmentManager` : `FragmentManager`
Riferimento al gestore dei Fragment
- - `homeActivity` : `HomeActivity`
Presenter della View
- - `navigationView` : `NavigationView`
View che gestisce la navigazione
- - `toolbar` : `Toolbar`
Barra degli strumenti che permette la visualizzazione del menu

Metodi:

- + `HomeViewImp(homeActivity : HomeActivity, fragmentManager : FragmentManager)`
Costruttore della classe `HomeViewImp`

Argomenti:

- `homeActivity` : `HomeActivity`
Presenter della View che viene creata
- `fragmentManager` : `FragmentManager`
Gestore dei fragment dell'oggetto

- + `onNavigationItemSelected(item : MenuItem) : boolean`
Gestisce i tap dell'utente nel Drawer: esegue l'azione appropriata rispetto alla voce di menù scelta dall'utente

Argomenti:

- `item` : `MenuItem`
Voce del menù scelta dall'utente

- + `setBuildingAddress(address : String) : void`
Override Imposta l'indirizzo dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- `address` : `String`
Indirizzo dell'edificio

- + `setBuildingDescription(description : String) : void`
Override Imposta la descrizione dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- `description : String`
Descrizione dell'edificio

- + `setBuildingName(name : String) : void`
Override Imposta il nome dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- `name : String`
Nome dell'edificio

- + `setBuildingOpeningHours(hours : String) : void`
Override Imposta gli orari di apertura dell'edificio all'interno del widget progettato per mostrarli

Argomenti:

- `hours : String`
Orari di apertura dell'edificio

- + `setPoiCategoryListAdapter(poiCategoryList : List<String>) : void`
Override Imposta la lista delle categorie di POI dell'edificio all'interno del widget progettato per mostrarlo

Argomenti:

- `poiCategoryList : List<String>`
Lista di categorie di POI dell'edificio

4.4.4.14 view::ImageDetailView



Figura 188: Interfaccia ImageDetailView

Nome: `ImageDetailView`;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per effettuare il binding della View con la lista di immagini relative ad una certa istruzione;

Descrizione: ImageDetailView espone i metodi utili al binding della View con la lista di immagini relative ad una certa istruzione;

Metodi:

- + *setAdapter(listLength : int) : void*

Collega l'Adapter appropriato alla View deputata a mostrare le immagini relative ad una certa istruzione di navigazione

Argomenti:

- *listLength : int*

Numero delle immagini da mostrare

4.4.4.15 view::ImageDetailViewImp

ImageDetailViewImp	
- presenter : ImageDetailActivity	
- pager : ViewPager	
+ setAdapter(listLength : int) : void	
+ ImageDetailViewImp(presenter : ImageDetailActivity) : void	

Figura 189: Classe ImageDetailViewImp

Nome: ImageDetailViewImp;

Tipo: Classe;

Implementa:

- ImageDetailView.

Visibilità: public;

Utilizzo: Viene utilizzata per eseguire il binding della View con la lista di immagini relative ad una certa istruzione;

Descrizione: La classe ImageDetailViewImp espone i metodi utili al binding della View con la lista di immagini relative ad una certa istruzione;

Attributi:

- - pager : ViewPager
Riferimento al widget responsabile dello slideshow
- - presenter : ImageDetailActivity
Presenter della View

Metodi:

- + ImageDetailViewImp(presenter : ImageDetailActivity) : void

Costruttore della classe ImageDetailViewImp

Argomenti:

- presenter : ImageDetailActivity
Presenter della View che viene creata

- + setAdapter(listLength : int) : void

Override Collega l'Adapter appropriato alla View deputata a mostrare le immagini relative ad una certa istruzione di navigazione

Argomenti:

- listLength : int
Numero delle immagini da mostrare

4.4.4.16 view::LocalMapManagerView

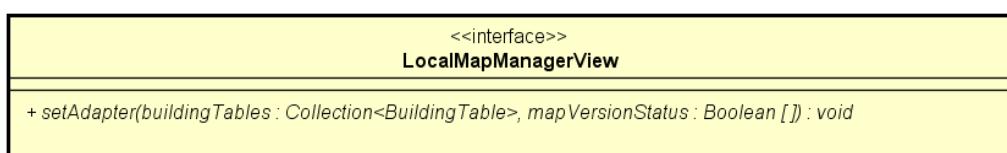


Figura 190: Interfaccia LocalMapManagerView

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:

- + *setAdapter(mapsVersionStatus : boolean [] , buildingTables : Collection<BuildingTable>) : void*

Metodo utilizzato per visualizzare la lista delle mappe salvate nel database locale

Argomenti:

- *mapsVersionStatus : boolean []*
Array contenente lo stato di ogni mappa installata. Se vero allora la mappa è da aggiornare, se falso non lo è
- *buildingTables : Collection<BuildingTable>*
Collegamento tra la lista delle mappe salvate nel database locale e la view in cui esse devono essere mostrate

4.4.4.17 view::LocalMapManagerViewImp

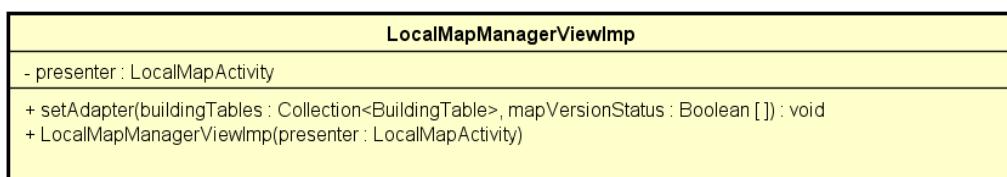


Figura 191: Classe LocalMapManagerViewImp

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

- + setAdapter(mapsVersionStatus : boolean [] , buildingTables : Collection<BuildingTable>) : void

Override Metodo utilizzato per visualizzare la lista delle mappe salvate nel database locale

Argomenti:

- mapsVersionStatus : boolean []
Array contenente lo stato di ogni mappa installata. Se vero allora la mappa è da aggiornare, se falso non lo è
- buildingTables : Collection<BuildingTable>
Collegamento tra la lista delle mappe salvate nel database locale e la view in cui esse devono essere mostrate

4.4.4.18 view::LoggingView

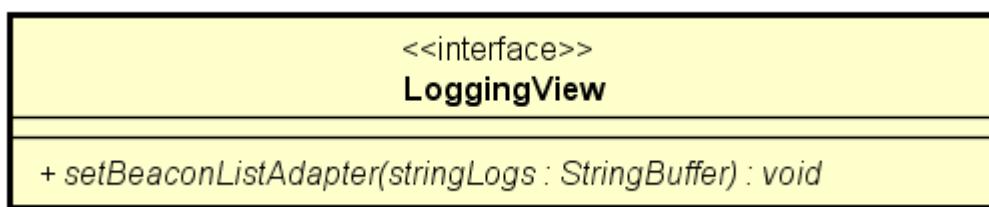


Figura 192: Interfaccia LoggingView

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(stringLogs : StringBuffer) : void*
Metodo utilizzato per visualizzare la lista dei beacon rilevati

Argomenti:

- *stringLogs : StringBuffer*
Collegamento tra la lista dei beacon rilevati e la view in cui essi devono essere mostrati

4.4.4.19 view::LoggingViewImp

LoggingViewImp	
- presenter : LoggingActivity	
- listLog : ListView	
- btnStopLog : FloatingActionButton	
+ LoggingViewImp(presenter : LoggingActivity)	
+ setBeaconListAdapter(stringLogs : StringBuffer) : void	

Figura 193: 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(stringLogs : StringBUffer) : void
Override Metodo utilizzato per visualizzare la lista dei beacon rilevati

Argomenti:

- stringLogs : StringBUffer
Collegamento tra la lista dei beacon rilevati e la view in cui essi devono essere mostrati

4.4.4.20 view::LogInformationView

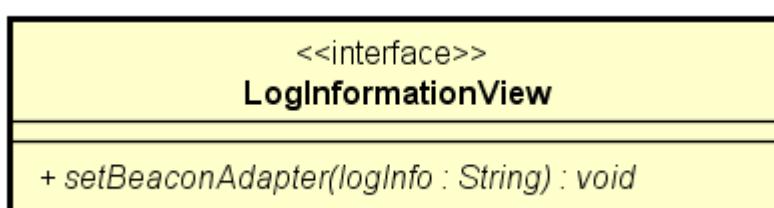


Figura 194: Interfaccia LogInformationView

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

4.4.4.21 view::LogInformationViewImp

LogInformationViewImp	
- presenter : LogInformationActivity	
- txtLog : TextView	
- btnDeleteLog : FloatingActinButton	
+ LogInformationViewImp(presenter : LogInformationActivity)	
+ setBeaconAdapter(logInfo : String) : void	
- showAlertDialog() : void	

Figura 195: 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 informazione dei log e la view in cui esse devono essere mostrate

- - showAlertDialog() : void

Metodo utilizzato per mostrare un dialog di conferma per poter cancellare un log

4.4.4.22 view::MainDeveloperView



Figura 196: 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(stringLogs : String []) : void**
Metodo utilizzato per visualizzare la lista di tutti i log salvati in locale

Argomenti:

- **stringLogs : String []**
Collegamento tra la lista dei log e la view in cui essi devono essere mostrati

4.4.4.23 view::MainDeveloperViewImp

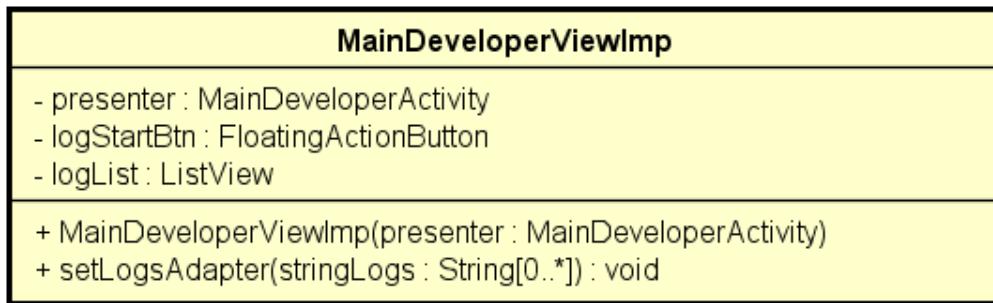


Figura 197: Classe MainDeveloperViewImp

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(stringLogs : String []) : void
Override Metodo utilizzato per visualizzare la lista di tutti i log salvati in locale

Argomenti:

- stringLogs : String []
Collegamento tra la lista dei log e la view in cui essi devono essere mostrati

4.4.4.24 view::NavigationView

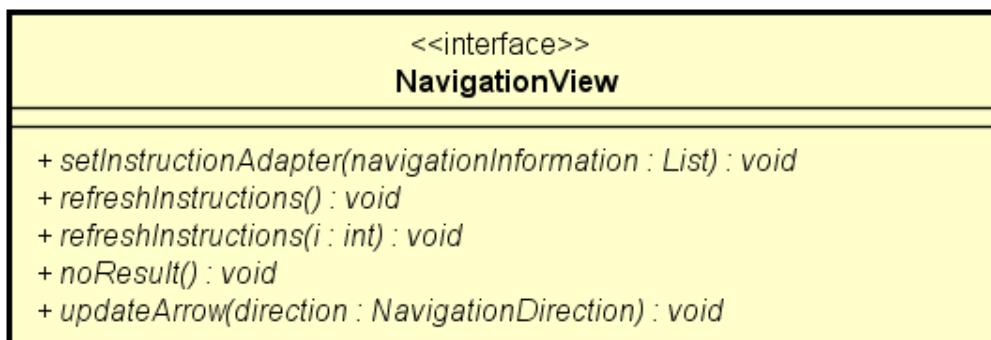


Figura 198: Interfaccia NavigationView

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:

- + **noResult()** : void
Metodo che viene invocato nel caso in cui non siano presenti risultati
- + **refreshInstructions(i : int)** : void
Ricarica le istruzioni

Argomenti:

- i : int

Intero che rappresenta l'istruzione da aggiornare

- + **refreshInstructions()** : void
Metodo utilizzato per aggiornare la lista di istruzioni
- + **setInstructionAdapter(navigationInformation : List<ProcessedInformation> : void)**
Collega l'Adapter appropriato alla View deputata a mostrare a lista di istruzioni di navigazione utili per raggiungere una certa destinazione

Argomenti:

- navigationInformation : List<ProcessedInformation>
Lista delle istruzioni di navigazione

- + **updateArrow(direction : NavigationDirection) : void**
Metodo che permette di aggiornare la direzione della freccia per la navigazione

Argomenti:

- direction : NavigationDirection
Direzione in cui la freccia deve puntare

4.4.4.25 view::NavigationViewImp

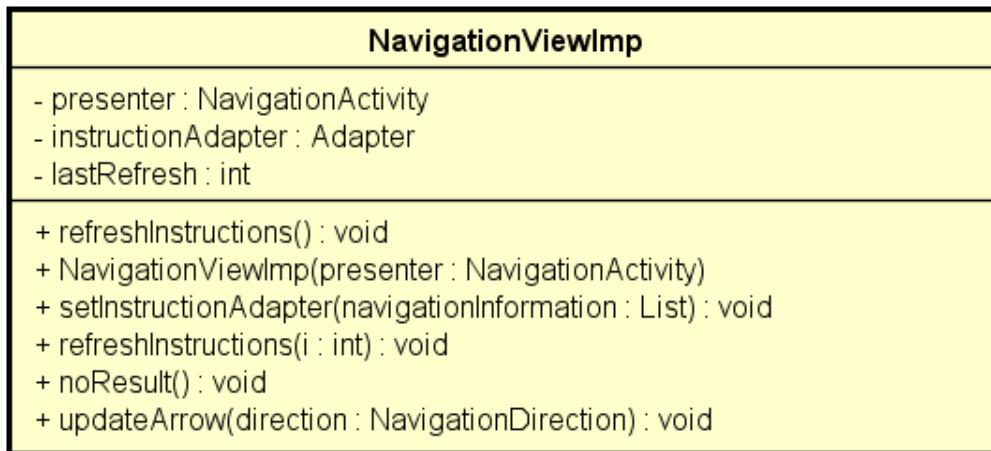


Figura 199: Classe NavigationViewImp

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
Riferimento al widget responsabile di mostrare la lista di istruzioni
- - presenter : NavigationActivity
Presenter della View

Metodi:

- + `NavigationViewImp(presenter : NavigationActivity)`

Costruttore della classe NavigationViewImp

Argomenti:

- `presenter : NavigationActivity`
Presenter della View che viene creata

- + `noResult() : void`

Override Metodo che viene invocato nel caso in cui non siano presenti risultati

- + `refreshInstructions(i : int) : void`

Override Ricarica le istruzioni

Argomenti:

- `i : int`
Intero che rappresenta l'istruzione da aggiornare

- + `refreshInstructions() : void`

Override Metodo utilizzato per aggiornare la lista di istruzioni

- + `setInstructionAdapter(navigationInformation : List<ProcessedInformation> : void)`

Override Collega l'Adapter appropriato alla View deputata a mostrare a lista di istruzioni di navigazione utili per raggiungere una certa destinazione

Argomenti:

- `navigationInformation : List<ProcessedInformation>`
Lista delle istruzioni di navigazione

- + `updateArrow(direction : NavigationDirection) : void`

Override Metodo che permette di aggiornare la direzione della freccia per la navigazione

Argomenti:

- `direction : NavigationDirection`
Direzione in cui la freccia deve puntare

4.4.4.26 view::NearbyPoiView

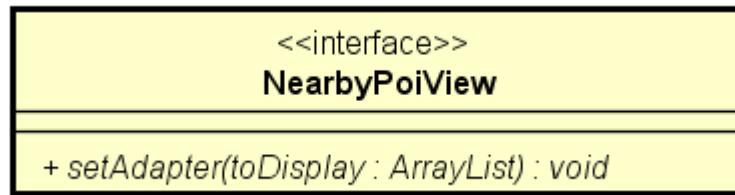


Figura 200: Interfaccia NearbyPoiView

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(toDisplay : ArrayList<String>) : void`
Metodo utilizzato per visualizzare tutti i POI nelle circostanze dell'utente

Argomenti:

- `toDisplay : ArrayList<String>`
Array contenente le stringhe che rappresentano i nomi dei PointOfInterest circostanti l'utente

4.4.4.27 view::NearbyPoiViewImp

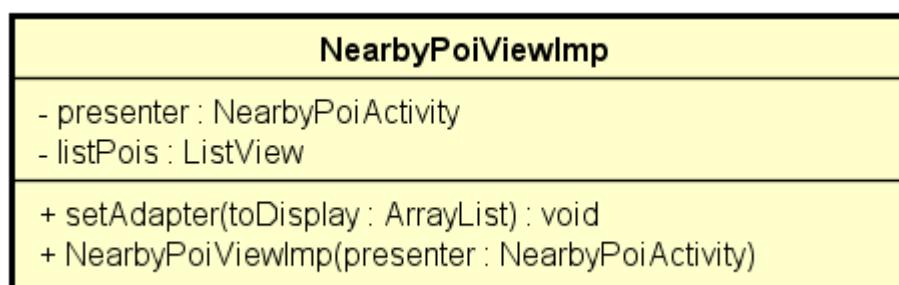


Figura 201: Classe NearbyPoiViewImp

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(toDisplay : ArrayList<String>) : void`
Override Metodo utilizzato per visualizzare tutti i POI nelle circostanze dell'utente

Argomenti:

- `toDisplay` : `ArrayList<String>`
Array contenente le stringhe che rappresentano i nomi dei PointOfInterest circostanti l'utente

4.4.4.28 view::PoiCategoryView



Figura 202: 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(toDisplay : List<String>) : void*
Metodo utilizzato per visualizzare tutti i POI appartenenti ad una certa categoria

Argomenti:

- *toDisplay : List<String>*
Array di stringhe che rappresentano le categorie che devono essere mostrate

4.4.4.29 view::PoiCategoryViewImp

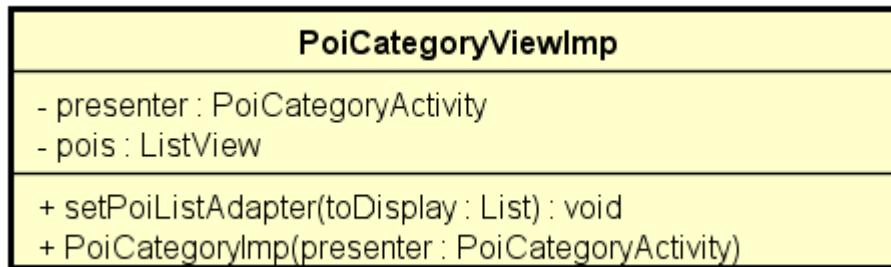


Figura 203: Classe PoiCategoryViewImp

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(toDisplay : List<String>) : void`
Override Metodo utilizzato per visualizzare tutti i POI appartenenti ad una certa categoria

Argomenti:

- `toDisplay : List<String>`
Array di stringhe che rappresentano le categorie che devono essere mostrate

4.4.4.30 view::PoiView

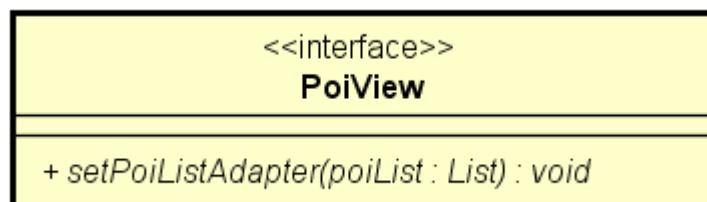


Figura 204: Interfaccia PoiView

Nome: PoiView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Utilizzata per visualizzare i POI di un edificio;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente la lista dei POI appartenenti ad una data categoria;

Metodi:

- + `setPoiListAdapter(poiList : List<String>) : void`
Metodo utilizzato per visualizzare tutti i POI

Argomenti:

- `poiList : List<String>`
lista di stringhe che rappresentano tutti i POI

4.4.4.31 view::PoiViewImp

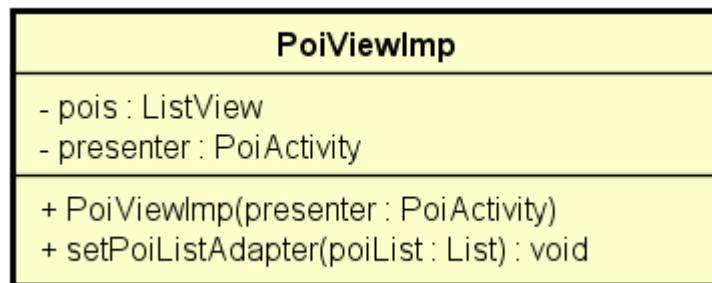


Figura 205: Classe PoiViewImp

Nome: PoiViewImp;

Tipo: Classe;

Implementa:

- PoiView.

Visibilità: public;

Utilizzo: Utilizzata per visualizzare le informazioni associate ad un certo POI;

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` : PoiActivity
Presenter della View

Metodi:

- + `PoiViewImp(presenter : PoiActivity)`
Costruttore della classe PoiViewImp

Argomenti:

- **presenter** : PoiActivity
Presenter della View che viene creata
- + **setPoiListAdapter(poiList : List<String>)**
Override Metodo utilizzato per visualizzare tutti i POI
 - Argomenti:**
 - **poiList** : List<String>
Lista di stringhe che rappresentano tutti i POI

4.4.4.32 view::PreferencesView

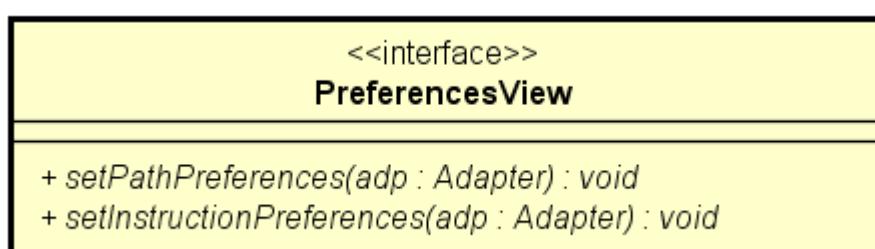


Figura 206: Interfaccia PreferencesView

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:

- + **setPreferences(xmlPreferenceResource : int) : void**
Metodo utilizzato per visualizzare le preferenze dell'applicazione

Argomenti:

- **xmlPreferenceResource : int**
Risorsa XML che contiene tutte le preferenze impostabili

4.4.4.33 view::PreferencesViewImp

PreferencesViewImp
- presenter : PreferencesActivity - preferenceFragment : PreferenceFragment
+ PreferencesViewImp(presenter : PreferencesActivity) + setPathPreferences(adp : Adapter) : void + setInstructionPreferences(adp : Adapter) : void + setPreferences(xmlPreferenceResource : int) : void + onSharedPreferenceChanged(sharedPreferences : SharedPreferences, key : String) : void

Figura 207: 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:

- - preferenceFragment : PreferenceFragment
Fragment per mostrare le preferenze
- - presenter : PreferencesActivity
Presenter della View

Metodi:

- + PreferencesViewImp(presenter : PreferencesActivity)
Costruttore della classe PreferencesViewImp

Argomenti:

- **presenter** : PreferencesActivity
Presenter della View che viene creata
- + onSharedPreferenceChanged(sharedPreferences : SharedPreferences, key : String) : void
Override Metodo che viene invocato ad ogni cambio di preferenza. Tale cambio viene propagato al presenter che ha il compito di salvarlo.

Argomenti:

- **sharedPreferences** : SharedPreferences
Mappa chiave-valore dove vengono salvate le preferenze
- **key** : String
Preferenza che viene cambiata
- + setPreferences(xmlPreferenceResource : int)
Override Metodo utilizzato per visualizzare le preferenze dell'applicazione

Argomenti:

- **xmlPreferenceResource** : int
Risorsa XML che contiene tutte le preferenze impostabili

4.4.4.34 view::RemoteMapView

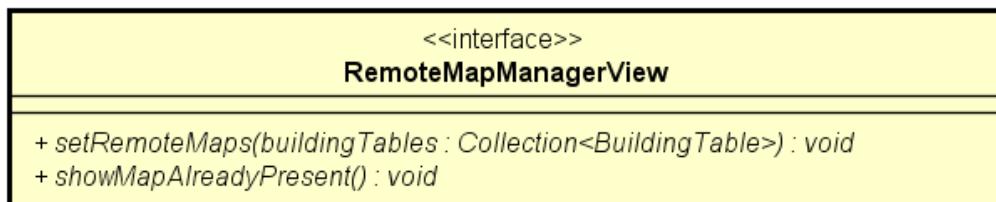


Figura 208: Interfaccia RemoteMapView

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(buildingTables : Collection<BuildingTable>)*
: void

Metodo utilizzate per visualizzare le mappe che è possibile scaricare da un server remoto

Argomenti:

- *buildingTables : Collection<BuildingTable>*
Collegamento tra la lista delle mappe che è possibile scaricare e la view in cui esse devono essere mostrate

- + *showMapAlreadyPresent() : void*

Metodo utilizzato per informare l'utente che la mappa selezionata è già installata sul dispositivo

4.4.4.35 view::RemoteMapViewManagerViewImp

RemoteMapViewManagerViewImp
- presenter : RemoteMapViewManagerActivity
+ setRemoteMaps(buildingTables : Collection<BuildingTable>) : void
+ RemoteMapViewManagerViewImp(presenter : RemoteMapViewManagerActivity)
+ showMapAlreadyPresent() : void

Figura 209: Classe RemoteMapViewManagerViewImp

Nome: RemoteMapViewManagerViewImp;

Tipo: Classe;

Implementa:

- RemoteMapViewManagerView.

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:

- + `RemoteMapManagerViewImp(presenter : RemoteMapManagerActivity)`
Costruttore della classe RemoteMapManagerViewImp

Argomenti:

- `presenter : RemoteMapManagerActivity`
Presenter della View che viene creata

- + `setRemoteMaps(buildingTables : Collection<BuildingTable>)`
: void

Override Metodo utilizzate per visualizzare le mappe che è possibile scaricare da un server remoto

Argomenti:

- `buildingTables : Collection<BuildingTable>`
Collegamento tra la lista delle mappe che è possibile scaricare e la view in cui esse devono essere mostrate

- + `showMapAlreadyPresent() : void`

Override Metodo utilizzato per infomrare l'utente che la mappa selezionata è già installata sul dispositivo

5 Schema base di dati

Di seguito viene presentata lo schema ER_g 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 rappresentante l'edificio di interesse. Si fa notare che la base di dati non memorizza separatamente gli elementi che compongono i grafici.

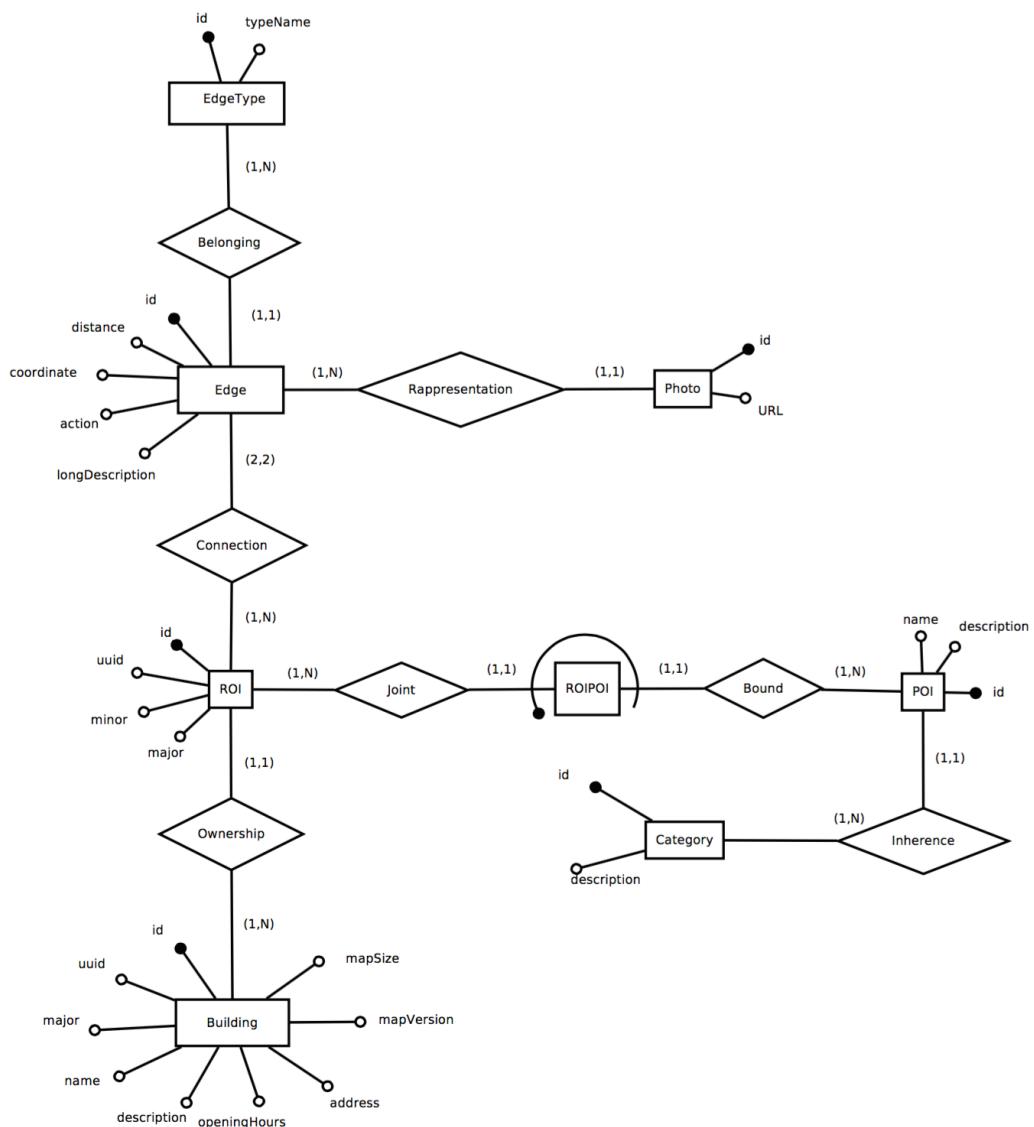


Figura 210: Schema ER - base di dati

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 211 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.

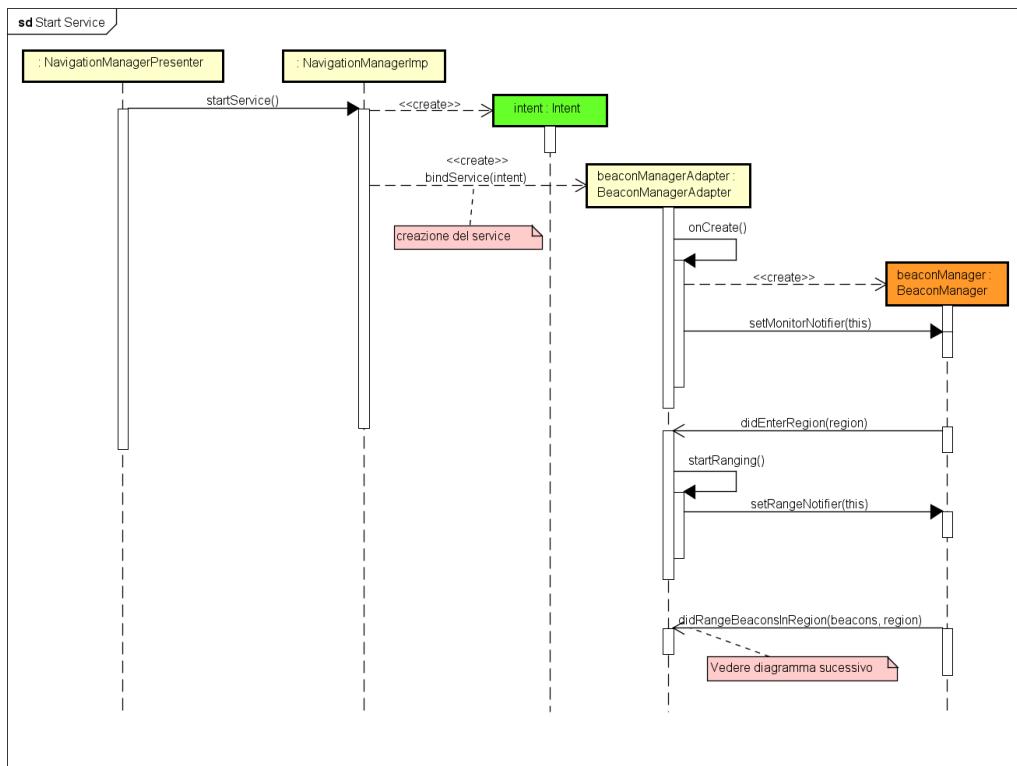


Figura 211: Diagramma di sequenza - Avvio di un service_g per il rilevamento beacon

6.2 Elaborazione beacon rilevati e comunicazione broadcast

Il diagramma in figura 212 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.

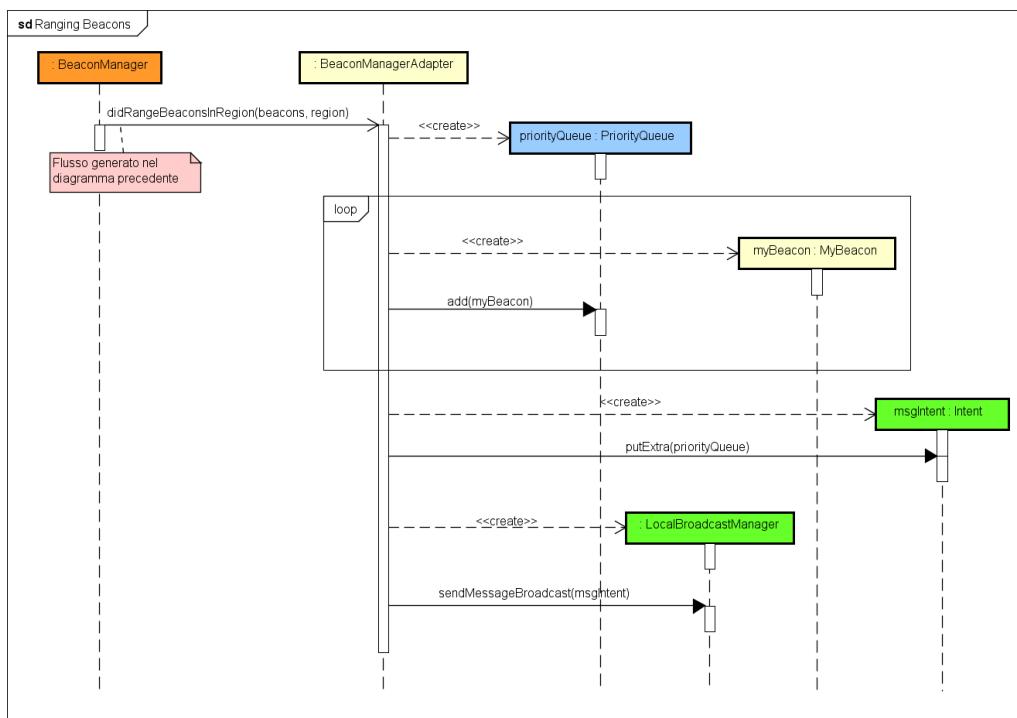


Figura 212: Diagramma di sequenza - Elaborazione beacon rilevati e comunicazione broadcast

6.3 Avvio navigazione

Il diagramma in figura 213 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_g) 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 scomincerà e le restituirà alla view e quindi all'utente.

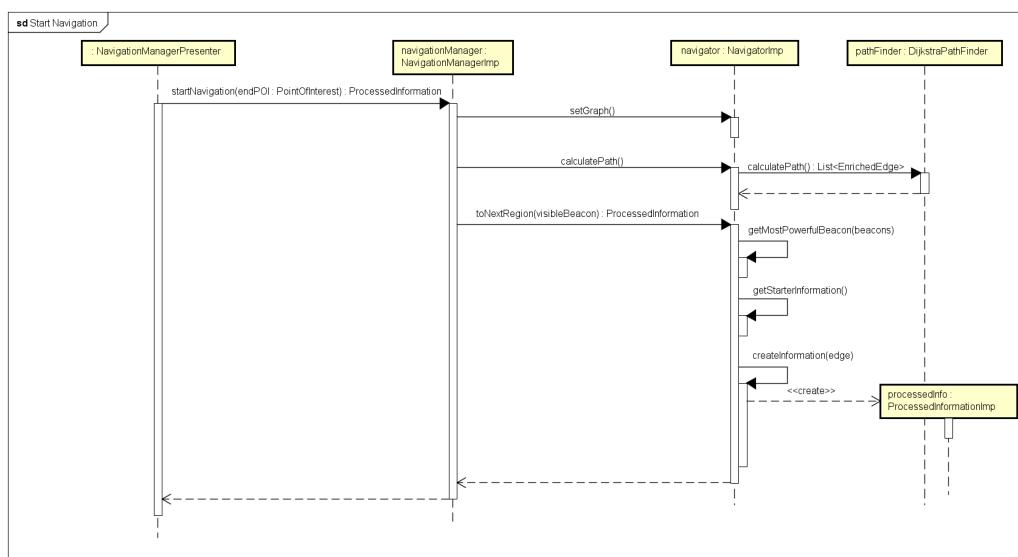


Figura 213: Diagramma di sequenza - Avvio navigazione

6.4 Avvio della bussola

Il diagramma in figura 214 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.

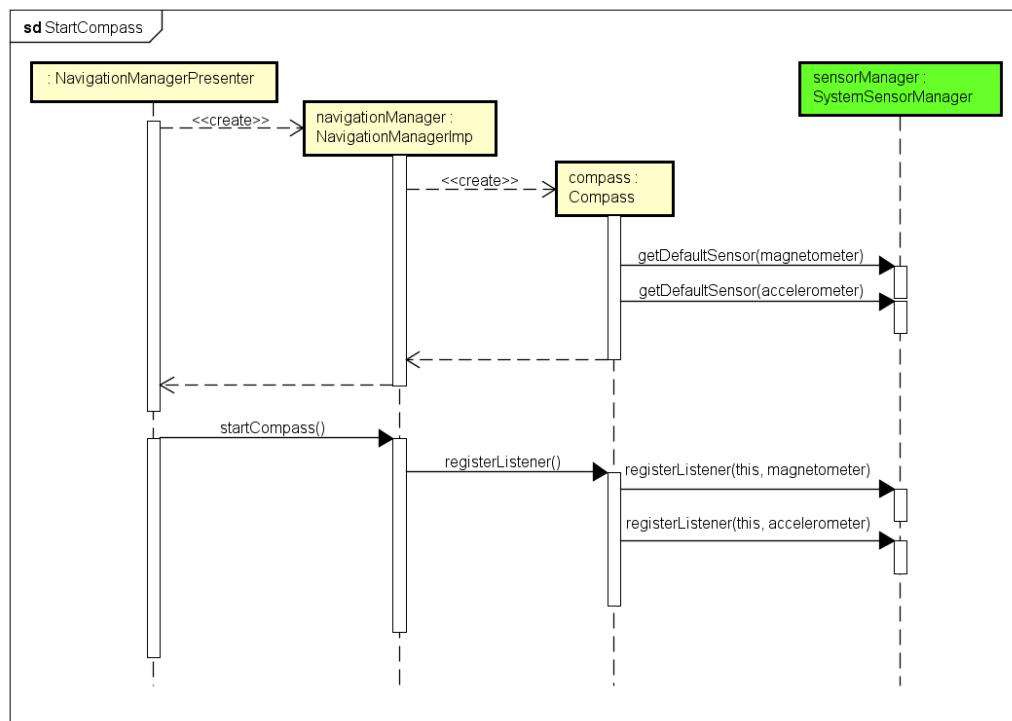


Figura 214: Diagramma di sequenza - Avvio della bussola

6.5 Caricamento della mappa dal database

Il diagramma in figura 215 rappresenta il flusso generato dalla chiamata del metodo `onReceive()` la quale è invocata solo quando è avvenuta la ricezione dei beacon nelle vicinanze attraverso l'uso degli Intent offerti dalla libreria SDK Android. Quando la classe `InformationManagerImp` riceve i beacon, chiama il metodo privato `loadMap()` preleva il primo beacon, ossia quello rilevato con la potenza di segnale più alta, dalla `PriorityQueue` attraverso la chiamata al metodo `peek()`. Una volta estratto il beacon, si preleva il Major, da esso e si chiama il metodo `isBuildingMapPresent()` della classe `BuidingService` per capire se la mappa è disponibile, se questa lo è viene invocato il metodo `findBuildingMapByMajor()` che restituisce la mappa che viene settata in un campo dati della classe `InformationManagerImp`.

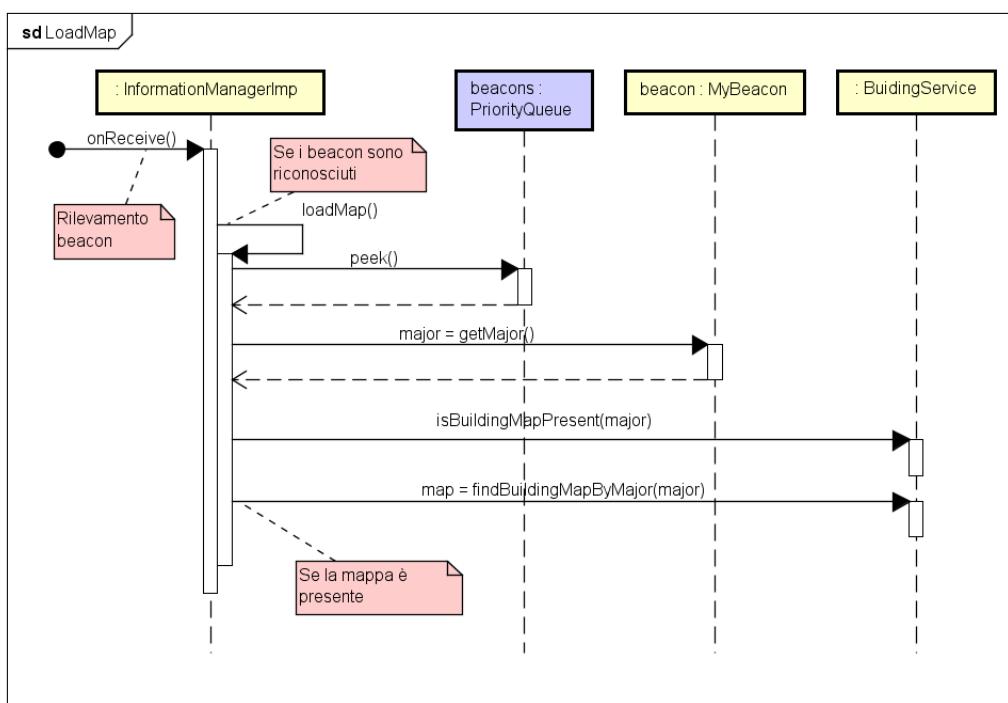


Figura 215: Diagramma di sequenza - Caricamento della mappa dal database

6.6 Costruzione di oggetti-Table da Json

Il diagramma in figura 216 rappresenta il flusso generato dall'oggetto della classe `RemoteEdgeDao` per la costruzione di un oggetto `EdgeTable` a partire da un oggetto `JsonObject`:

- viene creato un oggetto `Gson`. Tale oggetto permette di gestire l'oggetto `JsonObject` passato;
- su questo oggetto viene invocato il metodo `fromJson()`, passando come parametri l'oggetto `JsonObject` e l'oggetto `EdgeTable.class`. In questo modo viene creato l'oggetto `EdgeTable`, sfruttando il fatto che l'oggetto `JsonObject` è stato creato utilizzando gli stessi nomi per i campi dato di tale oggetto e i campi dato dell'oggetto `EdgeTable`.

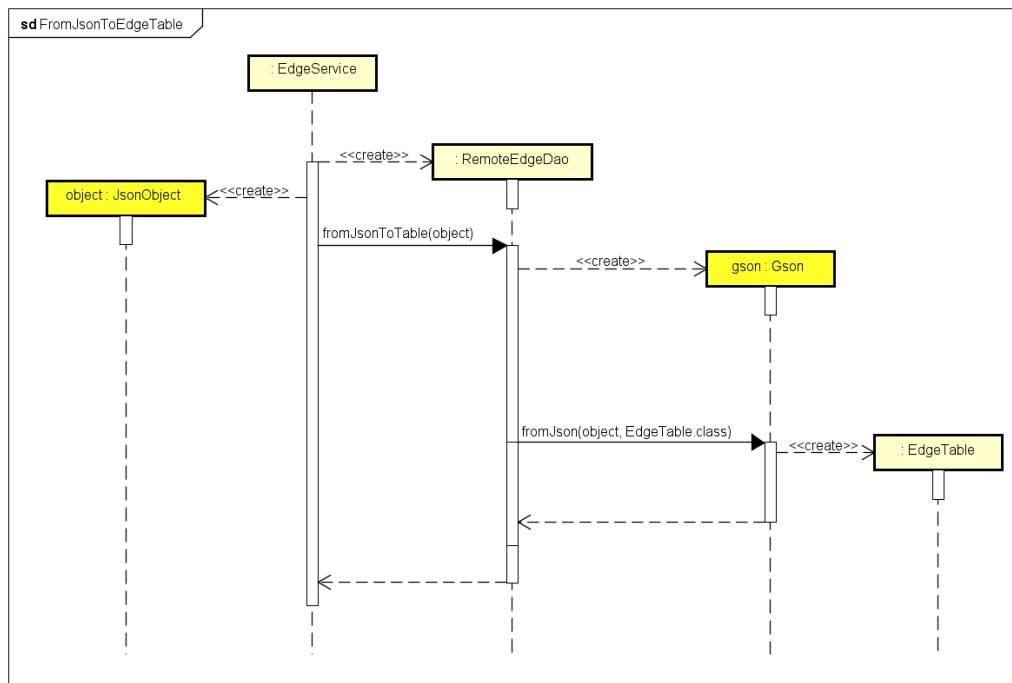


Figura 216: Diagramma di sequenza - Costruzione di `EdgeTable` da `Json`

Tale costruzione è analoga per oggetti `BuildingTable`, `CategoryTable`, `EdgeTypeTable`, `PhotoTable`, `PointOfInterestTable` e `RegionOfInterestTable`, mentre è differente per la classe `RoiPoiTable`. In questo caso infatti non è stato possibile scaricare un file JSON che abbia dei campi dato con lo stesso nome dei campi dato di `RoiPoiTable`.

Il diagramma in figura 217 rappresenta la costruzione di un oggetto `RoiPoiTable` a partire da un oggetto `JsonObject`:

- viene creato un oggetto `GsonBuilder`. Tale oggetto è necessario per la costruzione di un oggetto a partire da un oggetto `JsonObject` che abbia campi dato con nome differente dall'oggetto da costruire;
- viene creato un oggetto di una classe anonima che implementa la classe `JsonDeserializer<RoiPoiTable>`. In tale classe viene fatto l'override del metodo `deserialize()` definendo come un oggetto `RoiPoiTable` può essere costruito sfruttando i campi dati dell'oggetto `JsonObject`;
- viene invocato sull'oggetto `GsonBuilder registerTypeAdapter()` passando come parametri gli oggetti `RoiPoiTable.class` e l'oggetto della classe che implementa `JsonDeserializer<RoiPoiTable>`;
- viene creato un oggetto `Gson` invocando il metodo `create()` sull'oggetto `GsonBuilder`;
- infine, per creare l'oggetto `RoiPoiTable`, viene invocato il metodo `fromJson()` sull'oggetto `Gson` passando come argomenti i due oggetti `JsonObject` e `RoiPoiTable.class`.

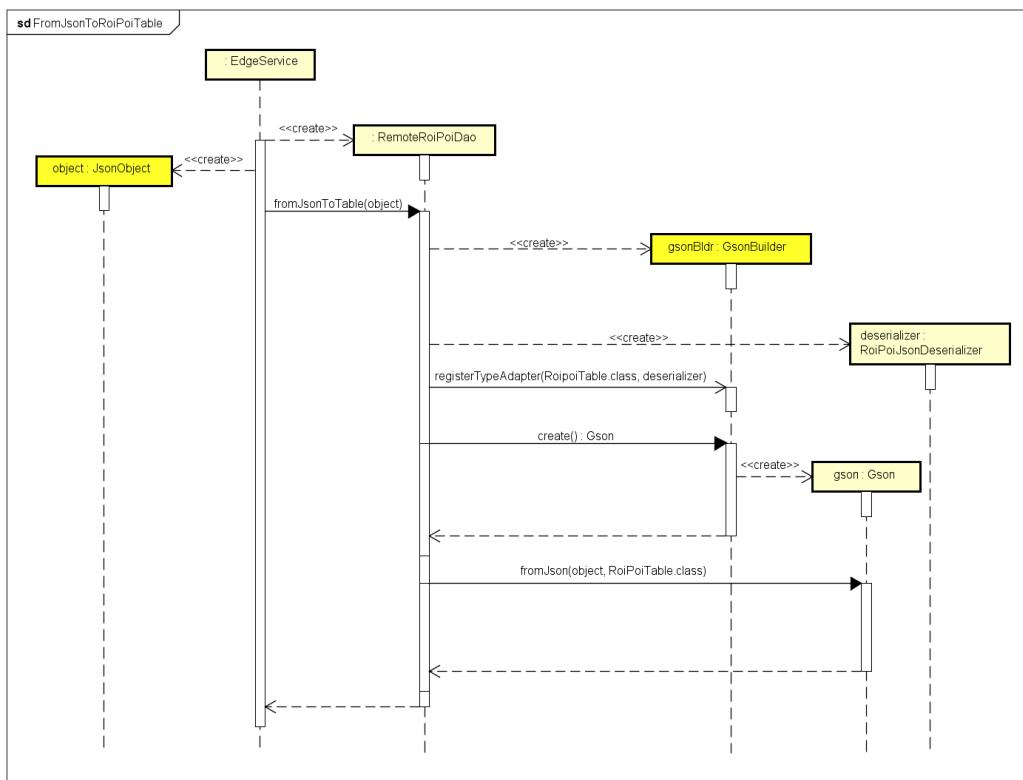


Figura 217: Diagramma di sequenza - Costruzione di RoiPoiTable da Json

6.7 Modifica delle preferenze

Il diagramma in figura 218 rappresenta il flusso generato dall'oggetto della classe `SettingImp` per la modifica delle preferenze sul percorso:

- per prima viene aggiornato il campo dati dell'oggetto `SettingImp`, egualandolo all'oggetto `PathPreference` passato;
- viene recuperata un'istanza di `SharedPreferences.Editor` per poter aggiornare le preferenze salvate nelle `SharedPreferences`;
- infine vengono aggiornate le `SharedPreferences` prima invocando il metodo `putInt()` sull'oggetto `SharedPreferences.Editor` passando come argomenti la costante di classe `PATH_PREFERENCES` e l'intero che rappresenta l'oggetto `PathPreference` passato alla chiamata del metodo `setPathPreference()`.

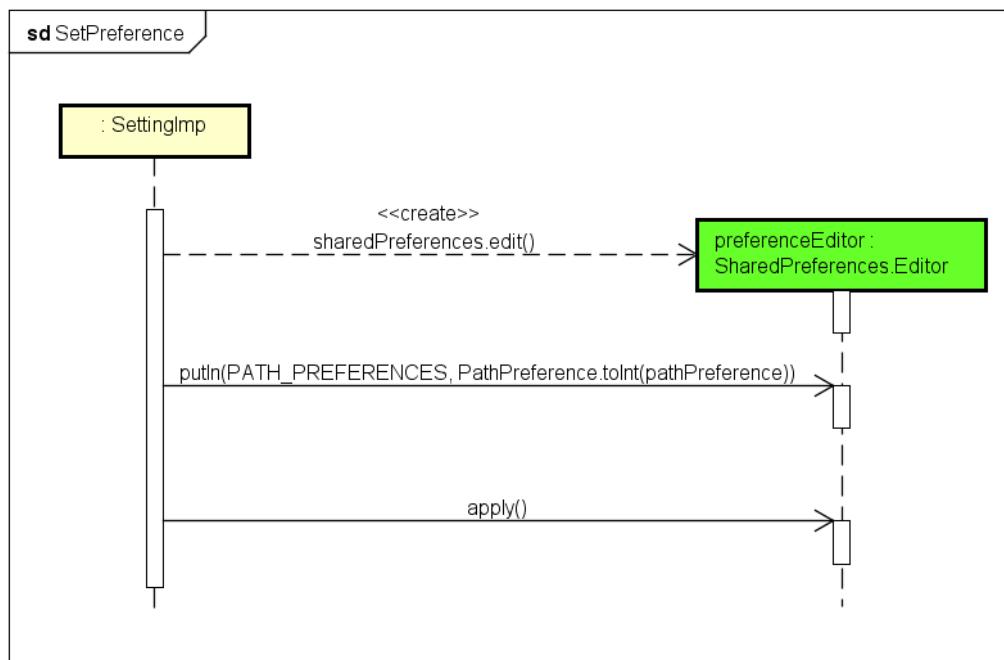


Figura 218: Diagramma di sequenza - Modifica delle preferenze

7 Tracciamento

7.1 Tracciamento Classi-Requisiti

Classe	Requisiti
model::InformationManagerImp	RDesF13.3.5 RObbF10 RObbF13.3 RObbF3 RObbF3.1 RObbF3.2 RObbF8.1 ROpzF10.2.2
model::MessageSendType	RDesF13.3.5 RDesF8.4.2.3 RObbF10 RObbF13.3 RObbF8 RObbF8.1 RObbF8.4 RObbF8.4.2 RObbF8.5 ROpzF10.2.2
model::NavigationManagerImp	RDesF8.4.2.3 RObbF8 RObbF8.4 RObbF8.4.2 RObbF8.5 ROpzF8.4.2.6
model::NoBeaconSeenException	ROpzF8.4.2.6

Classe	Requisiti
model::ServiceConnectionImp	RDesF13.3.5 RDesF8.4.2.3 RObbF10 RObbF13.3 RObbF8 RObbF8.1 RObbF8.4 RObbF8.4.2 RObbF8.5 ROpzF10.2.2
model::beacon::BeaconManagerAdapter	RDesF8.4.2.3 RObbF10.10 RObbF10.3 RObbF10.4 RObbF10.5 RObbF10.6 RObbF10.9 RObbF8 RObbF9 RObbF9.1 RObbF9.1.1 RObbF9.2 RObbF9.3 RObbF9.4 RObbF9.5 RObbF9.6 ROpzF10.2.2 ROpzF8.4.2.5 ROpzF8.4.2.6

Classe	Requisiti
model::beacon::LocalBinder	RDesF13.3.5 RDesF8.4.2.3 RObbF10 RObbF13.3 RObbF8 RObbF8.1 RObbF8.4 RObbF8.4.2 RObbF8.5 ROpzF10.2.2
model::beacon::LoggerImp	RDesF13.3.1 RDesF13.3.2 RDesF13.3.3 RDesF13.3.5 RObbF13.3
model::beacon::MyBeaconImp	RDesF8.4.2.3 RObbF9 RObbF9.1.1 RObbF9.1.2 RObbF9.2 RObbF9.3 RObbF9.4 RObbF9.5 RObbF9.6
model::beacon::MyDistanceCalculator	RObbF9.4
model::beacon::PeriodType	
model::compass::Compass	RObbF8.4.2
model::dataaccess::dao::BuildingContract	RDesF11.2.1
model::dataaccess::dao::BuildingTable	RDesF11.2 RObbF8.7
model::dataaccess::dao::CategoryContract	RDesF11.2.1
model::dataaccess::dao::CategoryTable	RDesF11.2

Classe	Requisiti
model::dataaccess::dao::DaoFactoryHelper	RDesF11.2 RDesF11.2.1 RDesF11.2.2
model::dataaccess::dao::EdgeContract	RDesF11.2.1
model::dataaccess::dao::EdgeTable	RDesF11.2
model::dataaccess::dao::EdgeTypeContract	RDesF11.2.1
model::dataaccess::dao::EdgeTypeTable	RDesF11.2
model::dataaccess::dao::MapsDbHelper	RDesF11.2.1
model::dataaccess::dao::PhotoContract	RDesF11.2.1
model::dataaccess::dao::PhotoTable	RDesF11.2
model::dataaccess::dao::PointOfInterestContract	RDesF11.2.1
model::dataaccess::dao::PointOfInterestTable	RDesF11.2
model::dataaccess::dao::RegionOfInterestContract	RDesF11.2.1
model::dataaccess::dao::RegionOfInterestTable	RDesF11.2
model::dataaccess::dao::RemoteBuildingDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.1 RDesF11.2.2.2 RDesF11.2.2.3 RDesF11.2.2.3.1 RDesF11.2.2.3.2 RDesF11.2.2.3.3 RDesF11.2.2.3.4 RDesF11.2.2.3.5
model::dataaccess::dao::RemoteCategoryDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::dao::RemoteDaoFactory	RDesF11.2.2

Classe	Requisiti
model::dataaccess::dao::RemoteEdgeDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::dao::RemoteEdgeTypeDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::dao::RemotePhotoDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::dao::RemotePointOfInterestDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::dao::RemoteRegionOfInterestDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::dao::RemoteRoiPoiDao	RDesF11.2.1.2 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::dao::RoiPoiContract	RDesF11.2.1
model::dataaccess::dao::RoiPoiTable	RDesF11.2
model::dataaccess::dao::SQLDao	RDesF11.2.1
model::dataaccess::dao::SQLiteBuildingDao	RDesF11.2.1 RDesF11.2.1.1 RDesF11.2.1.2 RDesF11.2.1.3 RDesF11.2.1.4 RDesF11.2.1.4.1 RDesF11.2.1.4.2 RDesF11.2.1.4.3 RDesF11.2.1.4.4 RDesF11.2.1.4.5 RDesF11.2.2.2 RObbF8.7 RObbF8.8

Classe	Requisiti
model::dataaccess::dao::SQLiteCategoryDao	RDesF11.2.1 RDesF11.2.1.2 RDesF11.2.1.4 RDesF11.2.2.2
model::dataaccess::dao::SQLiteDaoFactory	RDesF11.2.1
model::dataaccess::dao::SQLiteEdgeDao	RDesF11.2.1 RDesF11.2.1.2 RDesF11.2.1.4 RDesF11.2.2.2
model::dataaccess::dao::SQLiteEdgeTypeDao	RDesF11.2.1 RDesF11.2.1.2 RDesF11.2.1.4 RDesF11.2.2.2
model::dataaccess::dao::SQLitePhotoDao	RDesF11.2.1 RDesF11.2.1.2 RDesF11.2.1.4 RDesF11.2.2.2
model::dataaccess::dao::SQLitePointOfInterestDao	RDesF11.2.1 RDesF11.2.1.2 RDesF11.2.1.4 RDesF11.2.2.2
model::dataaccess::dao::SQLiteRegionOfInterestDao	RDesF11.2.1 RDesF11.2.1.2 RDesF11.2.1.4 RDesF11.2.2.2
model::dataaccess::dao::SQLiteRoiPoiDao	RDesF11.2.1 RDesF11.2.1.2 RDesF11.2.1.4 RDesF11.2.2.2

Classe	Requisiti
model::dataaccess::service::BuildingService	RDesF11.2 RDesF11.2.1 RDesF11.2.1.1 RDesF11.2.1.2 RDesF11.2.1.3 RDesF11.2.1.4 RDesF11.2.1.4.1 RDesF11.2.1.4.2 RDesF11.2.1.4.3 RDesF11.2.1.4.4 RDesF11.2.1.4.5 RDesF11.2.2 RDesF11.2.2.1 RDesF11.2.2.2 RDesF11.2.2.3 RDesF11.2.2.3.1 RDesF11.2.2.3.2 RDesF11.2.2.3.3 RDesF11.2.2.3.4 RDesF11.2.2.3.5 RDesF11.2.2.4 RDesF11.2.3 RObbF10.10 RObbF8.7
model::dataaccess::service::EdgeService	RDesF11.2 RDesF11.2.1 RDesF11.2.1.4 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::service::PhotoService	RDesF11.2 RDesF11.2.1 RDesF11.2.1.4 RDesF11.2.2 RDesF11.2.2.2

Classe	Requisiti
model::dataaccess::service::PointOfInterestService	RDesF11.2 RDesF11.2.1 RDesF11.2.1.4 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::service::RegionOfInterestService	RDesF11.2 RDesF11.2.1 RDesF11.2.1.4 RDesF11.2.2 RDesF11.2.2.2
model::dataaccess::service::ServiceHelper	RDesF11.2 RDesF11.2.1 RDesF11.2.2
model::navigator::BuildingInformation	RObbF10 RObbF10.10 RObbF10.3 RObbF10.4 RObbF10.5 RObbF10.6 RObbF10.9 RObbF8.1 ROpzF10.2
model::navigator::BuildingMapImp	RObbF10 RObbF10.1 RObbF8.1 ROpzF10.2 ROpzF10.2.2

Classe	Requisiti
model::navigator::NavigatorImp	RDesF8.1.1 RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2 RDesF8.3.1.3 RDesF8.4.2.2 RDesF8.4.2.3 RDesF8.4.3.1 RObbF8 RObbF8.1.2 RObbF8.3 RObbF8.4 RObbF8.4.2 RObbF8.4.2.1 RObbF8.5 RObbF9.1.1 ROpzF8.4.2.4 ROpzF8.4.2.5 ROpzF8.4.3.2 ROpzF8.4.3.3
model::navigator::NoGraphSetException	RObbF8
model::navigator::NoNavigationInformationException	RObbF8
model::navigator::PathException	RDesF8.4.2.3
model::navigator::ProcessedInformationImp	RObbF11.1.2.1 RObbF8.4.2 ROpzF8.4.2.4
model::navigator::algorithm::DijkstraPathFinder	RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2 RDesF8.3.1.3 RObbF8.3
model::navigator::graph::MapGraph	RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2 RObbF8.3

Classe	Requisiti
model::navigator::graph::area::PointOfInterestImp	RDesF8.1.1 RObbF10.2.3 RObbF10.2.4 RObbF8.1 RObbF8.1.2 RObbF9.1.1 RObbF9.1.2 ROpzF10.2 ROpzF10.2.1
model::navigator::graph::area:: PointOfInterestInformation	RObbF10.2.3 RObbF10.2.4 ROpzF10.2 ROpzF10.2.1
model::navigator::graph::area::RegionOfInterestImp	RDesF8.3.1 RDesF8.3.1.1 RObbF8.3
model::navigator::graph::edge::DefaultEdge	RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2
model::navigator::graph::edge::ElevatorEdge	RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2
model::navigator::graph::edge::StairEdge	RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2
model::navigator::graph::navigationinformation:: BasicInformation	RObbF11.1.2.1 RObbF8.4.2 RObbF8.4.2.1
model::navigator::graph::navigationinformation:: DetailedInformation	RDesF8.4.3 RObbF11.1.2.1 RObbF8.4.2 ROpzF8.4.3.2
model::navigator::graph::navigationinformation:: NavigationInformationImp	RDesF8.1.1 RObbF8 RObbF8.4.2 RObbF8.4.2.1

Classe	Requisiti
model::navigator::graph::navigationinformation::PhotoInformation	RDesF8.4.3.1 RObbF8.4.2
model::navigator::graph::navigationinformation::PhotoRef	RDesF8.4.3.1
model::navigator::graph::vertex::VertexImp	RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2 RObbF8.3
model::usersetting::DeveloperCodeManager	RObbF11.3
model::usersetting::InstructionPreference	RDesF11.1.2
model::usersetting::PathPreference	RDesF8.3.1 ROpzF11.1
model::usersetting::SettingImp	RDesF11.1.2 RDesF11.1.2.2 RDesF11.1.2.3 RDesF11.1.2.4 RDesF11.1.2.5 RDesF8.3.1 RDesF8.3.1.1 RDesF8.3.1.2 RObbF11.1.1.3 RObbF11.3 ROpzF11 ROpzF11.1 ROpzF11.1.1 ROpzF11.1.1.1 ROpzF11.1.1.2
module:: AppModule	
module:: DatabaseModule	
module:: InfoModule	
module:: NavModule	
module:: SettingModule	

Classe	Requisiti
presenter::BlankHomeFragment	
presenter::CompleteHomeFragment	
presenter::DetailedInformationActivity	RDesF8.4.3.4 ROpzF8.4.3.2
presenter::DeveloperUnlockerActivity	RObbF11.3.3
presenter::HelpActivity	ROpzF12
presenter::HomeActivity	RObbF10.10 RObbF10.9 RObbF8.1.2 RObbF8.1.2.1 RObbF8.1.3 RObbF8.1.4 RObbF8.2 RObbF8.4.1 RObbF8.4.1.1 RObbF8.4.1.2 RObbF8.4.1.3 RObbF8.4.4 RObbF8.8
presenter::ImageAdapter	RDesF8.4.3.1
presenter::ImageDetailActivity	RDesF8.4.3.1
presenter::ImageDetailFragment	
presenter::ImageListFragment	RDesF8.4.3.1
presenter::LocalMapActivity	RDesF11.2.1.4.1 RDesF11.2.1.4.2 RDesF11.2.1.4.3 RDesF11.2.1.4.4 RDesF11.2.1.4.5 RDesF11.2.2.4
presenter::LoggingActivity	RObbF13.6 RObbF13.8 RObbF13.9

Classe	Requisiti
presenter::LogInformationActivity	RDesF13.3.4
presenter::MainActivity	RObbF8.4.1
presenter::MainDeveloperActivity	RObbF13.3
presenter::MainDeveloperPresenter	RObbF13.3
presenter::MapDownloaderActivity	RDesF11.2.3
presenter::MyApplication	
presenter::NavigationActivity	RDesF8.4.2.3 RObbF8.4.1 RObbF8.4.1.1 RObbF8.4.1.2 RObbF8.4.1.3 RObbF8.4.2.1 RObbF8.6 RObbF8.7 ROpzF8.4.2.5 ROpzF8.4.2.6 ROpzF8.4.3.3
presenter::NavigationAdapter	RObbF8.4.2.1
presenter::NearbyPoiActivity	RObbF13.7
presenter::PoiCategoryActivity	RObbF8.1 RObbF8.1.2 RObbF8.2 RObbF8.4.4
presenter::PreferencesActivity	ROpzF11.1.1.1

Classe	Requisiti
presenter::RemoteMapManagerActivity	RDesF11.2.2.1 RDesF11.2.2.3 RDesF11.2.2.3.1 RDesF11.2.2.3.2 RDesF11.2.2.3.3 RDesF11.2.2.3.4 RDesF11.2.2.3.5 RDesF11.2.2.4 RDesF11.2.3
presenter::SearchSuggestionsProvider	RDesF8.1.1 RDesF8.1.1.1
view::DetailedInformationViewImp	RDesF8.4.3.4 ROpzF8.4.3.2
view::DeveloperUnlockerViewImp	RObbF11.3.2 RObbF11.3.3
view::HelpViewImp	ROpzF12
view::HomeViewImp	RDesF8.1.1 RDesF8.1.1.1 RObbF10.10 RObbF8.1 RObbF8.1.2 RObbF8.1.2.1 RObbF8.1.3 RObbF8.1.4 RObbF8.2 RObbF8.4.1 RObbF8.4.1.1 RObbF8.4.1.2 RObbF8.4.1.3 RObbF8.4.4 RObbF8.8
view::ImageDetailViewImp	RDesF8.4.3.1

Classe	Requisiti
view::LocalMapManagerViewImp	RDesF11.2.1.4.1 RDesF11.2.1.4.2 RDesF11.2.1.4.3 RDesF11.2.1.4.4 RDesF11.2.1.4.5 RDesF11.2.2.4
view::LoggingViewImp	RObbF13 RObbF13.1 RObbF13.2 RObbF13.5 RObbF13.6 RObbF13.8 RObbF13.9 ROpzF13.4
view::LogInformationViewImp	RDesF13.3.4
view::MainDeveloperViewImp	RObbF13.3
view::MapDownloaderViewImp	RDesF11.2.3
view::NavigationViewImp	RDesF8.4.2.3 RObbF8.4.1 RObbF8.4.1.1 RObbF8.4.1.2 RObbF8.4.1.3 RObbF8.4.2.1 RObbF8.6 RObbF8.7 ROpzF8.4.2.5 ROpzF8.4.2.6 ROpzF8.4.3.3
view::NearbyPoiViewImp	RObbF13.7
view::PoiCategoryViewImp	RObbF8.1 RObbF8.1.2 RObbF8.2 RObbF8.4.4
view::PreferencesViewImp	ROpzF11.1.1.1

Classe	Requisiti
view::RemoteMapManagerViewImp	RDesF11.2.2.1
	RDesF11.2.2.1.1
	RDesF11.2.2.3
	RDesF11.2.2.3.1
	RDesF11.2.2.3.2
	RDesF11.2.2.3.3
	RDesF11.2.2.3.4
	RDesF11.2.2.3.5
	RDesF11.2.2.4
	RDesF11.2.3

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	
RObbF3	model::InformationManagerImp
RObbF3.1	model::InformationManagerImp
RObbF3.2	model::InformationManagerImp
RObbV4	
RObbV4.1	

Requisito	Classi
ROpzV4.1.1	
RObbV4.1.2	
ROpzV4.1.3	
ROpzV4.1.4	
RObbQ5	
RObbQ5.1	
RObbQ5.1.1	
RObbQ5.1.2	
RObbQ5.2	
RObbQ5.2.1	
RObbQ5.3	
RObbQ5.3.1	
RObbQ5.3.2	
RObbQ5.3.3	
RObbQ5.4	
RObbQ5.5	
RObbQ5.6	
RObbQ6	
RObbQ6.1	
RObbQ6.1.1	
RObbQ6.1.2	
RObbQ6.1.3	
RObbQ6.2	

Requisito	Classi
RObbQ6.2.1	
RObbQ6.2.2	
RObbQ6.3	
RObbQ6.3.1	
RObbQ6.3.2	
RObbQ6.3.3	
RObbQ6.4	
RObbQ6.4.1	
RObbQ6.4.2	
RObbQ6.4.3	
RObbQ6.4.4	
RObbQ6.5	
RObbV7	
RObbF8	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::NavigationInformation

Requisito	Classi
RObbF8.1	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
RDesF8.1.1	model::navigator::NavigatorImp model::navigator::graph::area::PointOfInterestImp model::navigator::graph::navigationinformation::NavigationInformation presenter::SearchSuggestionsProvider view::HomeViewImp
RDesF8.1.1.1	presenter::SearchSuggestionsProvider view::HomeViewImp
RObbF8.1.2	model::navigator::NavigatorImp model::navigator::graph::area::PointOfInterestImp presenter::HomeActivity presenter::PoiCategoryActivity view::HomeViewImp view::PoiCategoryViewImp
RObbF8.1.2.1	presenter::HomeActivity view::HomeViewImp
RObbF8.1.3	presenter::HomeActivity view::HomeViewImp
RObbF8.1.4	presenter::HomeActivity view::HomeViewImp
RObbF8.2	presenter::HomeActivity presenter::PoiCategoryActivity view::HomeViewImp view::PoiCategoryViewImp

Requisito	Classi
RObbF8.3	model::navigator::NavigatorImp model::navigator::algorithm::DijkstraPathFinder model::navigator::graph::MapGraph model::navigator::graph::area::RegionOfInterestImp model::navigator::graph::vertex::VertexImp
RDesF8.3.1	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
RDesF8.3.1.1	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
RDesF8.3.1.2	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
RDesF8.3.1.3	model::navigator::NavigatorImp model::navigator::algorithm::DijkstraPathFinder

Requisito	Classi
RObbF8.4	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::LocalBinder model::navigator::NavigatorImp
RObbF8.4.1	presenter::HomeActivity presenter::MainActivity presenter::NavigationActivity view::HomeViewImp view::NavigationViewImp
RObbF8.4.1.1	presenter::HomeActivity presenter::NavigationActivity view::HomeViewImp view::NavigationViewImp
RObbF8.4.1.2	presenter::HomeActivity presenter::NavigationActivity view::HomeViewImp view::NavigationViewImp
RObbF8.4.1.3	presenter::HomeActivity presenter::NavigationActivity view::HomeViewImp view::NavigationViewImp
RObbF8.4.2	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::LocalBinder model::compass::Compass model::navigator::NavigatorImp model::navigator::ProcessedInformationImp model::navigator::graph::navigationinformation::BasicInfo model::navigator::graph::navigationinformation::DetailedInfo model::navigator::graph::navigationinformation::NavigationInfo model::navigator::graph::navigationinformation::PhotoInfo

Requisito	Classi
RObbF8.4.2.1	model::navigator::NavigatorImp model::navigator::graph::navigationinformation::BasicInfo model::navigator::graph::navigationinformation::Navigation presenter::NavigationActivity presenter::NavigationAdapter view::NavigationViewImp
RDesF8.4.2.2	model::navigator::NavigatorImp model::navigator::graph::navigationinformation::Navigation
RDesF8.4.2.3	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
ROpzF8.4.2.4	model::navigator::NavigatorImp model::navigator::ProcessedInformationImp
ROpzF8.4.2.5	model::beacon::BeaconManagerAdapter model::navigator::NavigatorImp presenter::NavigationActivity view::NavigationViewImp
ROpzF8.4.2.6	model::NavigationManagerImp model::NoBeaconSeenException model::beacon::BeaconManagerAdapter presenter::NavigationActivity view::NavigationViewImp
RDesF8.4.3	model::navigator::Navigator model::navigator::graph::navigationinformation::DetailedInfo

Requisito	Classi
RDesF8.4.3.1	model::navigator::NavigatorImp model::navigator::graph::navigationinformation::PhotoInfo model::navigator::graph::navigationinformation::PhotoResult presenter::ImageAdapter presenter::ImageDetailActivity presenter::ImageListFragment view::ImageDetailViewImp
ROpzF8.4.3.2	model::navigator::NavigatorImp model::navigator::graph::navigationinformation::DetailedInfo presenter::DetailedInformationActivity view::DetailedInformationViewImp
ROpzF8.4.3.3	model::navigator::NavigatorImp presenter::NavigationActivity view::NavigationViewImp
RDesF8.4.3.4	presenter::DetailedInformationActivity view::DetailedInformationViewImp
RObbF8.4.4	presenter::HomeActivity presenter::PoiCategoryActivity view::HomeViewImp view::PoiCategoryViewImp
RObbF8.5	model::MessageSendType model::NavigationManagerImp model::ServiceConnectionImp model::beacon::LocalBinder model::navigator::NavigatorImp
RObbF8.6	presenter::NavigationActivity view::NavigationViewImp
RObbF8.7	model::dataaccess::dao::BuildingTable model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::NavigationActivity view::NavigationViewImp

Requisito	Classi
RObbF8.8	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::DatabaseService presenter::HomeActivity view::HomeViewImp
RObbF9	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF9.1	model::beacon::BeaconManagerAdapter
RObbF9.1.1	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp model::navigator::NavigatorImp model::navigator::graph::area::PointOfInterestImp
RObbF9.1.2	model::beacon::MyBeaconImp model::navigator::Navigator model::navigator::graph::area::PointOfInterestImp
RObbF9.2	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF9.3	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF9.4	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp model::beacon::MyDistanceCalculator
RObbF9.5	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF9.6	model::beacon::BeaconManagerAdapter model::beacon::MyBeaconImp
RObbF10	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::LocalBinder model::navigator::BuildingInformation model::navigator::BuildingMapImp
RObbF10.1	model::navigator::BuildingMapImp

Requisito	Classi
ROpzF10.2	model::navigator::BuildingInformation model::navigator::BuildingMapImp model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation
ROpzF10.2.1	model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation
ROpzF10.2.2	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::BeaconManagerAdapter model::beacon::LocalBinder model::navigator::BuildingMapImp
RObbF10.2.3	model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation
RObbF10.2.4	model::navigator::graph::area::PointOfInterestImp model::navigator::graph::area::PointOfInterestInformation
RObbF10.3	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbF10.4	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbF10.5	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbF10.6	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation
RObbQ10.7	
RObbQ10.8	
RObbF10.9	model::beacon::BeaconManagerAdapter model::navigator::BuildingInformation presenter::HomeActivity

Requisito	Classi
RObbF10.10	model::beacon::BeaconManagerAdapter model::dataaccess::service::BuildingService model::navigator::BuildingInformation presenter::HomeActivity view::HomeViewImp
ROpzF11	model::usersetting::SettingImp
ROpzF11.1	model::usersetting::PathPreference model::usersetting::SettingImp
ROpzF11.1.1	model::usersetting::SettingImp
ROpzF11.1.1.1	model::usersetting::SettingImp presenter::PreferencesActivity view::PreferencesViewImp
ROpzF11.1.1.2	model::usersetting::SettingImp
RObbF11.1.1.3	model::usersetting::SettingImp
RDesF11.1.2	model::usersetting::InstructionPreference model::usersetting::SettingImp
RObbF11.1.2.1	model::navigator::ProcessedInformationImp model::navigator::graph::navigationinformation::BasicInfo model::navigator::graph::navigationinformation::DetailedInfo model::navigator::graph::navigationinformation::NavigationInfo
RDesF11.1.2.2	model::usersetting::SettingImp
RDesF11.1.2.3	model::usersetting::SettingImp
RDesF11.1.2.4	model::usersetting::SettingImp
RDesF11.1.2.5	model::usersetting::SettingImp

Requisito	Classi
RDesF11.2	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
RDesF11.2.1	model::dataaccess::dao::BuildingContract model::dataaccess::dao::CategoryContract model::dataaccess::dao::DaoFactoryHelper model::dataaccess::dao::EdgeContract model::dataaccess::dao::EdgeTypeContract model::dataaccess::dao::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
RDesF11.2.1.1	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService

Requisito	Classi
RDesF11.2.1.2	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 <u>model::dataaccess::service::BuildingService</u>
RDesF11.2.1.3	<u>model::dataaccess::dao::SQLiteBuildingDao</u> <u>model::dataaccess::service::BuildingService</u>
RDesF11.2.1.4	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 <u>model::dataaccess::service::RegionOfInterestService</u>
RDesF11.2.1.4.1	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapViewImp

Requisito	Classi
RDesF11.2.1.4.2	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF11.2.1.4.3	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF11.2.1.4.4	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF11.2.1.4.5	model::dataaccess::dao::SQLiteBuildingDao model::dataaccess::service::BuildingService presenter::LocalMapActivity view::LocalMapManagerViewImp
RDesF11.2.2	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
RDesF11.2.2.1	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp

Requisito	Classi
RDesF11.2.2.1.1	view::RemoteMapManagerViewImp
RDesF11.2.2.2	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
RDesF11.2.2.3	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF11.2.2.3.1	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF11.2.2.3.2	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp

Requisito	Classi
RDesF11.2.2.3.3	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF11.2.2.3.4	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF11.2.2.3.5	model::dataaccess::dao::RemoteBuildingDao model::dataaccess::service::BuildingService presenter::RemoteMapManagerActivity view::RemoteMapManagerViewImp
RDesF11.2.2.4	model::dataaccess::service::BuildingService presenter::LocalMapActivity presenter::RemoteMapManagerActivity view::LocalMapManagerViewImp view::RemoteMapManagerViewImp
RDesF11.2.3	model::dataaccess::service::BuildingService presenter::MapDownloaderActivity presenter::RemoteMapManagerActivity view::MapDownloaderViewImp view::RemoteMapManagerViewImp
RObbF11.3	model::usersetting::DeveloperCodeManager model::usersetting::SettingImp
RObbF11.3.1	view::DeveloperUnlockerView
RObbF11.3.2	view::DeveloperUnlockerViewImp
RObbF11.3.3	presenter::DeveloperUnlockerActivity view::DeveloperUnlockerViewImp
ROpzF12	presenter::HelpActivity view::HelpViewImp
RObbF13	view::LoggingViewImp
RObbF13.1	view::LoggingViewImp

Requisito	Classi
RObbF13.2	view::LoggingViewImp
RObbF13.3	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::LocalBinder model::beacon::LoggerImp presenter::MainDeveloperActivity presenter::MainDeveloperPresenter view::MainDeveloperViewImp
RDesF13.3.1	model::InformationManager model::beacon::LoggerImp
RDesF13.3.2	model::InformationManager model::beacon::LoggerImp
RDesF13.3.3	model::InformationManager model::beacon::LoggerImp
RDesF13.3.4	presenter::LogInformationActivity view::LogInformationViewImp
RDesF13.3.5	model::InformationManagerImp model::MessageSendType model::ServiceConnectionImp model::beacon::LocalBinder model::beacon::LoggerImp
ROpzF13.4	view::LoggingViewImp
RObbF13.5	view::LoggingViewImp
RObbF13.6	presenter::LoggingActivity view::LoggingViewImp
RObbF13.7	presenter::NearbyPoiActivity view::NearbyPoiViewImp
RObbF13.8	presenter::LoggingActivity view::LoggingViewImp
RObbF13.9	presenter::LoggingActivity view::LoggingViewImp

Requisito	Classi
ROpzQ14	
RDesP15	
ROpzP16	
RObbQ17	

Tabella 2: Tabella requisiti / classi

7.3 Tracciamento Metodi - test di unità

Metodo	Test
model::InformationManagerImp::- addListener()	TU156
model::InformationManagerImp::- getAllVisibleBeacons()	TU41
model::InformationManagerImp::- getBuildingMap()	TU41
model::InformationManagerImp::- getDatabaseService()	TU41
model::InformationManagerImp::- getNearbyPOIs()	TU41
model::InformationManagerImp::- removeListener()	TU156
model::InformationManagerImp::- saveRecordedBeaconInformation()	TU43
model::InformationManagerImp::- startRecordingBeacons()	TU43
model::NavigationManagerImp::- addListener()	TU42
model::NavigationManagerImp::- getAllNavigationInstruction()	TU44

Metodo	Test
model::NavigationManagerImp::- getNextInstruction()	TU44
model::NavigationManagerImp::- removeListener()	TU42
model::NavigationManagerImp::- startNavigation()	TU44
model::NavigationManagerImp::- stopNavigation()	TU44
model::beacon::- BeaconManagerAdapter::- modifyScanPeriod()	TU40
model::beacon::- BeaconManagerAdapter::- setBackgroundMode()	TU39
model::beacon::Logger::add()	TU37
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

Metodo	Test
model::beacon::MyBeacon::- getTxPower()	TU36
model::beacon::MyBeacon::- getUUID()	TU36
model::compass::Compass::- getLastCoordinate()	TU103
model::compass::Compass::- registerListener()	TU101
model::compass::Compass::- unregisterListener()	TU102
model::dataaccess::dao::- BuildingTable::getAddress()	TU71
model::dataaccess::dao::- BuildingTable::getDescription()	TU71
model::dataaccess::dao::- BuildingTable::getId()	TU71
model::dataaccess::dao::- BuildingTable::getMajor()	TU71
model::dataaccess::dao::- BuildingTable::getName()	TU71
model::dataaccess::dao::- BuildingTable::getOpeningHours()	TU71
model::dataaccess::dao::- BuildingTable::getSize()	TU71
model::dataaccess::dao::- BuildingTable::getUUID()	TU71
model::dataaccess::dao::- BuildingTable::getVersion()	TU71
model::dataaccess::dao::- CategoryTable::getDescription()	TU77

Metodo	Test
model::dataaccess::dao::- CategoryTable::getId()	TU77
model::dataaccess::dao::- DaoFactoryHelper::getInstance()	TU153
model::dataaccess::dao::- DaoFactoryHelper::- getRemoteDaoFactory()	TU61
model::dataaccess::dao::- DaoFactoryHelper::- getSQLiteDaoFactory()	TU61
model::dataaccess::dao::- EdgeTable::getAction()	TU75
model::dataaccess::dao::- EdgeTable::getCoordinate()	TU75
model::dataaccess::dao::- EdgeTable::getDistance()	TU75
model::dataaccess::dao::- EdgeTable::getEndROI()	TU75
model::dataaccess::dao::- EdgeTable::getId()	TU75
model::dataaccess::dao::- EdgeTable::getLongDescription()	TU75
model::dataaccess::dao::- EdgeTable::getStartROI()	TU75
model::dataaccess::dao::- EdgeTable::getTypeId()	TU75
model::dataaccess::dao::- EdgeTypeTable::getId()	TU76
model::dataaccess::dao::- EdgeTypeTable::getTypeName()	TU76

Metodo	Test
model::dataaccess::- dao::MapsDbHelper::- getRemoteDatabaseURL()	TU100
model::dataaccess::dao::- MapsDbHelper::onCreate()	TU98
model::dataaccess::dao::- MapsDbHelper::onUpgrade()	TU99
model::dataaccess::dao::- PhotoTable::getEdgeId()	TU78
model::dataaccess::dao::- PhotoTable::getId()	TU78
model::dataaccess::dao::- PhotoTable::getUrl()	TU78
model::dataaccess::dao::- PointOfInterestTable::- getCategoryId()	TU72
model::dataaccess::dao::- PointOfInterestTable::- getDescription()	TU72
model::dataaccess::dao::- PointOfInterestTable::getId()	TU72
model::dataaccess::dao::- PointOfInterestTable::getName()	TU72
model::dataaccess::dao::- RegionOfInterestTable::getId()	TU73
model::dataaccess::dao::- RegionOfInterestTable::getMajor()	TU73
model::dataaccess::dao::- RegionOfInterestTable::getMinor()	TU73
model::dataaccess::dao::- RegionOfInterestTable::getUUID()	TU73

Metodo	Test
model::dataaccess::dao::- RemoteBuildingDao::- fromJSONToTable()	TU63
model::dataaccess::dao::- RemoteCategoryDao::- fromJSONToTable()	TU69
model::dataaccess::dao::- RemoteDaoFactory::- getBuildingDao()	TU62
model::dataaccess::dao::- RemoteDaoFactory::- getCategoryDao()	TU62
model::dataaccess::dao::- RemoteDaoFactory::getEdgeDao()	TU62
model::dataaccess::dao::- RemoteDaoFactory::- getEdgeTypeDao()	TU62
model::dataaccess::dao::- RemoteDaoFactory::getPhotoDao()	TU62
model::dataaccess::dao::- RemoteDaoFactory::- getPointOfInterestDao()	TU62
model::dataaccess::dao::- RemoteDaoFactory::- getRegionOfInterestDao()	TU62
model::dataaccess::dao::- RemoteDaoFactory::- getRoiPoiDao()	TU62
model::dataaccess::- dao::RemoteEdgeDao::- fromJSONToTable()	TU67

Metodo	Test
model::dataaccess::dao::- RemoteEdgeTypeDao:::- fromJSONToTable()	TU68
model::dataaccess::- dao::RemotePhotoDao:::- fromJSONToTable()	TU70
model::dataaccess::dao::- RemotePointOfInterestDao:::- fromJSONToTable()	TU64
model::dataaccess::dao::- RemoteRegionOfInterestDao:::- fromJSONToTable()	TU65
model::dataaccess::dao::- RemoteRoiPoiDao:::- fromJSONToTable()	TU66
model::dataaccess::dao::- RoiPoiTable::getPoiID()	TU74
model::dataaccess::dao::- RoiPoiTable::getRoiID()	TU74
model::dataaccess::dao::SQLDao::- delete()	TU97
model::dataaccess::dao::SQLDao::- insert()	TU97
model::dataaccess::dao::SQLDao::- query()	TU97
model::dataaccess::dao::SQLDao::- rawQuery()	TU97
model::dataaccess::dao::SQLDao::- update()	TU97
model::dataaccess::dao::- SQLiteBuildingDao:::- cursorToType()	TU81

Metodo	Test
model::dataaccess::dao::- SQLiteBuildingDao::- deleteBuilding()	TU80
model::dataaccess::dao::- SQLiteBuildingDao::- findAllBuildings()	TU80
model::dataaccess::dao::- SQLiteBuildingDao::- findBuildingById()	TU80
model::dataaccess::dao::- SQLiteBuildingDao::- findBuildingByMajor()	TU80
model::dataaccess::dao::- SQLiteBuildingDao::- insertBuilding()	TU80
model::dataaccess::dao::- SQLiteBuildingDao::- isBuildingMapPresent()	TU82
model::dataaccess::dao::- SQLiteBuildingDao::- updateBuilding()	TU80
model::dataaccess::dao::- SQLiteCategoryDao::- cursorToType()	TU92
model::dataaccess::dao::- SQLiteCategoryDao::- deleteCategory()	TU91
model::dataaccess::dao::- SQLiteCategoryDao::- findCategory()	TU91
model::dataaccess::dao::- SQLiteCategoryDao::- insertCategory()	TU91

Metodo	Test
model::dataaccess::dao::- SQLiteCategoryDao::- updateCategory()	TU91
model::dataaccess::dao::- SQLiteDaoFactory::- getBuildingDao()	TU79
model::dataaccess::dao::- SQLiteDaoFactory::- getCategoryDao()	TU79
model::dataaccess::dao::- SQLiteDaoFactory::getEdgeDao()	TU79
model::dataaccess::dao::- SQLiteDaoFactory::- getEdgeTypeDao()	TU79
model::dataaccess::dao::- SQLiteDaoFactory::getPhotoDao()	TU79
model::dataaccess::dao::- SQLiteDaoFactory::- getPointOfInterestDao()	TU79
model::dataaccess::dao::- SQLiteDaoFactory::- getRegionOfInterestDao()	TU79
model::dataaccess::dao::- SQLiteDaoFactory::getRoiPoiDao()	TU79
model::dataaccess::dao::- SQLiteEdgeDao::cursorToType()	TU90
model::dataaccess::dao::- SQLiteEdgeDao::deleteEdge()	TU89
model::dataaccess::- dao::SQLiteEdgeDao::- findAllEdgesOfBuilding()	TU89

Metodo	Test
model::dataaccess::dao::- SQLiteEdgeDao::findEdge()	TU89
model::dataaccess::dao::- SQLiteEdgeDao::insertEdge()	TU89
model::dataaccess::dao::- SQLiteEdgeDao::updateEdge()	TU89
model::dataaccess::dao::- SQLiteEdgeTypeDao::- cursorToType()	TU94
model::dataaccess::dao::- SQLiteEdgeTypeDao::- deleteEdgeType()	TU93
model::dataaccess::dao::- SQLiteEdgeTypeDao::- findEdgeType()	TU93
model::dataaccess::dao::- SQLiteEdgeTypeDao::- insertEdgeType()	TU93
model::dataaccess::dao::- SQLiteEdgeTypeDao::- updateEdgeType()	TU93
model::dataaccess::dao::- SQLitePhotoDao::cursorToType()	TU96
model::dataaccess::dao::- SQLitePhotoDao::deletePhoto()	TU95
model::dataaccess::- dao::SQLitePhotoDao::- findAllPhotosOfEdge()	TU95
model::dataaccess::dao::- SQLitePhotoDao::findPhoto()	TU95
model::dataaccess::dao::- SQLitePhotoDao::insertPhoto()	TU95

Metodo	Test
model::dataaccess::dao::- SQLitePhotoDao::updatePhoto()	TU95
model::dataaccess::dao::- SQLitePointOfInterestDao::: cursorToType()	TU84
model::dataaccess::dao::- SQLitePointOfInterestDao::: deletePointOfInterest()	TU83
model::dataaccess::dao::- SQLitePointOfInterestDao::: findAllPointsWithMajor()	TU83
model::dataaccess::dao::- SQLitePointOfInterestDao::: findPointOfInterest()	TU83
model::dataaccess::dao::- SQLitePointOfInterestDao::: insertPointOfInterest()	TU83
model::dataaccess::dao::- SQLitePointOfInterestDao::: updatePointOfInterest()	TU83
model::dataaccess::dao::- SQLiteRegionOfInterestDao::: cursorToType()	TU86
model::dataaccess::dao::- SQLiteRegionOfInterestDao::: deleteRegionOfInterest()	TU85
model::dataaccess::dao::- SQLiteRegionOfInterestDao::: findAllRegionsWithMajor()	TU85
model::dataaccess::dao::- SQLiteRegionOfInterestDao::: findRegionOfInterest()	TU85

Metodo	Test
model::dataaccess::dao::- SQLiteRegionOfInterestDao::- insertRegionOfInterest()	TU85
model::dataaccess::dao::- SQLiteRegionOfInterestDao::- updateRegionOfInterest()	TU85
model::dataaccess::dao::- SQLiteRoiPoiDao::cursorToType()	TU88
model::dataaccess::dao::- SQLiteRoiPoiDao::- deleteRoiPoisWherePoi()	TU87
model::dataaccess::dao::- SQLiteRoiPoiDao::- deleteRoiPoisWhereRoi()	TU87
model::dataaccess::dao::- SQLiteRoiPoiDao::- findAllPointsWithRoi()	TU87
model::dataaccess::dao::- SQLiteRoiPoiDao::- findAllRegionsWithPoi()	TU87
model::dataaccess::dao::- SQLiteRoiPoiDao::insertRoiPoi()	TU87
model::dataaccess::dao::- SQLiteRoiPoiDao::updateRoiPoi()	TU87
model::dataaccess::- service::BuildingService::- convertAndInsert()	TU59
model::dataaccess::service::- BuildingService::deleteBuilding()	TU57
model::dataaccess::service::- BuildingService::findAllBuildings()	TU57

Metodo	Test
model::dataaccess::- service::BuildingService::- findAllRemoteBuildings()	TU58
model::dataaccess::- service::BuildingService::- findBuildingByMajor()	TU57
model::dataaccess::- service::BuildingService::- findRemoteBuildingByMajor()	TU58
model::dataaccess::- service::BuildingService::- isBuildingMapPresent()	TU60
model::dataaccess::- service::BuildingService::- isBuildingMapUpdated()	TU60
model::dataaccess::- service::BuildingService::- isRemoteMapPresent()	TU161
model::dataaccess::- service::BuildingService::- updateBuildingMap()	TU60
model::dataaccess::service::- EdgeService::convertAndInsert()	TU51
model::dataaccess::- service::EdgeService::- convertAndInsertEdgeType()	TU52
model::dataaccess::- service::EdgeService::- convertAndInsertPhoto()	TU154
model::dataaccess::service::- EdgeService::deleteEdge()	TU50

Metodo	Test
model::dataaccess::- service::EdgeService::- findAllEdgesOfBuilding()	TU50
model::dataaccess::service::- EdgeService::findEdge()	TU50
model::dataaccess::service::- PhotoService::convertAndInsert()	TU47
model::dataaccess::service::- PhotoService::deletePhoto()	TU46
model::dataaccess::- service::PhotoService::- findAllPhotosOfEdge()	TU46
model::dataaccess::service::- PhotoService::findPhoto()	TU46
model::dataaccess::service::- PointOfInterestService::- convertAndInsert()	TU54
model::dataaccess::service::- PointOfInterestService::- convertAndInsertCategory()	TU55
model::dataaccess::service::- PointOfInterestService::- convertAndInsertRoiPoi()	TU56
model::dataaccess::service::- PointOfInterestService::- deletePointOfInterest()	TU53
model::dataaccess::service::- PointOfInterestService::- findAllPointsWithMajor()	TU53
model::dataaccess::service::- PointOfInterestService::- findAllRegionsWithPoi()	TU160

Metodo	Test
model::dataaccess::service::- PointOfInterestService::- findPointOfInterest()	TU53
model::dataaccess::service::- RegionOfInterestService::- convertAndInsert()	TU49
model::dataaccess::service::- RegionOfInterestService::- deleteRegionOfInterest()	TU48
model::dataaccess::service::- RegionOfInterestService::- findAllPointsWithRoi()	TU159
model::dataaccess::service::- RegionOfInterestService::- findAllRegionsWithMajor()	TU48
model::dataaccess::service::- RegionOfInterestService::- findRegionOfInterest()	TU48
model::dataaccess::service::- RegionOfInterestService::- getTracedRois()	TU157
model::dataaccess::service::- RegionOfInterestService::- setTracedRois()	TU158
model::dataaccess::service::- ServiceHelper::getService()	TU45
model::navigator::- BuildingInformation::getAddress()	TU4
model::navigator::- BuildingInformation:: getDescription()	TU4
model::navigator::- BuildingInformation::getName()	TU4

Metodo	Test
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::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::getNearbyPOIs()	TU6
model::navigator::-BuildingMapImp::getOpeningHours()	TU5
model::navigator::-BuildingMapImp::getSize()	TU5
model::navigator::-BuildingMapImp::getVersion()	TU5

Metodo	Test
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::- ProcessedInformationImp::- getDetailedInstruction()	TU7
model::navigator::- ProcessedInformationImp::- getDirection()	TU7
model::navigator::- ProcessedInformationImp::- getPhotoInstruction()	TU7
model::navigator::- ProcessedInformationImp::- 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

Metodo	Test
model::navigator::graph::- MapGraph::getGraph()	TU31
model::navigator::graph::- area::PointOfInterestImp::- getAllBelongingROIs()	TU25
model::navigator::graph::area::- PointOfInterestImp::getCategory()	TU24
model::navigator::graph::area::- 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

Metodo	Test
model::navigator::graph::- edge::DefaultEdge::- getDetailedInformation()	TU17
model::navigator::graph::edge::- DefaultEdge::getWeight()	TU22
model::navigator::graph::- edge::ElevatorEdge::- getBasicInformation()	TU19
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

Metodo	Test
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::-getId()	TU11
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
model::usersetting::SettingImp::-setPathPreference()	TU1

Metodo	Test
model::usersetting::SettingImp::- unlockDeveloper()	TU3
presenter::- DetailedInformationActivity::- updateDetailedDescription()	TU131
presenter::- DetailedInformationActivity::- updatePhoto()	TU131
presenter::- DeveloperUnlockerActivity::- unlockDeveloper()	TU122
presenter::HelpActivity::onCreate()	TU128
presenter::HomeActivity::- enableSuggestions()	TU110
presenter::HomeActivity::- showHelp()	TU113
presenter::HomeActivity::- showLocalMaps()	TU115
presenter::HomeActivity::- showPoisCategory()	TU111
presenter::HomeActivity::- showPreferences()	TU112
presenter::HomeActivity::- startNavigation()	TU114
presenter::HomeActivity::- updateBuildingAddress()	TU109
presenter::HomeActivity::- updateBuildingDescription()	TU106
presenter::HomeActivity::- updateBuildingName()	TU105

Metodo	Test
presenter::HomeActivity::- updateBuildingOpeningHours()	TU107
presenter::HomeActivity::- updatePoiCategoryList()	TU108
presenter::ImageAdapter::- getCount()	TU119
presenter::ImageAdapter::getItem()	TU119
presenter::ImageAdapter::- getItemId()	TU132
presenter::ImageAdapter::- getView()	TU119
presenter::ImageDetailActivity::- onCreate()	TU129
presenter::ImageListFragment::- newInstance()	TU120
presenter::ImageListFragment::- onCreateView()	TU120
presenter::ImageListFragment::- onItemClick()	TU120
presenter::LocalMapActivity::- deleteMap()	TU127
presenter::LocalMapActivity::- updateMap()	TU127
presenter::LoggingActivity::- stopLogging()	TU134
presenter::- LogInformationActivity::- deleteLog()	TU135
presenter::MainActivity::- onCreate()	TU133

Metodo	Test
presenter::MainDeveloperActivity::- showDetailedLog()	TU123
presenter::MainDeveloperActivity::- startNewLog()	TU123
presenter::- MainDeveloperPresenter::- isDeveloper()	TU121
presenter::- MainDeveloperPresenter::- startDeveloperOptions()	TU121
presenter::- MainDeveloperPresenter::- startDeveloperUnlocker()	TU121
presenter::- MapDownloaderActivity::- downloadFinished()	TU125
presenter::NavigationActivity::- informationUpdate()	TU117
presenter::NavigationActivity::- pathError()	TU117
presenter::NavigationActivity::- showDetailedInformation()	TU118
presenter::NavigationAdapter::- getCount()	TU136
presenter::NavigationAdapter::- getItem()	TU136
presenter::NavigationAdapter::- getItemId()	TU136
presenter::NavigationAdapter::- getView()	TU136
presenter::NearbyPoiActivity::- onCreate()	TU130

Metodo	Test
presenter::PoiCategoryActivity::- startNavigation()	TU116
presenter::PreferencesActivity::- savePreferences()	TU124
presenter::- RemoteMapManagerActivity::- downloadMap()	TU126
presenter::- SearchSuggestionsProvider::- getType()	TU104
presenter::- SearchSuggestionsProvider::insert()	TU104
presenter::- SearchSuggestionsProvider::- onCreate()	TU104
presenter::- SearchSuggestionsProvider::query()	TU104
presenter::- SearchSuggestionsProvider::- update()	TU104
view::- DetailedInformationViewImpl::- setDetailedDescription()	TU140
view::- DetailedInformationViewImpl::- setPhoto()	TU140
view::DeveloperUnlockerViewImpl::- showWrongCode()	TU148
view::HelpViewImpl::setHelp()	TU143
view::HomeViewImpl::- setBuildingAddress()	TU137

Metodo	Test
view::HomeViewImp::- setBuildingName()	TU137
view::HomeViewImp::- setBuildingOpeningHours()	TU137
view::HomeViewImp::- setPoiCategoryListAdapter()	TU137
view::ImageDetailViewImp::- setAdapter()	TU142
view::LocalMapManagerViewImp::- refreshMaps()	TU144
view::LocalMapManagerViewImp::- setAdapter()	TU144
view::LoggingViewImp::- setBeaconListAdapter()	TU150
view::LogInformationViewImp::- setBeaconAdapter()	TU151
view::MainDeveloperViewImp::- setLogsAdapter()	TU149
view::MapDownloaderView::- setDowloadingMap()	TU146
view::MapDownloaderView::- setProgressDowload()	TU146
view::NavigationViewImp::- refreshInstructions()	TU138
view::NavigationViewImp::- setInstructionAdapter()	TU138
view::NearbyPoiViewImp::- setAdapter()	TU141
view::PoiCategoryViewImp::- setPoiListAdapter()	TU139

Metodo	Test
view::PreferencesViewImp::- setInstructionPreferences()	TU147
view::PreferencesViewImp::- setPathPreferences()	TU147
view::- RemoteMapManagerViewImp::- setRemoteMaps()	TU145

Tabella 3: Tabella metodi / test unità