

CLIPS

Communication & Localization with Indoor Positioning Systems

UNIVERSITÀ DI PADOVA

DEFINIZIONE DI PRODOTTO v1.00



leaf.gruppo@gmail.com

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

Diario delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
1.00	2016-04-10	Oscar Elia Conti	Responsabile di progetto	Approvazione del documento
0.20	2016-04-10	Cristian Andrigutto	Verificatore	Verifica del documento
0.19	2016-04-10	Davide Castello	Progettista	Aggiunto tracciamento classi-requisiti e viceversa
0.18	2016-04-09	Eduard Bicego	Progettista	Correzioni delle classi ImageDetailActivity, ImageListFragment, ImageListFragmentViewImp, PreferenceSViewImp, LoggingView, LoggingViewImp, DeveloperUnlockerViewImp, 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

Versione	Data	Autore	Ruolo	Descrizione
0.15	2016-04-08	Cristian Andrigutto	Verificatore	Verifica del documento
0.14	2016-04-08	Marco Zanella	Progettista	Correzioni generali su componenti del model
0.13	2016-04-08	Andrea Tombolato	Progettista	Aggiunte componenti view
0.12	2016-04-08	Oscar Elia Conti	Progettista	Aggiunte componenti presenter
0.11	2016-04-07	Cristian Andrigutto	Verificatore	Verifica del documento
0.10	2016-04-07	Marco Zanella	Progettista	Aggiunte componenti model
0.09	2016-04-06	Eduard Bicego	Progettista	Aggiunta sottosezione Metodo e formalismo di specifica
0.06	2016-04-06	Eduard Bicego	Progettista	Aggiungi diagrammi di sequenza Avvio Service, Ranging Beacons e avvio navigazione
0.05	2016-04-03	Eduard Bicego	Progettista	Completata sezione Standard di progetto
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"



Versione	Data	Autore	Ruolo	Descrizione
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	Specifiche dei componenti	4
3.1	Metodo e formalismo di specifica	4
3.2	Sistema CLIPS	5
3.3	Componenti	6
3.3.1	model	6
3.3.2	model::beacon	7
3.3.3	model::compass	8
3.3.4	model::dataaccess	8
3.3.5	model::dataaccess::dao	9
3.3.6	model::dataaccess::service	11
3.3.7	model::navigator	12
3.3.8	model::navigator::algorithm	13
3.3.9	model::navigator::graph	14
3.3.10	model::navigator::graph::area	15
3.3.11	model::navigator::graph::edge	15
3.3.12	model::navigator::graph::navigationinformation	16
3.3.13	model::navigator::graph::vertex	17
3.3.14	model::usersetting	17
3.3.15	presenter	18
3.3.16	view	19
3.4	Classi	21
3.4.1	model::AbsBeaconReceiverManager	21
3.4.2	model::InformationManager	23
3.4.3	model::InformationManagerImp	25
3.4.4	model::MessageSendType	28

3.4.5	model::NavigationManager	29
3.4.6	model::NavigationManagerImp	31
3.4.7	model::NoBeaconSeenException	34
3.4.8	model::ServiceConnectionImp	35
3.4.9	model::beacon::BeaconManagerAdapter	36
3.4.10	model::beacon::BeaconRanger	39
3.4.11	model::beacon::LocalBinder	40
3.4.12	model::beacon::Logger	41
3.4.13	model::beacon::LoggerImp	43
3.4.14	model::beacon::MyBeacon	45
3.4.15	model::beacon::MyBeaconImp	47
3.4.16	model::beacon::MyDistanceCalculator	49
3.4.17	model::compass::Compass	50
3.4.18	model::dataaccess::dao::BuildingContract	52
3.4.19	model::dataaccess::dao::BuildingDao	54
3.4.20	model::dataaccess::dao::BuildingTable	56
3.4.21	model::dataaccess::dao::CategoryContract	58
3.4.22	model::dataaccess::dao::CategoryDao	59
3.4.23	model::dataaccess::dao::CategoryTable	61
3.4.24	model::dataaccess::dao::CursorConverter	62
3.4.25	model::dataaccess::dao::DaoFactoryHelper	63
3.4.26	model::dataaccess::dao::EdgeContract	64
3.4.27	model::dataaccess::dao::EdgeDao	65
3.4.28	model::dataaccess::dao::EdgeTable	67
3.4.29	model::dataaccess::dao::EdgeTypeContract	70
3.4.30	model::dataaccess::dao::EdgeTypeDao	71
3.4.31	model::dataaccess::dao::EdgeTypeTable	72
3.4.32	model::dataaccess::dao::MapsDbContract	73
3.4.33	model::dataaccess::dao::MapsDbHelper	74
3.4.34	model::dataaccess::dao::PhotoContract	76
3.4.35	model::dataaccess::dao::PhotoDao	77
3.4.36	model::dataaccess::dao::PhotoTable	79
3.4.37	model::dataaccess::dao::PointOfInterestContract	80
3.4.38	model::dataaccess::dao::PointOfInterestDao	81
3.4.39	model::dataaccess::dao::PointOfInterestTable	83
3.4.40	model::dataaccess::dao::RegionOfInterestContract	85
3.4.41	model::dataaccess::dao::RegionOfInterestDao	86
3.4.42	model::dataaccess::dao::RegionOfInterestTable	88
3.4.43	model::dataaccess::dao::RemoteBuildingDao	89
3.4.44	model::dataaccess::dao::RemoteCategoryDao	90
3.4.45	model::dataaccess::dao::RemoteDaoFactory	92

3.4.46	model::dataaccess::dao::RemoteEdgeDao	93
3.4.47	model::dataaccess::dao::RemoteEdgeTypeDao	94
3.4.48	model::dataaccess::dao::RemotePhotoDao	95
3.4.49	model::dataaccess::dao::RemotePointOfInterestDao . .	96
3.4.50	model::dataaccess::dao::RemoteRegionOfInterestDao .	97
3.4.51	model::dataaccess::dao::RemoteRoiPoiDao	98
3.4.52	model::dataaccess::dao::RoiPoiContract	99
3.4.53	model::dataaccess::dao::RoiPoiDao	100
3.4.54	model::dataaccess::dao::RoiPoiTable	102
3.4.55	model::dataaccess::dao::SQLDao	104
3.4.56	model::dataaccess::dao::SQLiteBuildingDao	107
3.4.57	model::dataaccess::dao::SQLiteCategoryDao	109
3.4.58	model::dataaccess::dao::SQLiteDaoFactory	111
3.4.59	model::dataaccess::dao::SQLiteEdgeDao	113
3.4.60	model::dataaccess::dao::SQLiteEdgeTypeDao	115
3.4.61	model::dataaccess::dao::SQLitePhotoDao	117
3.4.62	model::dataaccess::dao::SQLitePointOfInterestDao . .	119
3.4.63	model::dataaccess::dao::SQLiteRegionOfInterestDao .	122
3.4.64	model::dataaccess::dao::SQLiteRoiPoiDao	124
3.4.65	model::dataaccess::service::BuildingService	127
3.4.66	model::dataaccess::service::DatabaseService	131
3.4.67	model::dataaccess::service::EdgeService	133
3.4.68	model::dataaccess::service::PhotoService	136
3.4.69	model::dataaccess::service::PointOfInterestService . .	138
3.4.70	model::dataaccess::service::RegionOfInterestService .	142
3.4.71	model::dataaccess::service::ServiceHelper	144
3.4.72	model::navigator::BuildingInformation	146
3.4.73	model::navigator::BuildingMap	148
3.4.74	model::navigator::BuildingMapImp	151
3.4.75	model::navigator::NavigationExceptions	154
3.4.76	model::navigator::Navigator	155
3.4.77	model::navigator::NavigatorImp	157
3.4.78	model::navigator::NoGraphSetException	160
3.4.79	model::navigator::NoNavigationInformationException .	161
3.4.80	model::navigator::PathException	163
3.4.81	model::navigator::ProcessedInformation	164
3.4.82	model::navigator::ProcessedInformationImp	165
3.4.83	model::navigator::algorithm::DijkstraPathFinder . . .	167
3.4.84	model::navigator::algorithm::PathFinder	168
3.4.85	model::navigator::graph::MapGraph	169
3.4.86	model::navigator::graph::area::PointOfInterest	171

3.4.87 model::navigator::graph::area::PointOfInterestImp . . .	172
3.4.88 model::navigator::graph::area::PointOfInterestInformation	174
3.4.89 model::navigator::graph::area::RegionOfInterest	176
3.4.90 model::navigator::graph::area::RegionOfInterestImp . .	178
3.4.91 model::navigator::graph::edge::AbsEnrichedEdge	181
3.4.92 model::navigator::graph::edge::DefaultEdge	184
3.4.93 model::navigator::graph::edge::Edge	185
3.4.94 model::navigator::graph::edge::ElevatorEdge	186
3.4.95 model::navigator::graph::edge::EnrichedEdge	188
3.4.96 model::navigator::graph::edge::StairEdge	189
3.4.97 model::navigator::graph::navigationinformation::BasicInformation	191
3.4.98 model::navigator::graph::navigationinformation::DetailedInformation	192
3.4.99 model::navigator::graph::navigationinformation::NavigationInformation	193
3.4.100 model::navigator::graph::navigationinformation::NavigationInformationImp	194
3.4.101 model::navigator::graph::navigationinformation::PhotoInformation	195
3.4.102 model::navigator::graph::navigationinformation::PhotoRef	196
3.4.103 model::navigator::graph::vertex::Vertex	197
3.4.104 model::navigator::graph::vertex::VertexImp	198
3.4.105 model::usersetting::DeveloperCodeManager	199
3.4.106 model::usersetting::InstructionPreference	200
3.4.107 model::usersetting::PathPreference	200
3.4.108 model::usersetting::Setting	201
3.4.109 model::usersetting::SettingImp	203
3.4.110 presenter::Checker	205
3.4.111 presenter::DatabaseServiceManager	206
3.4.112 presenter::DetailedInformationActivity	207
3.4.113 presenter::DeveloperUnlockerActivity	208
3.4.114 presenter::HelpActivity	209
3.4.115 presenter::HomeActivity	211
3.4.116 presenter::ImageAdapter	213
3.4.117 presenter::ImageDetailActivity	215
3.4.118 presenter::ImageListFragment	216
3.4.119 presenter::ImagePageAdapter	218
3.4.120 presenter::InformationManagerPresenter	219
3.4.121 presenter::LocalMapActivity	220
3.4.122 presenter::LoggingActivity	221
3.4.123 presenter::LogInformationActivity	222
3.4.124 presenter::MainActivity	223
3.4.125 presenter::MainDeveloperActivity	224
3.4.126 presenter::MainDeveloperPresenter	226
3.4.127 presenter::MapDownloaderActivity	227

3.4.128 presenter::NavigationActivity	228
3.4.129 presenter::NavigationAdapter	229
3.4.130 presenter::NavigationManagerPresenter	231
3.4.131 presenter::NearbyPoiActivity	232
3.4.132 presenter::PoiCategoryActivity	233
3.4.133 presenter::PreferencesActivity	234
3.4.134 presenter::RemoteMapManagerActivity	235
3.4.135 presenter::SearchSuggestionsProvider	236
3.4.136 presenter::SettingManager	238
3.4.137 view::DetailedInformationView	239
3.4.138 view::DetailedInformationViewImp	240
3.4.139 view::DeveloperUnlockerView	242
3.4.140 view::DeveloperUnlockerViewImp	243
3.4.141 view::HelpView	244
3.4.142 view::HelpViewImp	245
3.4.143 view::HomeView	246
3.4.144 view::HomeViewImp	248
3.4.145 view::ImageDetailView	251
3.4.146 view::ImageDetailViewImp	252
3.4.147 view::ImageListFragmentView	253
3.4.148 view::ImageListFragmentViewImp	254
3.4.149 view::LocalMapManagerView	255
3.4.150 view::LocalMapManagerViewImp	256
3.4.151 view::LoggingView	257
3.4.152 view::LoggingViewImp	258
3.4.153 view::LogInformationView	259
3.4.154 view::LogInformationViewImp	260
3.4.155 view::MainDeveloperView	262
3.4.156 view::MainDeveloperViewImp	263
3.4.157 view::MainView	264
3.4.158 view::MainViewImp	265
3.4.159 view::MapDownloaderView	266
3.4.160 view::MapDownloaderViewImp	267
3.4.161 view::NavigationView	268
3.4.162 view::NavigationViewImp	269
3.4.163 view::NearbyPoiView	271
3.4.164 view::NearbyPoiViewImp	272
3.4.165 view::PoiCategoryView	273
3.4.166 view::PoiCategoryViewImp	274
3.4.167 view::PreferencesView	275
3.4.168 view::PreferencesViewImp	276

3.4.169	view::RemoteMapView	278
3.4.170	view::RemoteMapViewImp	279
4	Schema base di dati	281
5	Diagrammi di sequenza	282
5.1	Avvio Service per il rilevamento beacon	282
5.2	Elaborazione beacon rilevati e comunicazione broadcast	284
5.3	Avvio navigazione	286
6	Tracciamento	287
6.1	Tracciamento Classi-Requisiti	287
6.2	Requisiti-Classi	287
A	Diagrammi riassuntivi package significativi	288

Elenco delle figure

1	Diagramma dei package - sistema CLIPS	5
2	Componente model	6
3	Componente model::beacon	7
4	Componente model::compass	8
5	Componente model::dataaccess	8
6	Componente model::dataaccess::dao	9
7	Componente model::dataaccess::service	11
8	Componente model::navigator	12
9	Componente model::navigator::algorithm	13
10	Componente model::navigator::graph	14
11	Componente model::navigator::graph::area	15
12	Componente model::navigator::graph::edge	15
13	Componente model::navigator::graph::navigationinformation .	16
14	Componente model::navigator::graph::vertex	17
15	Componente model::usersetting	17
16	Componente presenter	18
17	Componente view	19
18	Classe astratta AbsBeaconReceiverManager	21
19	Interfaccia InformationManager	23
20	Classe InformationManagerImp	25
21	Classe MessageSendType	28
22	Interfaccia NavigationManager	29
23	Classe NavigationManagerImp	31
24	Classe NoBeaconSeenException	34
25	Classe ServiceConnectionImp	35
26	Classe BeaconManagerAdapter	36
27	Interfaccia BeaconRanger	39
28	Classe LocalBinder	40
29	Interfaccia Logger	41
30	Classe LoggerImp	43
31	Interfaccia MyBeacon	45
32	Classe MyBeaconImp	47
33	Classe MyDistanceCalculator	49
34	Classe Compass	50
35	Classe BuildingContract	52
36	Interfaccia BuildingDao	54
37	Classe BuildingTable	56
38	Classe CategoryContract	58
39	Interfaccia CategoryDao	59

40	Classe CategoryTable	61
41	Interfaccia CursorConverter	62
42	Classe DaoFactoryHelper	63
43	Classe EdgeContract	64
44	Interfaccia EdgeDao	65
45	Classe EdgeTable	67
46	Classe EdgeTypeContract	70
47	Interfaccia EdgeTypeDao	71
48	Classe EdgeTypeTable	72
49	Classe MapsDbContract	73
50	Classe MapsDbHelper	74
51	Classe PhotoContract	76
52	Interfaccia PhotoDao	77
53	Classe PhotoTable	79
54	Classe PointOfInterestContract	80
55	Interfaccia PointOfInterestDao	81
56	Classe PointOfInterestTable	83
57	Classe RegionOfInterestContract	85
58	Interfaccia RegionOfInterestDao	86
59	Classe RegionOfInterestTable	88
60	Classe RemoteBuildingDao	89
61	Classe RemoteCategoryDao	90
62	Classe RemoteDaoFactory	92
63	Classe RemoteEdgeDao	93
64	Classe RemoteEdgeTypeDao	94
65	Classe RemotePhotoDao	95
66	Classe RemotePointOfInterestDao	96
67	Classe RemoteRegionOfInterestDao	97
68	Classe RemoteRoiPoiDao	98
69	Classe RoiPoiContract	99
70	Interfaccia RoiPoiDao	100
71	Classe RoiPoiTable	102
72	Classe SQLDao	104
73	Classe SQLiteBuildingDao	107
74	Classe SQLiteCategoryDao	109
75	Classe SQLiteDaoFactory	111
76	Classe SQLiteEdgeDao	113
77	Classe SQLiteEdgeTypeDao	115
78	Classe SQLitePhotoDao	117
79	Classe SQLitePointOfInterestDao	119
80	Classe SQLiteRegionOfInterestDao	122

81	Classe SQLiteRoiPoiDao	124
82	Classe BuildingService	127
83	Interfaccia DatabaseService	131
84	Classe EdgeService	133
85	Classe PhotoService	136
86	Classe PointOfInterestService	138
87	Classe RegionOfInterestService	142
88	Classe ServiceHelper	144
89	Classe BuildingInformation	146
90	Interfaccia BuildingMap	148
91	Classe BuildingMapImp	151
92	Classe astratta NavigationExceptions	154
93	Interfaccia Navigator	155
94	Classe NavigatorImp	157
95	Classe NoGraphSetException	160
96	Classe NoNavigationInformationException	161
97	Classe PathException	163
98	Interfaccia ProcessedInformation	164
99	Classe ProcessedInformationImp	165
100	Classe DijkstraPathFinder	167
101	Interfaccia PathFinder	168
102	Classe MapGraph	169
103	Interfaccia PointOfInterest	171
104	Classe PointOfInterestImp	172
105	Classe PointOfInterestInformation	174
106	Interfaccia RegionOfInterest	176
107	Classe RegionOfInterestImp	178
108	Classe astratta AbsEnrichedEdge	181
109	Classe DefaultEdge	184
110	Interfaccia Edge	185
111	Classe ElevatorEdge	186
112	Interfaccia EnrichedEdge	188
113	Classe StairEdge	189
114	Classe BasicInformation	191
115	Classe DetailedInformation	192
116	Interfaccia NavigationInformation	193
117	Classe NavigationInformationImp	194
118	Classe PhotoInformation	195
119	Classe PhotoRef	196
120	Interfaccia Vertex	197
121	Classe VertexImp	198

122	Classe DeveloperCodeManager	199
123	Classe InstructionPreference	200
124	Classe PathPreference	200
125	Interfaccia Setting	201
126	Classe SettingImp	203
127	Classe Checker	205
128	Classe DatabaseServiceManager	206
129	Classe DetailedInformationActivity	207
130	Classe DeveloperUnlockerActivity	208
131	Classe HelpActivity	209
132	Classe HomeActivity	211
133	Classe ImageAdapter	213
134	Classe ImageDetailActivity	215
135	Classe ImageListFragment	216
136	Classe ImagePageAdapter	218
137	Classe InformationManagerPresenter	219
138	Classe LocalMapActivity	220
139	Classe LoggingActivity	221
140	Classe LogInformationActivity	222
141	Classe MainActivity	223
142	Classe MainDeveloperActivity	224
143	Classe MainDeveloperPresenter	226
144	Classe MapDownloaderActivity	227
145	Classe NavigationActivity	228
146	Classe NavigationAdapter	229
147	Classe NavigationManagerPresenter	231
148	Classe NearbyPoiActivity	232
149	Classe PoiCategoryActivity	233
150	Classe PreferencesActivity	234
151	Classe RemoteMapManagerActivity	235
152	Classe SearchSuggestionsProvider	236
153	Classe SettingManager	238
154	Interfaccia DetailedInformationView	239
155	Classe DetailedInformationViewImp	240
156	Interfaccia DeveloperUnlockerView	242
157	Classe DeveloperUnlockerViewImp	243
158	Interfaccia HelpView	244
159	Classe HelpViewImp	245
160	Interfaccia HomeView	246
161	Classe HomeViewImp	248
162	Interfaccia ImageDetailView	251

163	Classe ImageDetailViewImp	252
164	Interfaccia ImageListFragmentView	253
165	Classe ImageListFragmentManagerView	254
166	Interfaccia LocalMapManagerView	255
167	Classe LocalMapManagerViewImp	256
168	Interfaccia LoggingView	257
169	Classe LoggingViewImp	258
170	Interfaccia LogInformationView	259
171	Classe LogInformationViewImp	260
172	Interfaccia MainDeveloperView	262
173	Classe MainDeveloperViewImp	263
174	Interfaccia MainView	264
175	Classe MainViewImp	265
176	Interfaccia MapDownloaderView	266
177	Classe MapDownloaderViewImp	267
178	Interfaccia NavigationView	268
179	Classe NavigationViewImp	269
180	Interfaccia NearbyPoiView	271
181	Classe NearbyPoiViewImp	272
182	Interfaccia PoiCategoryView	273
183	Classe PoiCategoryViewImp	274
184	Interfaccia PreferencesView	275
185	Classe PreferencesViewImp	276
186	Interfaccia RemoteMapManagerView	278
187	Classe RemoteMapManagerViewImp	279
188	Schema UML - base di dati	281
189	Diagramma di sequenza - Avvio di un service ₉ per il rilevamento beacon	283
190	Diagramma di sequenza - Elaborazione beacon rilevati e comunicazione broadcast	285
191	Diagramma di sequenza - Avvio navigazione	286
192	Diagramma delle classi - model	289
193	Diagramma delle classi - model::navigator	290
194	Diagramma delle classi - model::dataaccess	291
195	Diagramma delle classi - presenter	292

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 v4.00*.

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 v4.00* nel quale verranno raccolte le spiegazioni di terminologia tecnica o ambigua, abbreviazioni ed acronimi. Per evidenziare un termine presente in tale documento, esso verrà marcato con il pedice _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 v4.00*.

1.4.2 Riferimenti informativi

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

- Documentazione SQLite: <https://www.sqlite.org/docs.html>;
- Documentazione JavaDoc JGraphT Library: <http://jgrapht.org/javadoc/>;
- Materiale di riferimento del corso di Ingegneria del Software₉ - Diagrammi delle classi: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E03.pdf>;
- Materiale di riferimento del corso di Ingegneria del Software₉ - Model View Presenter: http://www.math.unipd.it/~rcardin/sweb/Design%20Pattern%20Architetturali%20-%20Model%20View%20Controller_4x4.pdf;
- Design Pattern: elementi per il riuso di software ad oggetti - Gamma, Helm, Johnson, Vlissides - editore Pearson - 2002;
- UML e ingegneria del software: dalla teoria alla pratica - Luca Vetti Tagliati - 2015.

2 Standard di progetto

2.1 Standard di documentazione del codice

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

2.2 Standard di denominazione di entità e relazioni

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

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

2.3 Standard di programmazione

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

2.4 Strumenti di lavoro e procedure

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

3 Specifica dei componenti

3.1 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*.

3.2 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.

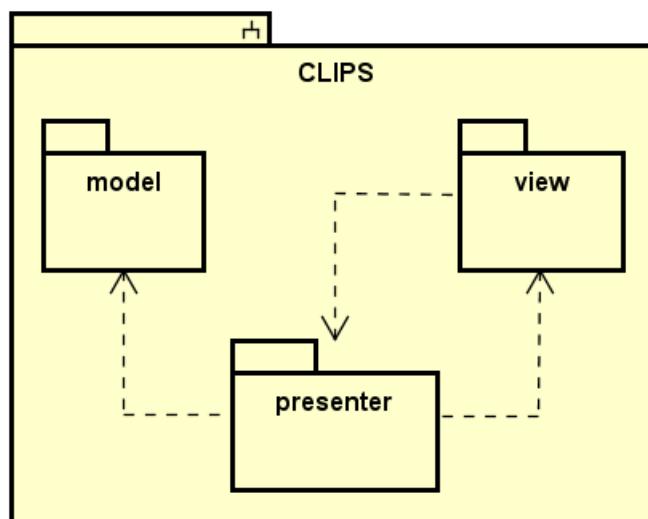


Figura 1: Diagramma dei package - sistema CLIPS

3.3 Componenti

3.3.1 model

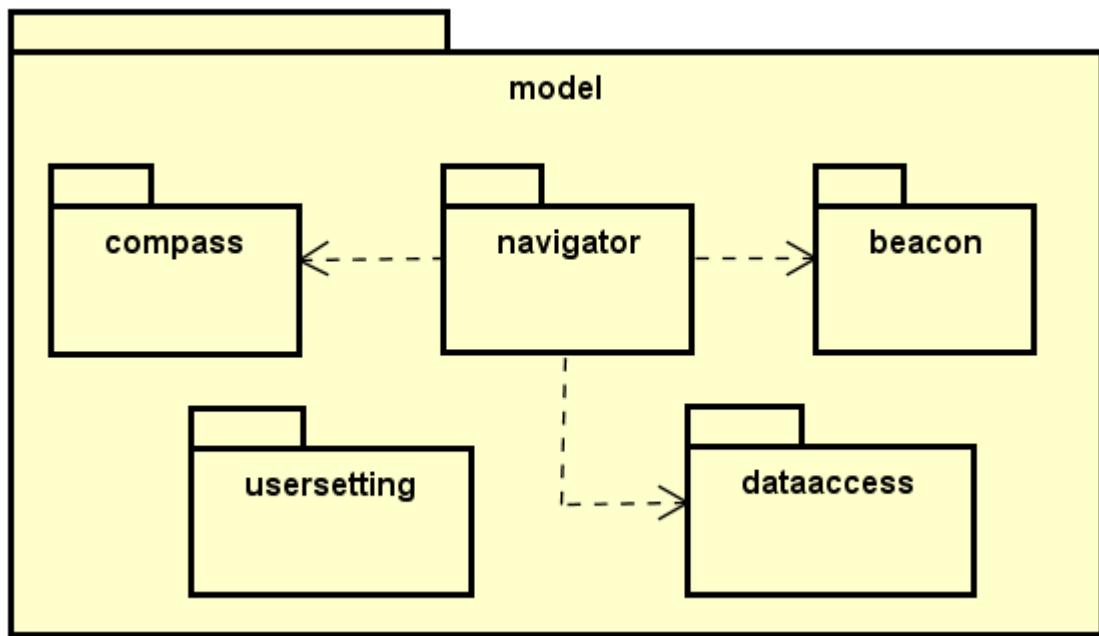


Figura 2: Componente model

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

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

- **Interfacce Contenute:**

- InformationManager;
- NavigationManager.

3.3.2 model::beacon

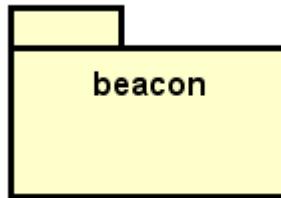


Figura 3: Componente model::beacon

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

3.3.3 model::compass

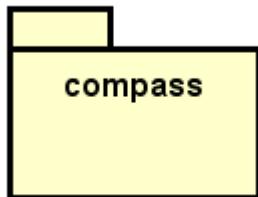


Figura 4: Componente model::compass

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

3.3.4 model::dataaccess

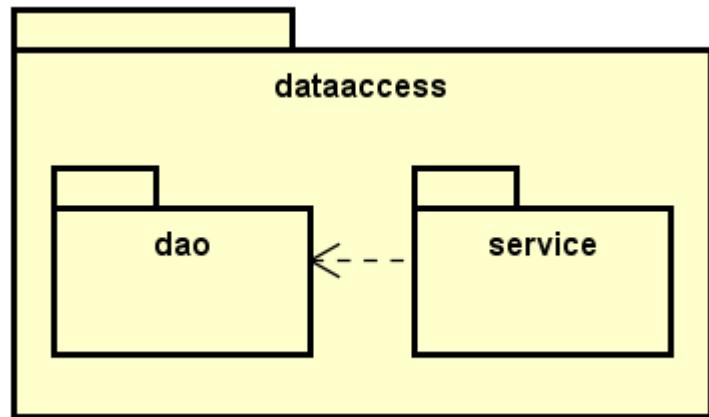


Figura 5: Componente model::dataaccess

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

- dao;
- service.

3.3.5 model::dataaccess::dao

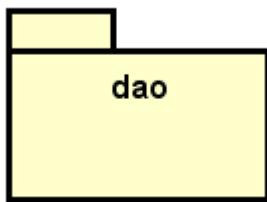


Figura 6: Componente model::dataaccess::dao

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

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

- **Interfacce Contenute:**

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

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

3.3.6 model::dataaccess::service

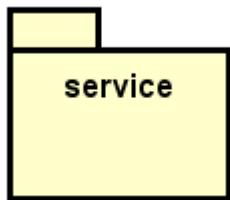


Figura 7: Componente model::dataaccess::service

- **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.

3.3.7 model::navigator

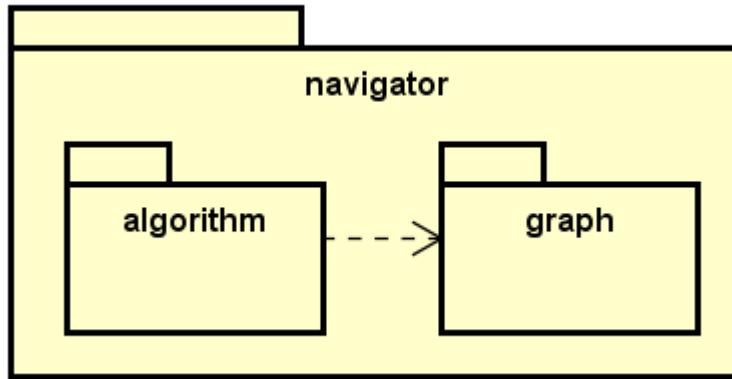


Figura 8: Componente model::navigator

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

- BuildingMap;
- Navigator;
- ProcessedInformation.

3.3.8 model::navigator::algorithm

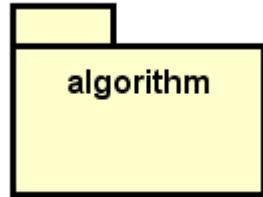


Figura 9: Componente model::navigator::algorithm

- **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.

3.3.9 model::navigator::graph

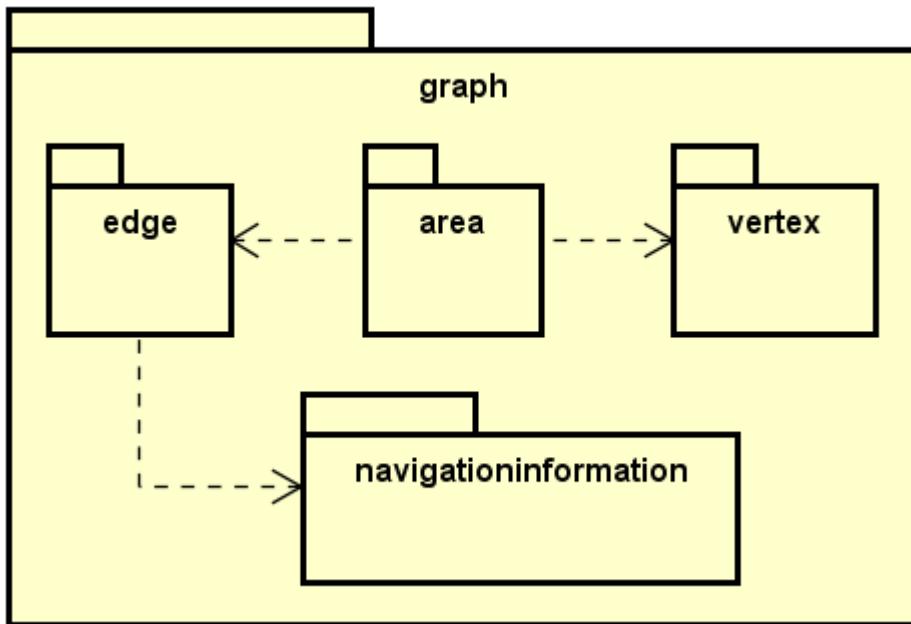


Figura 10: Componente `model::navigator::graph`

- **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`.

3.3.10 model::navigator::graph::area

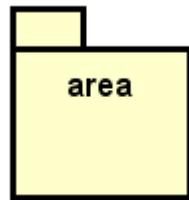


Figura 11: Componente model::navigator::graph::area

- **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.

3.3.11 model::navigator::graph::edge

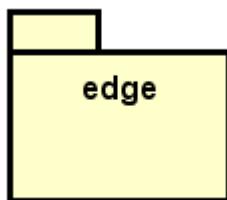


Figura 12: Componente model::navigator::graph::edge

- **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.

3.3.12 model::navigator::graph::navigationinformation



Figura 13: 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.

3.3.13 model::navigator::graph::vertex

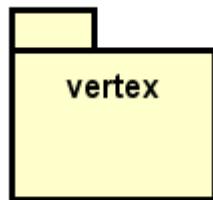


Figura 14: Componente model::navigator::graph::vertex

- **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.

3.3.14 model::usersetting

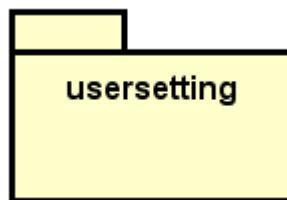


Figura 15: Componente model::usersetting

- **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.

3.3.15 presenter

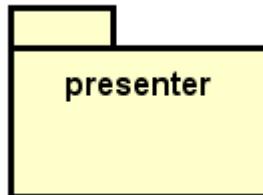


Figura 16: Componente presenter

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

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

3.3.16 view

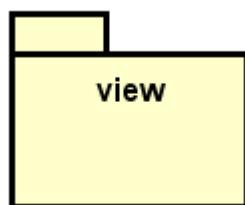


Figura 17: Componente view

- **Descrizione:** Package per il componente View del pattern architetture MVP. Questo pacchetto contiene tutte le classi che compongono la presentation logic;

- **Classi Contenute:**

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

- **Interfacce Contenute:**

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

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

3.4 Classi

3.4.1 model::AbsBeaconReceiverManager

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

Figura 18: Classe astratta AbsBeaconReceiverManager

Nome: *AbsBeaconReceiverManager*;

Tipo: Classe astratta;

Estende:

- ServiceConnectionImp.

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Attributi:

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

Metodi:

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

Argomenti:

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

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

3.4.2 model::InformationManager

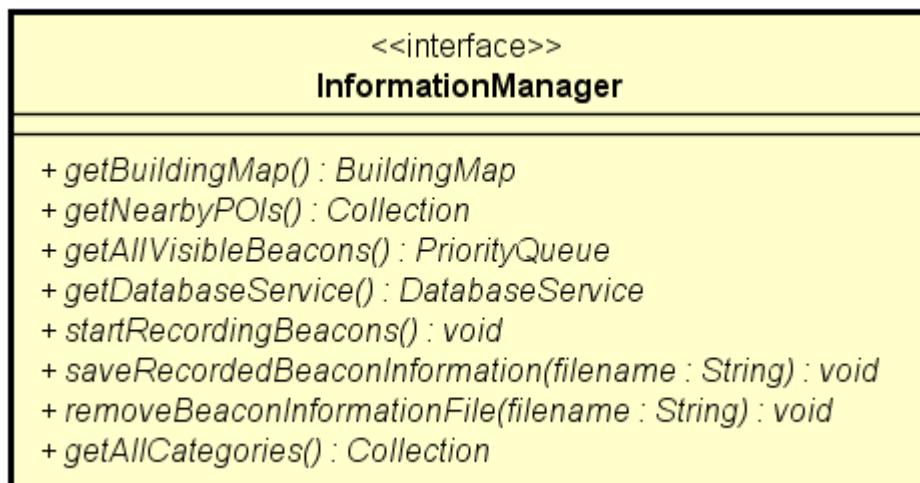


Figura 19: Interfaccia InformationManager

Nome: *InformationManager*;

Tipo: Interfaccia;

Visibilità: public;

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

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

Metodi:

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

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

Argomenti:

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

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

Argomenti:

- filename : String
Nome del file da rimuovere

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

Argomenti:

- filename : String
Nome da dare al file da salvare

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

3.4.3 model::InformationManagerImp

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

Figura 20: Classe InformationManagerImp

Nome: InformationManagerImp;

Tipo: Classe;

Estende:

- AbsBeaconReceiverManager.

Implementa:

- InformationManager.

Visibilità: public;

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

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

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

Attributi:

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

Metodi:

- + `getAllCategories()` : `Collection<String>`
Metodo che ritorna tutte le categorie di POI presenti 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 poichè 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 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

- + **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

- - **loadMap() : BuildingMap**

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

- + **onReceive() : void**

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

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

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

Argomenti:

– **filename : String**

Nome del file da rimuovere

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

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

Argomenti:

– **filename : String**

Nome del file in cui salvare le informazioni dei beacon

- - `setVisibleBeacon(beacons : PriorityQueue<MyBeacon>) : void`

Metodo che setta il campo dati lastBeaconsSeen

Argomenti:

- `beacons : PriorityQueue<MyBeacon>`
Lista dei beacon visibili

- + `startRecordingBeacons() : void`

Metodo che permette di avviare il log delle informazioni dei beacon visibili

3.4.4 model::MessageSendType



Figura 21: Classe MessageSendType

Nome: MessageSendType;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- - `VISIBLE_BEACON : String`

Rappresenta il messaggio che viene scambiato all'interno dell'applicazione

3.4.5 model::NavigationManager

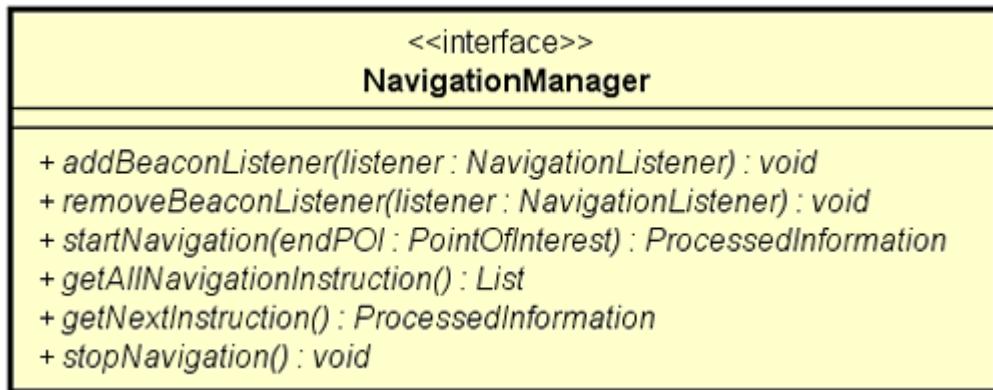


Figura 22: Interfaccia NavigationManager

Nome: *NavigationManager*;

Tipo: Interfaccia;

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

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

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

Argomenti:

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

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

Argomenti:

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

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

3.4.6 model::NavigationManagerImp

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

Figura 23: Classe NavigationManagerImp

Nome: NavigationManagerImp;

Tipo: Classe;

Estende:

- AbsBeaconReceiverManager.

Implementa:

- NavigationManager.

Visibilità: public;

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

Descrizione: Classe che si occupa della gestione della navigazione;

Attributi:

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

Metodi:

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

Argomenti:

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

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

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

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

Argomenti:

- **graph** : `MapGraph`
Grafo dell'edificio in cui si desidera navigare
 - **context** : `Context`
Contesto dell'applicazione
- + **onReceive()** : `void`
Metodo che si occupa di settare il campo dati `lastBeaconsSeen` con la `PriorityQueue<MyBeacon>` contenente gli ultimi beacon rilevati e di aggiornare tutti i listeners con le ultime istruzioni di navigazione
 - + **removeBeaconListener(listener : NavigationListener) : void**
Metodo che permette di rimuovere un listener

Argomenti:

- **listener** : `NavigationListener`
Listener che deve essere rimosso dalla lista di `NavigationListener`
- - **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**
Metodo che permette di fermare il rilevamento dei dati ottenuti dalla bussola
 - + **startNavigation() : ProcessedInformation**
Metodo che permette di avviare la navigazione verso uno specifico POI
 - + **stopCompass() : void**
Metodo che permette di attivare il rilevamento dei dati dalla bussola
 - + **stopNavigation() : void**
Metodo che permette di fermare la navigazione

3.4.7 model::NoBeaconSeenException

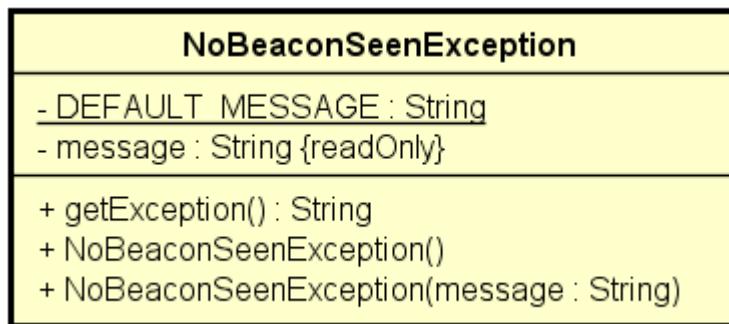


Figura 24: Classe NoBeaconSeenException

Nome: NoBeaconSeenException;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

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

Metodi:

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

Argomenti:

– message : String

Questo parametro richiede che un messaggio di tipo String

3.4.8 model::ServiceConnectionImp

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

Figura 25: 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

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**
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

3.4.9 model::beacon::BeaconManagerAdapter

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

Figura 26: Classe BeaconManagerAdapter

Nome: BeaconManagerAdapter;

Tipo: Classe;

Implementa:

- **BeaconRanger.**

Componenti delle librerie utilizzate:

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

Visibilità: public;**Utilizzo:** È utilizzata per rilevare i beacon;**Descrizione:** Classe che si occupa del rilevamento dei beacon. Estende la classe `android.app.Service` e implementa le interfacce `org.altbeacon.beacon.BeaconConsumer` e `org.altbeacon.beacon.startup.BootstrapNotifier`;**Attributi:**

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

Metodi:

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

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`

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`

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

Argomenti:

– `region : Region`

Criterio che serve ad eseguire il match con un beacon

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

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

Argomenti:

– `p : long`

Periodo di scansione

– `backOrFore : boolean`

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

– `betweenOrNot : boolean`

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

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

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`

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

- + `onDestroy() : void`

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

3.4.10 model::beacon::BeaconRanger

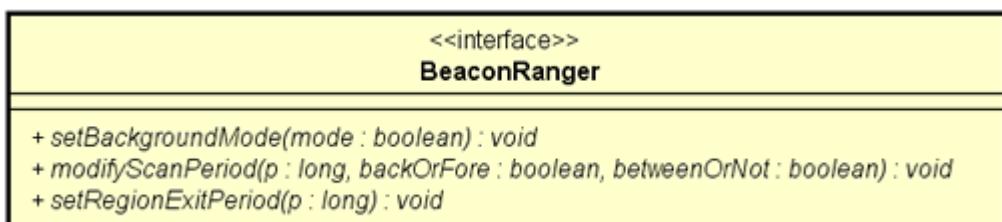


Figura 27: Interfaccia BeaconRanger

Nome: BeaconRanger;

Tipo: Interfaccia;

Visibilità: public;

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

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

Metodi:

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

3.4.11 model::beacon::LocalBinder



Figura 28: 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

3.4.12 model::beacon::Logger

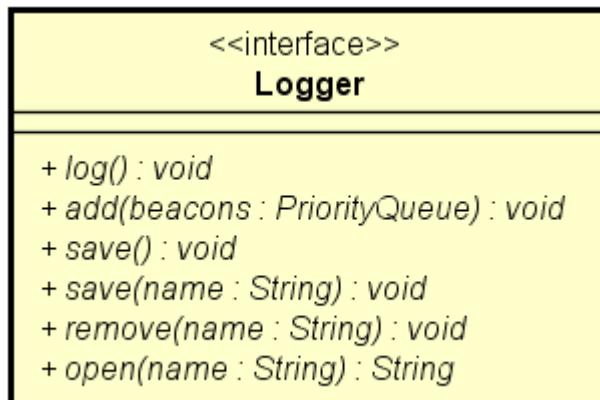


Figura 29: Interfaccia Logger

Nome: Logger;

Tipo: Interfaccia;

Visibilità: public;

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

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

Metodi:

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

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

Argomenti:

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

- + *log() : void*

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

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

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

Argomenti:

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

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

Metodo per la rimozione di un log precedentemente salvato

Argomenti:

- *name : String*
Nome del log da rimuovere

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

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

Argomenti:

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

3.4.13 model::beacon::LoggerImp

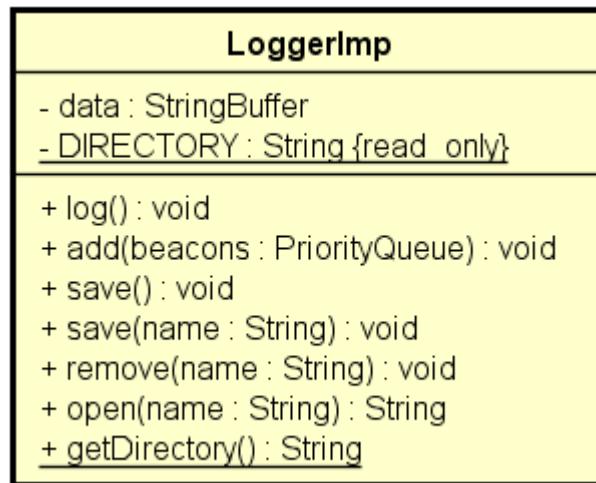


Figura 30: Classe LoggerImp

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**
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**

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

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

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

Argomenti:

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

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

Metodo per la rimozione di un log precedentemente salvato

Argomenti:

- **name : String**
Nome del log da rimuovere

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

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

3.4.14 model::beacon::MyBeacon

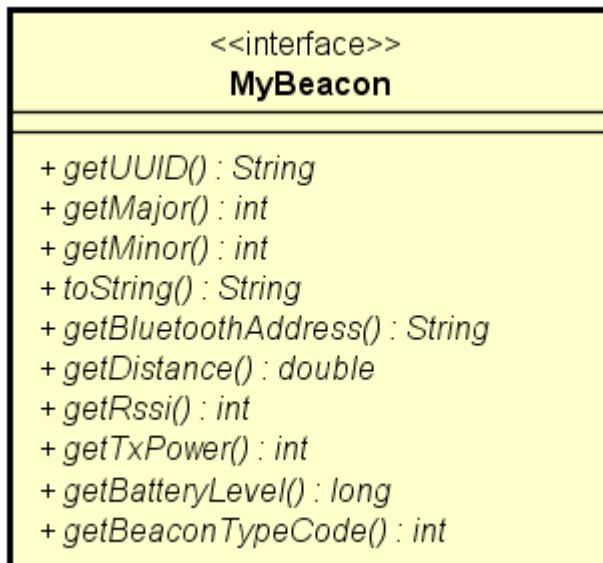


Figura 31: Interfaccia MyBeacon

Nome: MyBeacon;

Tipo: Interfaccia;

Visibilità: public;

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

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

Metodi:

- + *getBatteryLevel()* : long

Metodo che ritorna il livello di batteria del beacon rilevato

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

3.4.15 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 32: Classe MyBeaconImp

Nome: MyBeaconImp;

Tipo: Classe;

Implementa:

- MyBeacon.

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Attributi:

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

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
- + **MyBeaconImp(Beacon : Beacon)**
Costruttore della classe

Argomenti:

- **beacon : Beacon**
Questo parametro richiede un beacon
- + **recalculateDistance() : void**
Questo metodo permette di ricalcolare la distanza tra il beacon e il dispositivo
- + **toString() : String**
Questo metodo permette di fornire una conversione di MyBeaconImp a tipo String

3.4.16 model::beacon::MyDistanceCalculator

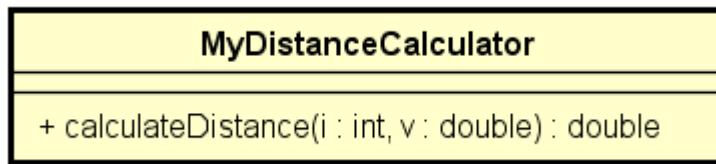


Figura 33: Classe MyDistanceCalculator

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'intefaccia org.altbeacon.beacon.distance.DistanceCaluclator.

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

3.4.17 model::compass::Compass

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

Figura 34: 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'accellerazione del device sui tre assi fisici
- - **lastAccelerometerSet** : boolean
variabile di guardia per accertarsi il reperimento di almeno un dato dall'accellerometro
- - **lastMagnetometerSet** : boolean
variabile di guardia per accertarsi il reperimento di almeno un dato dal magnetometro
- - **magnetometer** : Sensor
Sensore che misura il campo magnetico per i tre assi fisici
- - **orientation** : float[]
gradi di orientamento sui tre assi fisici
- - **rotationMatrix** : float[]
matrice di rotazione ottenuta dai dati rilevati dai sensori
- - **sensorManager** : SensorManager
oggetto fornito dal sistema Android per ottenere le istanze dei sensori

Metodi:

- + **Compass()**
Costruttore della classe Compass
- + **getLastCoordinate() : float**
Metodo che restituisce l'ultimo dato calcolato dai dati ricavati dai sensori che indica l'orientamento del dispositivo.
- + **onAccuracyChanged(sensor : Sensor, accuracy : int) : void**
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**
Metodo invocato ad ogni evento generato dai sensori attivi di Compass. Fornisce quindi i dati misurati dal sensore e elaborandoli ne ricava l'orientamento del device

Argomenti:

– event : SensorEvent

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

- + registerListener() : void

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

- + unregisterListener() : void

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

3.4.18 model::dataaccess::dao::BuildingContract

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

Figura 35: Classe BuildingContract

Nome: BuildingContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + BUILDING_TABLE_CREATE : String {readOnly}
Query per la creazione della tabella
- + COLUMN_ADDRESS : String {readOnly}
Valore della colonna address. Valore di default "address"
- + COLUMN_DESCRIPTION : String {readOnly}
Valore della colonna description. Valore di default "description"
- + COLUMN_ID : String {readOnly}
Valore della colonna id. Valore di default "id"
- + COLUMN_MAPVERSION : String {readOnly}
Valore della colonna mapVersion. Valore di default "mapVersion"
- + COLUMN_NAME : String {readOnly}
name
- + COLUMN_OPENINGHOURS : String {readOnly}
Valore della colonna openingHours. Valore di default "openingHours"
- + COLUMN_UUID : String {readOnly}
Valore della colonna uuid. Valore di default "uuid"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "Building"

3.4.19 model::dataaccess::dao::BuildingDao

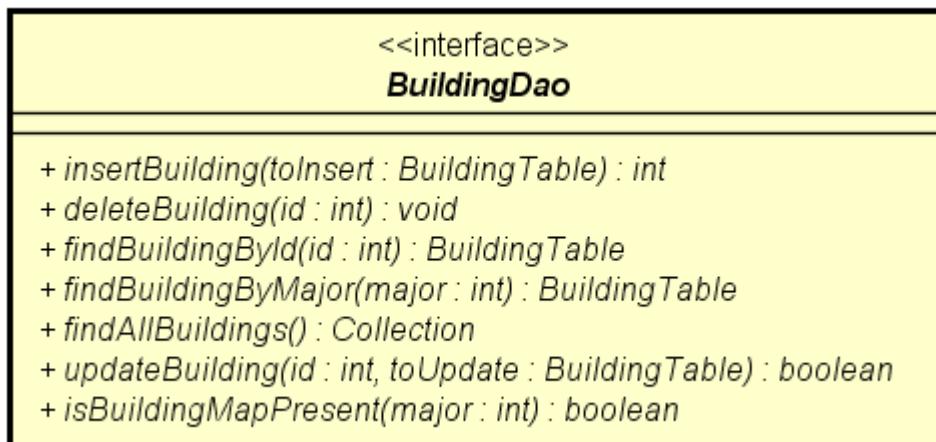


Figura 36: Interfaccia BuildingDao

Nome: BuildingDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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

3.4.20 model::dataaccess::dao::BuildingTable

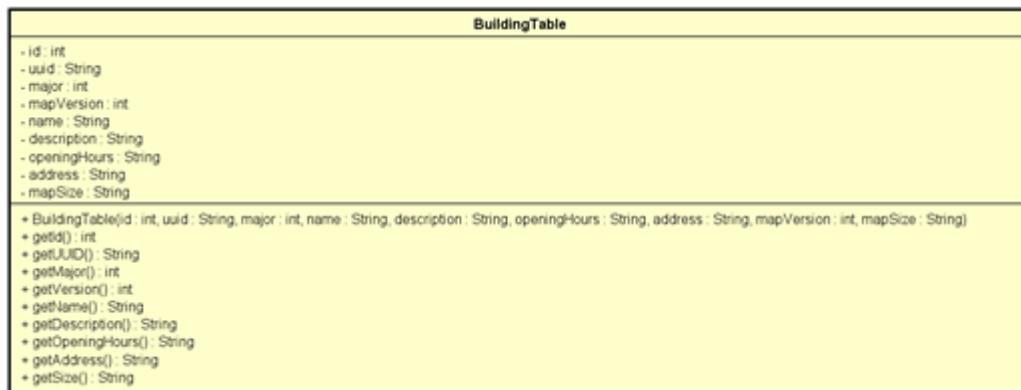


Figura 37: Classe BuildingTable

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

3.4.21 model::dataaccess::dao::CategoryContract

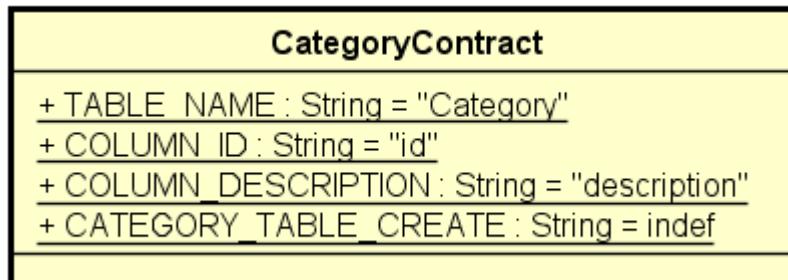


Figura 38: Classe CategoryContract

Nome: CategoryContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + CATEGORY_TABLE_CREATE : String {readOnly}
Query per la creazione della tabella
- + COLUMN_DESCRIPTION : String {readOnly}
Valore della colonna description. Valore di default "description"
- + COLUMN_ID : String {readOnly}
Valore della colonna id. Valore di default "id"
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "Category"

3.4.22 model::dataaccess::dao::CategoryDao

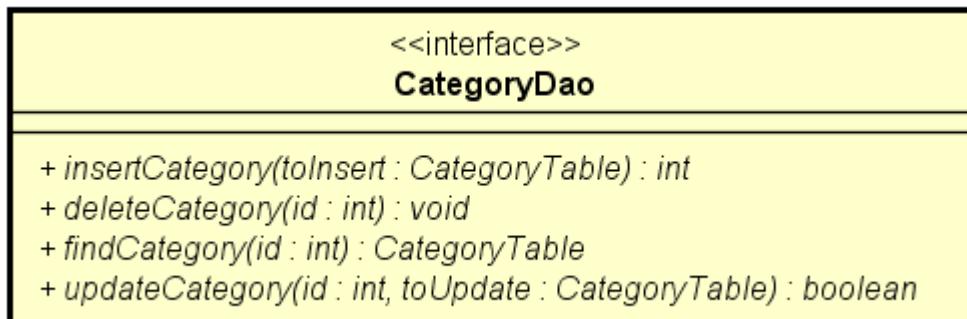


Figura 39: Interfaccia CategoryDao

Nome: CategoryDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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

3.4.23 model::dataaccess::dao::CategoryTable

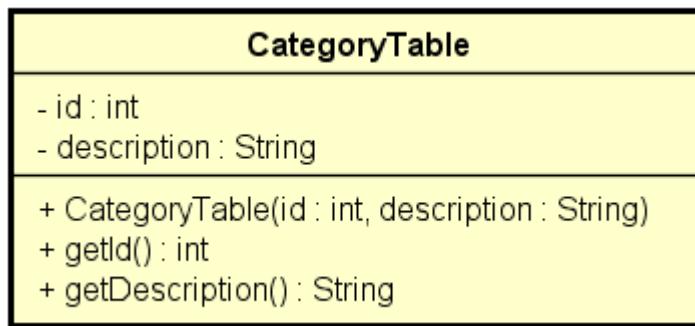


Figura 40: Classe CategoryTable

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

3.4.24 model::dataaccess::dao::CursorConverter

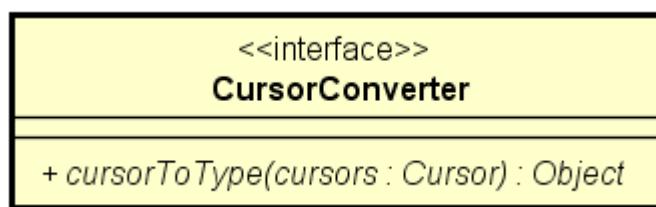


Figura 41: Interfaccia CursorConverter

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

3.4.25 model::dataaccess::dao::DaoFactoryHelper

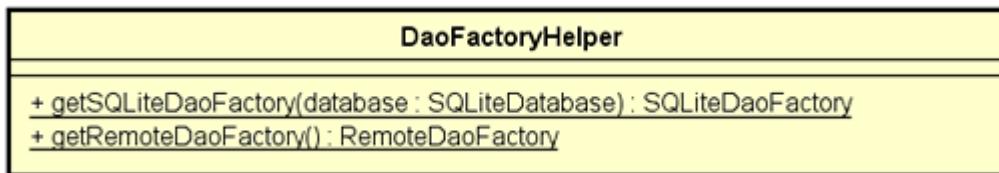


Figura 42: Classe DaoFactoryHelper

Nome: DaoFactoryHelper;

Tipo: Classe;

Visibilità: public;

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

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

Metodi:

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

Argomenti:

- database : SQLiteDatabase
Il database locale

3.4.26 model::dataaccess::dao::EdgeContract

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

Figura 43: Classe EdgeContract

Nome: EdgeContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + COLUMN_ACTION : String {readOnly}
Valore della colonna action. Valore di default "action"
- + COLUMN_COORDINATE : String {readOnly}
Valore della colonna coordinate. Valore di default "coordinate"
- + COLUMN_DISTANCE : String {readOnly}
Valore della colonna distance. Valore di default "distance"
- + COLUMN_ENDROI : String {readOnly}
Valore della colonna endROI. Valore di default "endROI"

- + COLUMN_ID : String {readOnly}
Valore della colonna id. Valore di default "id"
- + COLUMN_LONGDESCRIPTION : String {readOnly}
Valore della colonna longDescription. Valore di default "longDescription"
- + COLUMN_STARTROI : String {readOnly}
Valore della colonna startROI. Valore di default "startROI"
- + COLUMN_TYPEID : String {readOnly}
Valore della colonna typeId. Valore di default "typeId"
- + EDGE_TABLE_CREATE : String {readOnly}
Query per la creazione della tabella
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "Edge"

3.4.27 model::dataaccess::dao::EdgeDao

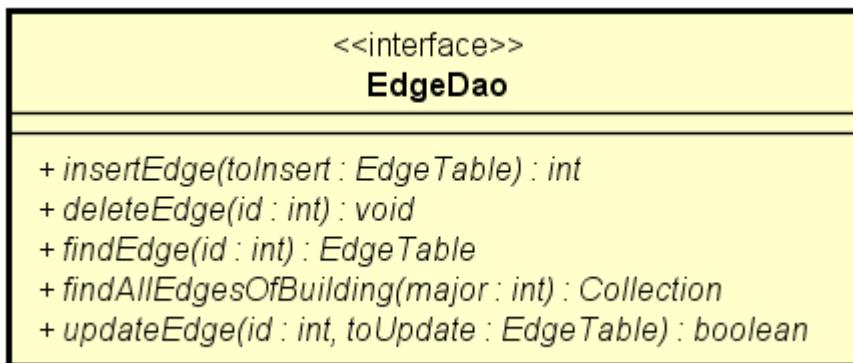


Figura 44: Interfaccia EdgeDao

Nome: EdgeDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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

3.4.28 model::dataaccess::dao::EdgeTable

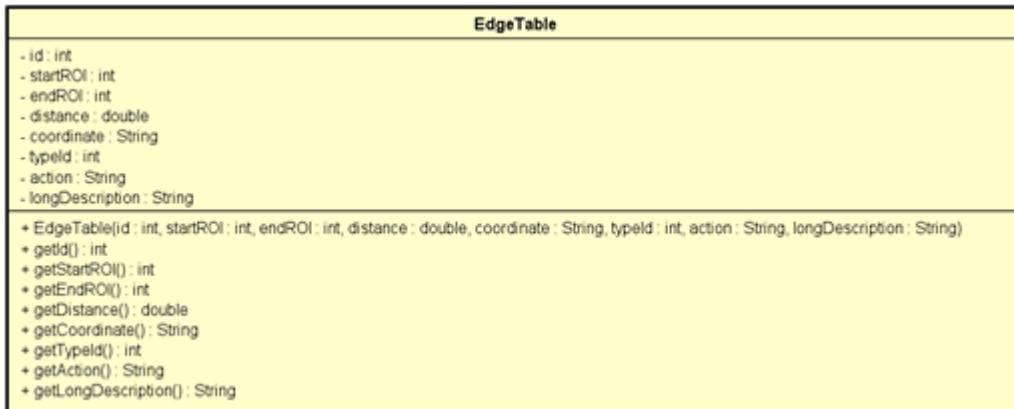


Figura 45: Classe EdgeTable

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

3.4.29 model::dataaccess::dao::EdgeTypeContract

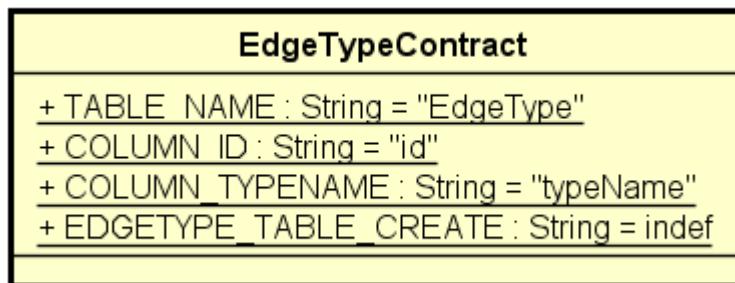


Figura 46: Classe EdgeTypeContract

Nome: EdgeTypeContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + COLUMN_ID : String {readOnly}
Valore della colonna id. Valore di default "id"
- + COLUMN_TYPENAME : String {readOnly}
Valore della colonna typeName. Valore di default "typeName"
- + EDGETYPE_TABLE_CREATE : String {readOnly}
Query per la creazione della tabella
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "EdgeType"

3.4.30 model::dataaccess::dao::EdgeTypeDao

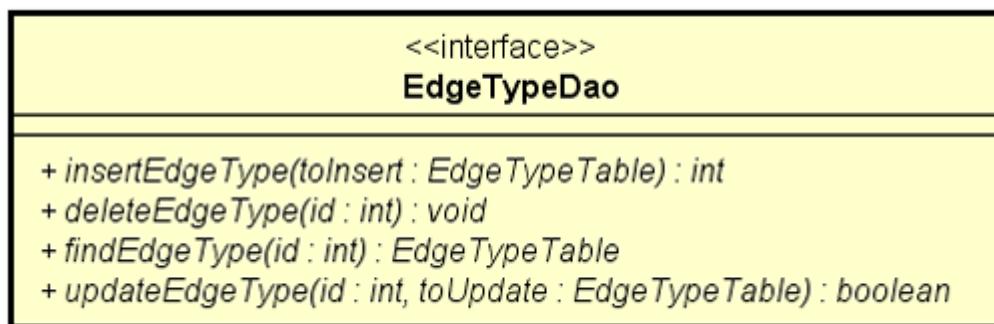


Figura 47: Interfaccia EdgeTypeDao

Nome: EdgeTypeDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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

3.4.31 model::dataaccess::dao::EdgeTypeTable

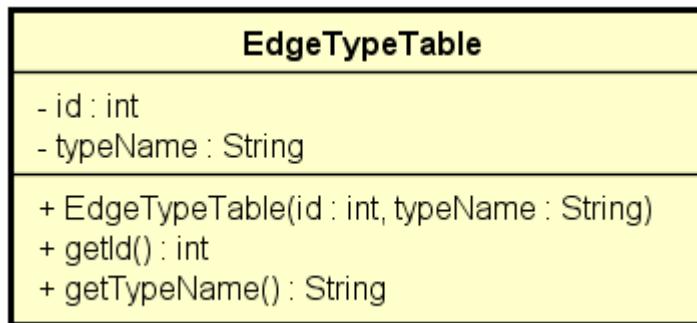


Figura 48: Classe EdgeTypeTable

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

3.4.32 model::dataaccess::dao::MapsDbContract



Figura 49: Classe MapsDbContract

Nome: MapsDbContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

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

3.4.33 model::dataaccess::dao::MapsDbHelper

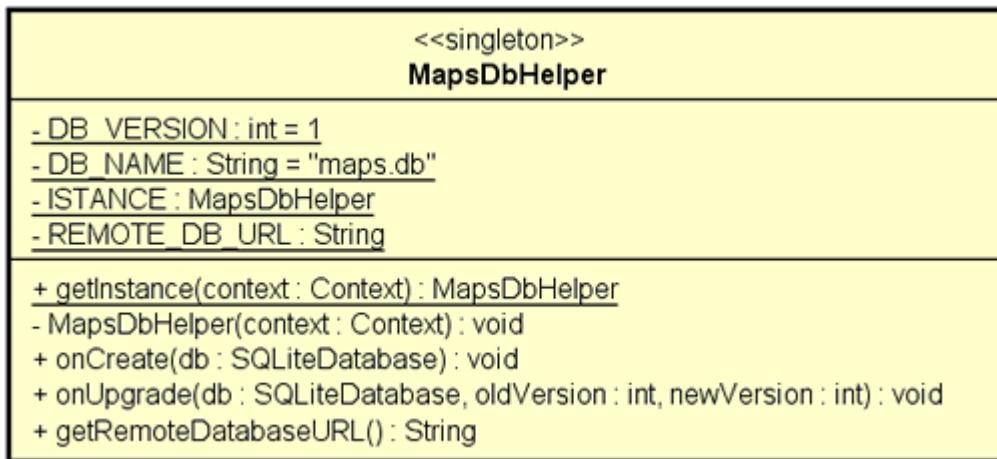


Figura 50: Classe MapsDbHelper

Nome: MapsDbHelper;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Attributi:

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

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

Metodi:

- + getInstance() : MapsDbHelper
Metodo che ritorna una istanza di MapsDbHelper
- + getRemoteDatabaseURL() : String
Metodo che ritorna l'URL del database remoto
- - MapsDbHelper()
Costruttore della classe MapsDbHelper
- + onCreate(db : SQLiteDatabase) : void
Metodo che viene chiamato la prima volta che viene creato il database

Argomenti:

- db : SQLiteDatabase
Riferimento al database
- + onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersione : int) : void
Metodo che viene chiamato per effettuare l'upgrade del database

Argomenti:

- db : SQLiteDatabase
Riferimento al database
- oldVersion : int
Numero di versione del vecchio database
- newVersione : int
Numero di versione del nuovo database

3.4.34 model::dataaccess::dao::PhotoContract

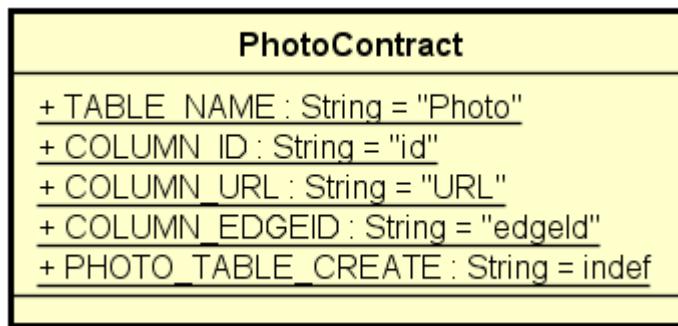


Figura 51: Classe PhotoContract

Nome: PhotoContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + COLUMN_EDGEID : String {readOnly}
Valore della colonna edgeId. Valore di default "edgeId"
- + COLUMN_ID : String {readOnly}
Valore della colonna id. Valore di default "id"
- + COLUMN_URL : String {readOnly}
Valore della colonna url. Valore di default "url"
- + PHOTO_TABLE_CREATE : String {readOnly}
Query per la creazione della tabella
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "Photo"

3.4.35 model::dataaccess::dao::PhotoDao

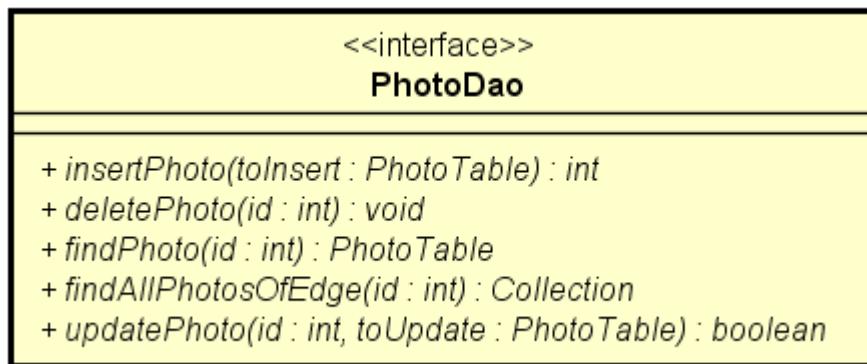


Figura 52: Interfaccia PhotoDao

Nome: PhotoDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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

3.4.36 model::dataaccess::dao::PhotoTable

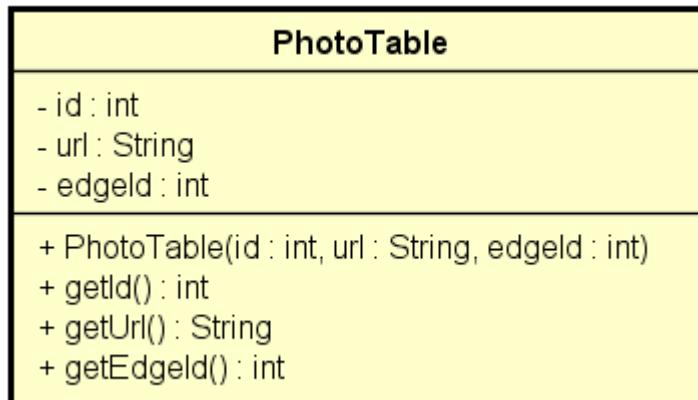


Figura 53: Classe PhotoTable

Nome: PhotoTable;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

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

Metodi:

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

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

Argomenti:

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

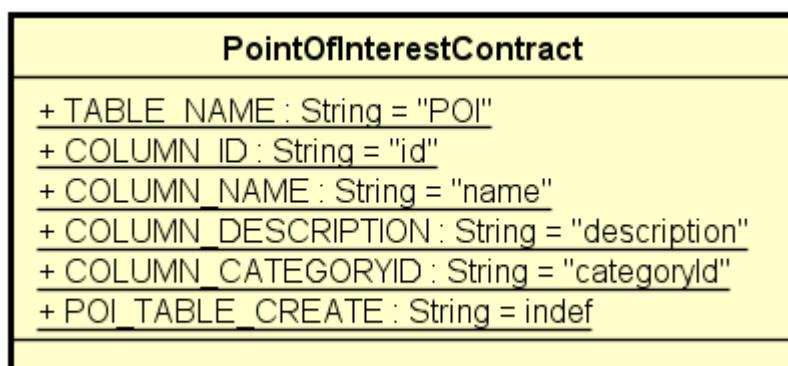
3.4.37 model::dataaccess::dao::PointOfInterestContract

Figura 54: Classe PointOfInterestContract

Nome: PointOfInterestContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + COLUMN_CATEGORYID : String {readOnly}
Valore della colonna categoryId. Valore di default "categoryId"
- + COLUMN_DESCRIPTION : String {readOnly}
Valore della colonna description. Valore di default "description"
- + COLUMN_ID : String {readOnly}
Valore della colonna id. Valore di default "id"
- + COLUMN_NAME : String {readOnly}
Valore della colonna name. Valore di default "name"
- + POI_TABLE_CREATE : String {readOnly}
Contiene la query di creazione della tabella
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "POI"

3.4.38 model::dataaccess::dao::PointOfInterestDao

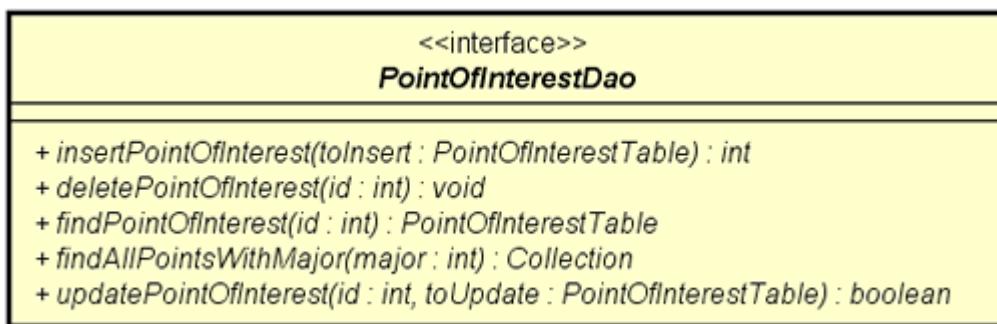


Figura 55: Interfaccia PointOfInterestDao

Nome: PointOfInterestDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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

3.4.39 model::dataaccess::dao::PointOfInterestTable

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

Figura 56: Classe PointOfInterestTable**Nome:** PointOfInterestTable;**Tipo:** Classe;**Visibilità:** public;**Utilizzo:** Viene utilizzata per recuperare le informazioni di una ennupla della tabella PointOfInterest del database locale ;**Descrizione:** Classe che rappresenta una ennupla della tabella PointOfInterest del database locale;**Attributi:**

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

- - `name : String`

Nome del POI

Metodi:

- + `getCategoryId() : int`

Metodo che ritorna l'identificativo del POI

- + `getDescription() : String`

Metodo che ritorna la descrizione del POI

- + `getId() : int`

Metodo che ritorna l'identificativo del POI

- + `getName() : String`

Metodo che ritorna il nome dell'edificio

- + `PointOfInterestTable(description : String, id : int,`

`name : String, category : int)`

Costruttore della classe PointOfInterestTable

Argomenti:

- `description : String`

Descrizione del POI

- `id : int`

Identificativo del POI

- `name : String`

Nome del POI

- `category : int`

Identificativo della categoria a cui appartiene il POI

3.4.40 model::dataaccess::dao::RegionOfInterestContract



Figura 57: Classe RegionOfInterestContract

Nome: RegionOfInterestContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + COLUMN_ID : String {readOnly}
Valore della colonna id. Valore di default "id"
- + COLUMN_MAJOR : String {readOnly}
Valore della colonna major. Valore di default "major"
- + COLUMN_MINOR : String {readOnly}
Valore della colonna minor. Valore di default "minor"
- + COLUMN_UUID : String {readOnly}
Valore della colonna uuid. Valore di default "uuid"
- + ROI_TABLE_CREATE : String {readOnly}
Contiene la query di creazione della tabella
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "ROI"

3.4.41 model::dataaccess::dao::RegionOfInterestDao



Figura 58: Interfaccia RegionOfInterestDao

Nome: RegionOfInterestDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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<RegionOfInterestT`

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

3.4.42 model::dataaccess::dao::RegionOfInterestTable



Figura 59: Classe RegionOfInterestTable

Nome: RegionOfInterestTable;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

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

Metodi:

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

Argomenti:

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

3.4.43 model::dataaccess::dao::RemoteBuildingDao

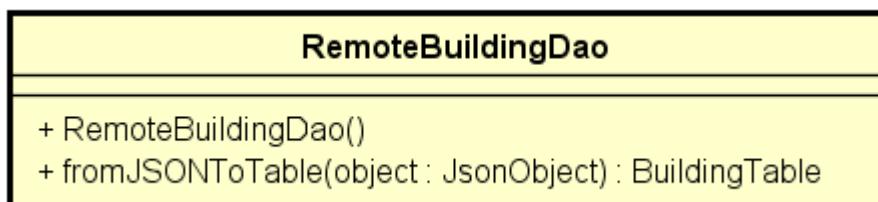


Figura 60: Classe RemoteBuildingDao

Nome: RemoteBuildingDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

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

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

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

3.4.44 model::dataaccess::dao::RemoteCategoryDao

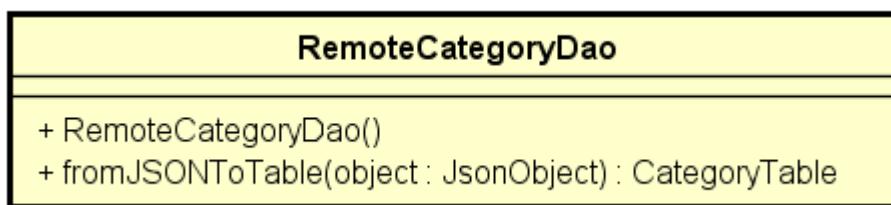


Figura 61: Classe RemoteCategoryDao

Nome: RemoteCategoryDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

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

3.4.45 model::dataaccess::dao::RemoteDaoFactory

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

Figura 62: Classe RemoteDaoFactory

Nome: RemoteDaoFactory;

Tipo: Classe;

Visibilità: public;

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

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

Metodi:

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

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

3.4.46 model::dataaccess::dao::RemoteEdgeDao

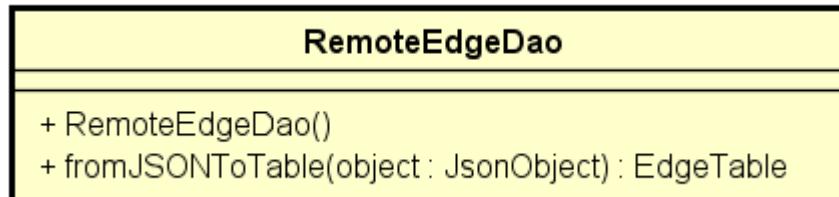


Figura 63: Classe `RemoteEdgeDao`

Nome: `RemoteEdgeDao`;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

- com.google.gson.JsonObject.

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

- + RemoteEdgeDao()

Costruttore di default per la classe RemoteEdgeDao

3.4.47 model::dataaccess::dao::RemoteEdgeTypeDao

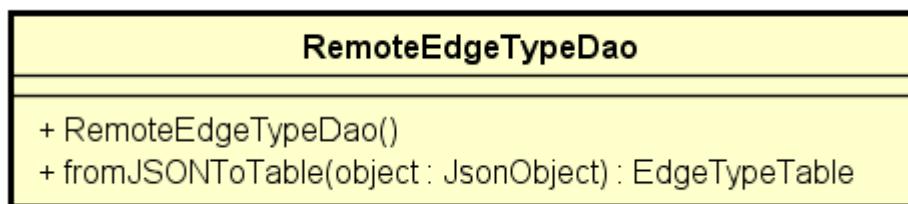


Figura 64: Classe RemoteEdgeTypeDao

Nome: RemoteEdgeTypeDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

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

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

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

3.4.48 model::dataaccess::dao::RemotePhotoDao

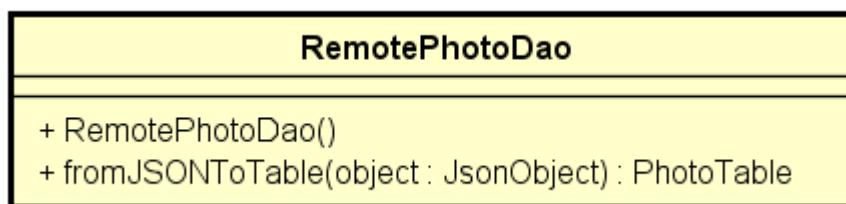


Figura 65: Classe RemotePhotoDao

Nome: RemotePhotoDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

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

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

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

3.4.49 model::dataaccess::dao::RemotePointOfInterestDao

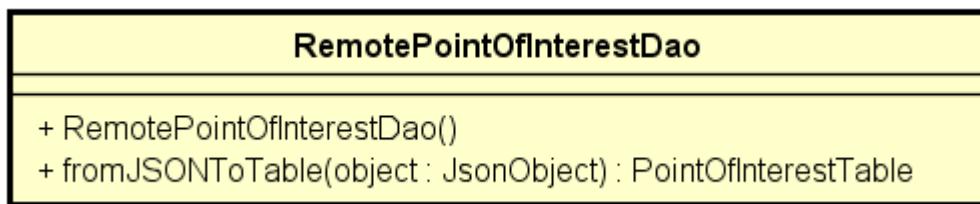


Figura 66: Classe RemotePointOfInterestDao

Nome: RemotePointOfInterestDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

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

3.4.50 model::dataaccess::dao::RemoteRegionOfInterestDao



Figura 67: Classe RemoteRegionOfInterestDao

Nome: RemoteRegionOfInterestDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

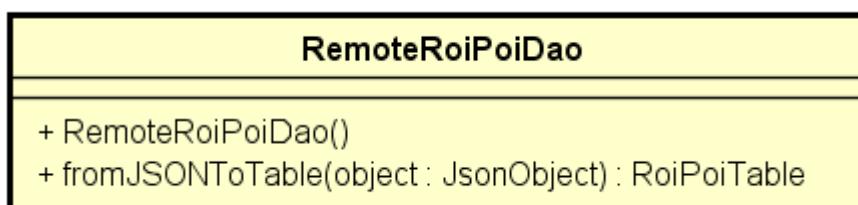
Visibilità: public;**Utilizzo:** È utilizzata per la conversione di oggetti JSON in oggetti persistenti RegionOfInterestTable che rappresentano la tabella "ROI" del database locale;**Descrizione:** Classe di utility per la conversione da JSON a RegionOfInterestTable;**Metodi:**

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

Argomenti:

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

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

3.4.51 model::dataaccess::dao::RemoteRoiPoiDao**Figura 68:** Classe RemoteRoiPoiDao**Nome:** RemoteRoiPoiDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Metodi:

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

Argomenti:

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

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

3.4.52 model::dataaccess::dao::RoiPoiContract

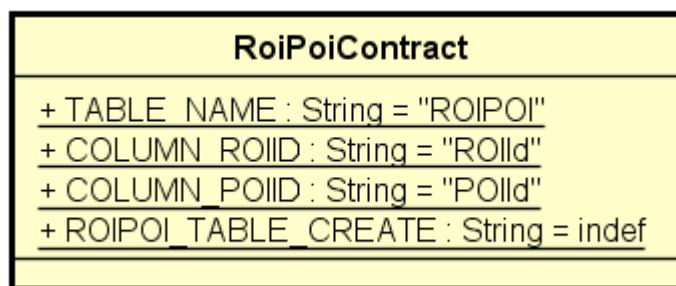


Figura 69: Classe RoiPoiContract

Nome: RoiPoiContract;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- + COLUMN_POIID : String {readOnly}
Valore della colonna poiId. Valore di default "poiId"
- + COLUMN_ROIID : String {readOnly}
Valore della colonna roiId. Valore di default "roiId"
- + ROIPOI_TABLE_CREATE : String {readOnly}
Query per la creazione della tabella
- + TABLE_NAME : String {readOnly}
Nome della tabella. Valore di default "ROIPOI"

3.4.53 model::dataaccess::dao::RoiPoiDao

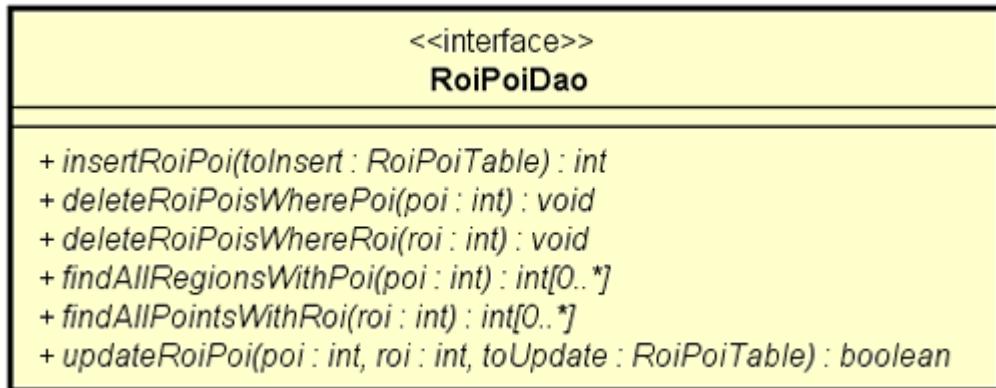


Figura 70: Interfaccia RoiPoiDao

Nome: RoiPoiDao;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: Viene utilizzata per rendere indipendenti l'invocazioni 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:

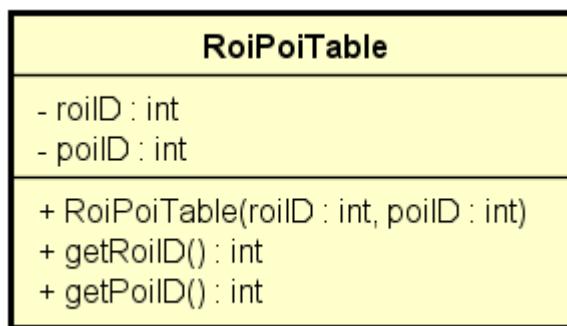
- **poi : int**
Identificativo del POI di cui recuperare gli identificativi di tutti i ROI associati
- + **insertRoiPoi(toInsert : RoiPoiTable) : int**
Metodo che permette l'inserimento tra ROI ed POI nel database locale utilizzando un oggetto RoiPoiTable

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

3.4.54 model::dataaccess::dao::RoiPoiTable**Figura 71:** Classe RoiPoiTable**Nome:** RoiPoiTable;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

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

Metodi:

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

Argomenti:

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

3.4.55 model::dataaccess::dao::SQLDao

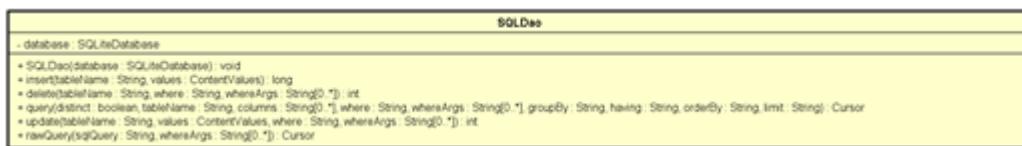


Figura 72: Classe SQLDao

Nome: SQLDao;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

Descrizione: Classe che contiene le operazioni di query dirette;

Attributi:

- - database : SQLiteDatabase {readOnly}
Database locale

Metodi:

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

Argomenti:

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

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

Argomenti:

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

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

Metodo per effettuare una query sul database locale

Argomenti:

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

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

Metodo per eseguire una query fornendola sotto forma di stringa

Argomenti:

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

- + SQLDao(database : SQLiteDatabase)

Costruttore della classe SQLDao

Argomenti:

- database : SQLiteDatabase

Database locale dell'applicazione

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

: String, whereArgs : String[]) : int

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

Argomenti:

- tableName : String

Nome della tabella su cui eseguire l'operazione

- values : ContentValues

Valori aggiornati che sostituiranno i presenti

- where : String

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

- whereArgs : String[]

Valori delle condizioni where

3.4.56 model::dataaccess::dao::SQLiteBuildingDao

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

Figura 73: Classe SQLiteBuildingDao

Nome: SQLiteBuildingDao;

Tipo: Classe;

Implementa:

- BuildingDao.

Visibilità: public;

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

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

Metodi:

- + cursorToType(cursor : Cursor) : BuildingTable

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

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 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`

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`

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

Argomenti:

– `toInsert : BuildingTable`

Oggetto di tipo BuildingTable che contiene le informazioni dell'edificio

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

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

Argomenti:

– `major : int`

major dell'edificio

- + `SQLiteBuildingDao(database : SQLiteDatabase)`

Costruttore della classe SQLiteBuildingDao

Argomenti:

– `database : SQLiteDatabase`

Il database locale

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

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

3.4.57 model::dataaccess::dao::SQLiteCategoryDao

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

Figura 74: Classe SQLiteCategoryDao

Nome: SQLiteCategoryDao;

Tipo: Classe;

Implementa:

- CategoryDao.

Visibilità: public;

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

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

Metodi:

- + cursorToType(cursor : Cursor) : CategoryTable

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`

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

- + `SQLiteCategoryDao(database : SQLiteDatabase)`

Costruttore della classe SQLiteCategoryDao

Argomenti:

- `database : SQLiteDatabase`

Il database locale

- + `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

3.4.58 model::dataaccess::dao::SQLiteDaoFactory

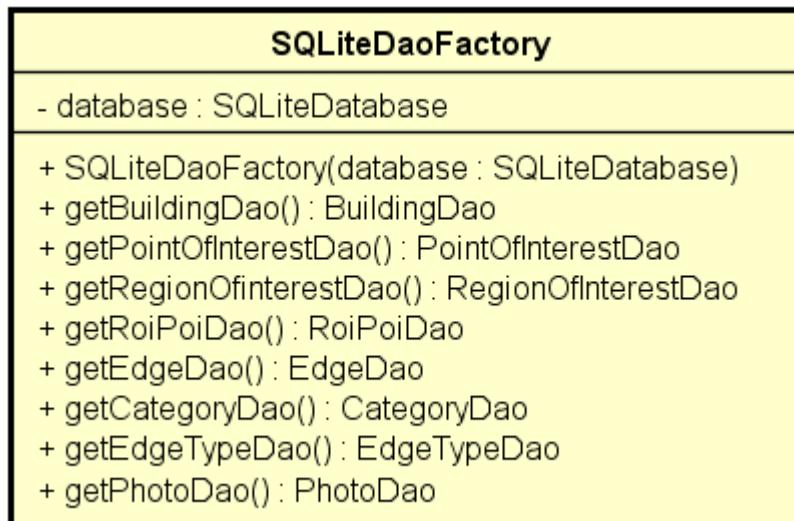


Figura 75: Classe SQLiteDaoFactory

Nome: SQLiteDaoFactory;

Tipo: Classe;

Visibilità: public;

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

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

Attributi:

- - database : SQLiteDatabase {readOnly}
Il database locale

Metodi:

- + getBuildingDao() : BuildingDao
Metodo che viene utilizzato per ottenere un'istanza di SQLiteBuildingDao
- + getCategoryDao() : CategoryDao
Metodo che viene utilizzato per ottenere un'istanza di SQLiteCategoryDao

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

Argomenti:

- `database` : `SQLiteDatabase`
Il database locale

3.4.59 model::dataaccess::dao::SQLiteEdgeDao

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

Figura 76: Classe SQLiteEdgeDao

Nome: SQLiteEdgeDao;

Tipo: Classe;

Implementa:

- EdgeDao.

Visibilità: public;

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

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

Metodi:

- + cursorToType(cursor : Cursor) : EdgeTable

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`

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

- + `SQLiteEdgeDao(database : SQLiteDatabase)`

Costruttore della classe SQLiteEdgeDao

Argomenti:

- `database : SQLiteDatabase`

Il database locale

- + `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

3.4.60 model::dataaccess::dao::SQLiteEdgeTypeDao

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

Figura 77: Classe SQLiteEdgeTypeDao

Nome: SQLiteEdgeTypeDao;

Tipo: Classe;

Implementa:

- `EdgeTypeDao`.

Visibilità: public;

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

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

Metodi:

- `+ cursorToType(cursor : Cursor) : EdgeTypeTable`
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

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

- + SQLiteEdgeTypeDao(database : SQLiteDatabase)

Costruttore della classe SQLiteEdgeTypeDao

Argomenti:

– database : SQLiteDatabase

Il database locale

- + 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

3.4.61 model::dataaccess::dao::SQLitePhotoDao

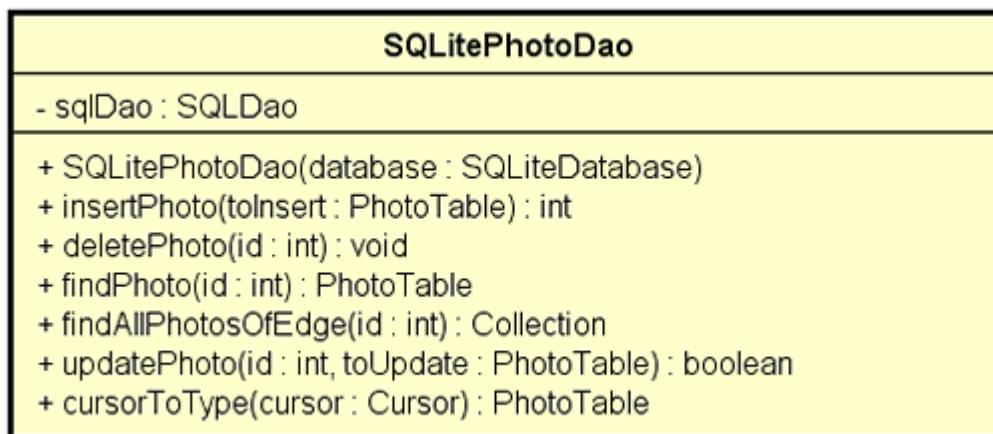


Figura 78: Classe SQLitePhotoDao

Nome: SQLitePhotoDao;

Tipo: Classe;

Implementa:

- PhotoDao.

Visibilità: public;

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

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

Metodi:

- + `cursorToType(cursor : Cursor) : PhotoTable`

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`

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(toInsert : PhotoTable) : int`

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

Argomenti:

- `toInsert : PhotoTable`

Oggetto di tipo Photo che contiene le informazioni della foto

- + `SQLitePhotoDao(database : SQLiteDatabase)`

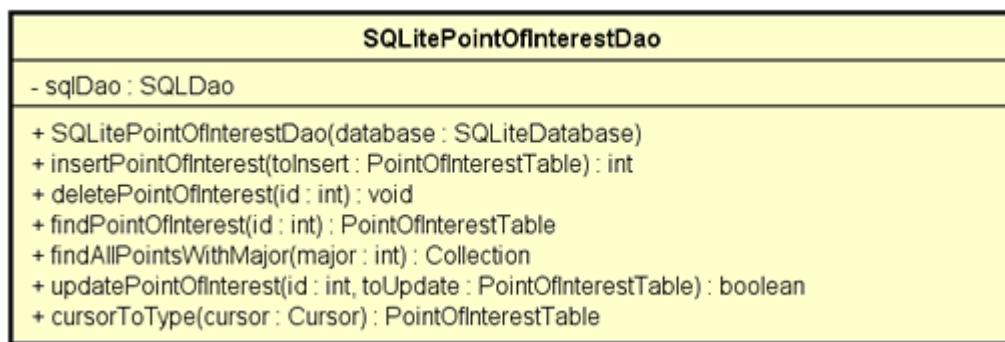
Costruttore della classe SQLitePhotoDao

Argomenti:

- database : SQLiteDatabase
Il database locale
- + updatePhoto(id : int, toUpdate : PhotoTable) : boolean
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

3.4.62 model::dataaccess::dao::SQLitePointOfInterestDao**Figura 79:** Classe SQLitePointOfInterestDao**Nome:** SQLitePointOfInterestDao;**Tipo:** Classe;**Implementa:**

- PointOfInterestDao.

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

Metodi:

- + **cursorToType(cursor : Cursor) : PointOfInterestTable**
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**

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

Argomenti:

- `database` : `SQLiteDatabase`
Il database locale
- + `updatePointOfInterest(id : int, toUpdate : PointOfInterestTable) : boolean`
Metodo per aggiornare le informazioni di un edificio nella tabella "POI" del database locale
- + `updatePointOfInterest(id : int, toUpdate : PointOfInterestTable) : void`
Oggetto che contiene le informazioni aggiornate del POI

3.4.63 model::dataaccess::dao::SQLiteRegionOfInterestDao

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

Figura 80: Classe `SQLiteRegionOfInterestDao`**Nome:** `SQLiteRegionOfInterestDao`;**Tipo:** Classe;**Implementa:**

- `RegionOfInterestDao`.

Visibilità: public;

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

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

Metodi:

- + cursorToType(cursor : Cursor) : RegionOfInterestTable
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

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

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

Argomenti:

- `database : SQLiteDatabase`

Il database locale

- + `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

3.4.64 model::dataaccess::dao::SQLiteRoiPoiDao

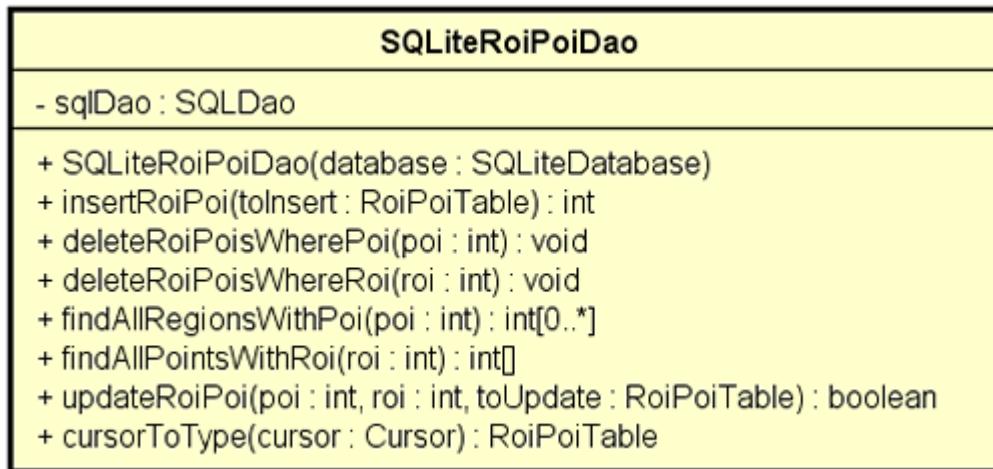


Figura 81: Classe SQLiteRoiPoiDao

Nome: SQLiteRoiPoiDao;

Tipo: Classe;

Implementa:

- RoiPoiDao.

Visibilità: public;

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

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

Metodi:

- + cursorToType(cursor : Cursor) : RoiPoiTable

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**
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:

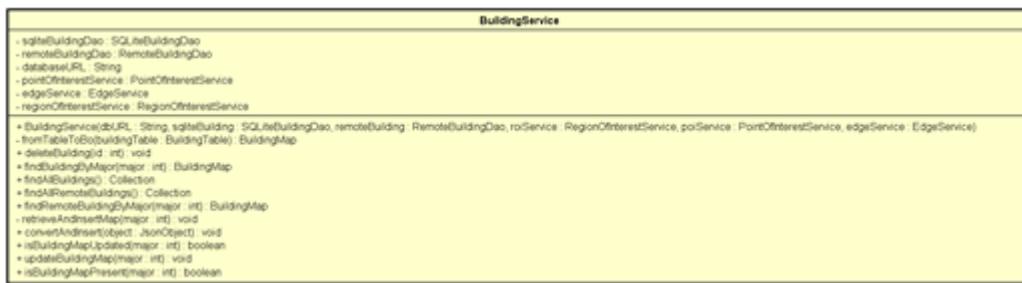
- **toInsert** : RoiPoiTable
Oggetto di tipo RoiPoiTable che contiene le associazioni tra ROI e POI
- + **SQLiteRoiPoiDao(database : SQLiteDatabase)**
Costruttore della classe SQLiteRoiPoiDao

Argomenti:

- **database** : SQLiteDatabase
Il database locale
- + **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

3.4.65 model::dataaccess::service::BuildingService**Figura 82:** Classe BuildingService**Nome:** BuildingService;**Tipo:** Classe;**Implementa:**

- DatabaseService.

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Attributi:

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

Metodi:

- + BuildingService(dbURL : String, sqliteBuilding : SQLiteDao, remoteBuilding : RemoteBuildingDao, roiService : RegionOfInterestService, poiService : PointOfInterestService, edgeService : EdgeService)
Costruttore della classe BuildingService

Argomenti:

- **dbURL** : `String`
URL del database remoto
- **sqliteBuilding** : `SQLiteDao`
Oggetto che rappresenta un DAO per la tabella "Building" del database locale
- **remoteBuilding** : `RemoteBuildingDao`
Oggetto di utility per la conversione da JSON a Building-Table
- **roiService** : `RegionOfInterestService`
Oggetto che si pone come layer Service tra gli oggetti RegionOfInterest e gli oggetti DAO corrispettivi
- **poiService** : `PointOfInterestService`
Oggetto che si pone come layer Service tra gli oggetti PointOfInterest e gli oggetti DAO corrispettivi
- **edgeService** : `EdgeService`
Oggetto che si pone come layer Service tra gli oggetti EnrichedEdge e gli oggetti DAO corrispettivi
- + **convertAndInsert(object : JsonObject) : void**
Metodo per la conversione di un JsonObject in un oggetto BuildingTable, che verrà inserito nel database locale

Argomenti:

- **object** : `JsonObject`
Oggetto JsonObject che contiene le informazioni di un edificio
- + **deleteBuilding(id : int) : void**
Metodo per rimuovere la mappa di un edificio dal database locale

Argomenti:

- **id** : `int`
Identificativo dell'edificio da rimuovere
- + **findAllBuildings() : Collection<BuildingTable>**
Metodo per recuperare le informazioni di tutte le mappe degli edifici presenti nel database locale
- + **findAllRemoteBuildings() : Collection<BuildingTable>**
Metodo per recuperare le informazioni di tutte le mappe degli edifici presenti nel database remoto

- + **findBuildingByMajor(major : int) : BuildingMap**
Metodo per recuperare la mappa di un edificio ricercandola nel database locale

Argomenti:

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

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

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

Argomenti:

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

- + `updateBuildingMap(major : int) : void`
Metodo per aggiornare la mappa di un edificio all'ultima versione disponibile

Argomenti:

- `major : int`
Major dell'edificio

3.4.66 model::dataaccess::service::DatabaseService

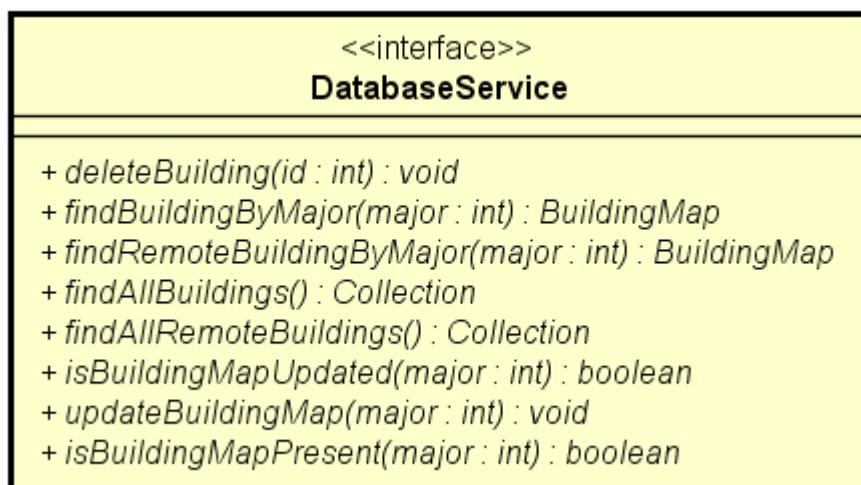


Figura 83: Interfaccia DatabaseService

Nome: DatabaseService;

Tipo: Interfaccia;

Visibilità: public;

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

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

Metodi:

- + `deleteBuilding(id : int) : void`
Metodo per cancellare una mappa a partire dall'identificativo di un edificio

Argomenti:

– `id : int`

Identificativo numerico di un oggetto BuildingMap

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

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

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

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

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

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

Argomenti:

– `major : int`

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

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

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

Argomenti:

– `major : int`

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

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

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

Argomenti:

– `major : int`

Major dell'edificio

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

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

Argomenti:

– `major : int`

Major dell'edificio

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

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

Argomenti:

- *major : int*
Major dell'edificio

3.4.67 model::dataaccess::service::EdgeService



Figura 84: Classe EdgeService

Nome: EdgeService;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: Viene utilizzata come layer Service tra gli oggetti EnrichedEdge e gli oggetti DAO 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 corrispondenti
- - **sqliteEdgeDao** : `SQLiteEdgeDao`
Oggetto che rappresenta un DAO per la tabella "Edge" del database locale
- - **sqliteEdgeTypeDao** : `SQLiteEdgeTypeDao`
Oggetto che rappresenta un DAO per la tabella "EdgeType" del database locale

Metodi:

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

Argomenti:

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

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

Argomenti:

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

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

Argomenti:

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

- + **EdgeService(sqliteEdge : SQLiteEdgeDao, remoteEdge : RemoteEdgeDao, sqliteEdgeType : SQLiteEdgeTypeDao, remoteEdgeType : RemoteEdgeTypeDao, photoService : PhotoService, roiService**

: RegionOfInterestService)
Costruttore della classe EdgeService

Argomenti:

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

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

Argomenti:

- **major** : int
Major dell'edificio
- + **findEdge(id : int)** : EnrichedEdge
Metodo per recuperare un Edge ricercandolo nel database locale

Argomenti:

- **id** : int
Identificativo numerico dell'Edge da ricercare
- - **fromTableToBo(edgeTable : EdgeTable)** : EnrichedEdge
Metodo per la costruzione di oggetto EnrichedEdge a partire da un EdgeTable

Argomenti:

- **edgeTable** : EdgeTable
Oggetto contenente le informazioni di un Edge

3.4.68 model::dataaccess::service::PhotoService

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

Figura 85: Classe PhotoService

Nome: PhotoService;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

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

Attributi:

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

Metodi:

- + **convertAndInsert(object : JsonObject) : void**

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

Argomenti:

- object : JsonObject

Oggetto JsonObject che contiene le informazioni di una foto

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

Metodo per rimuovere una foto di un Edge dal database locale

Argomenti:

- id : int

Identificativo numerico della foto da rimuovere

- + **findAllPhotosOfEdge(id : int) : Collection<PhotoRef>**

Metodo per recuperare tutte le foto di un Edge dal database locale

Argomenti:

- id : int

Identificativo dell'Edge di cui si vuole recuperare tutte le foto

- + **findPhoto(id : int) : PhotoRef**

Metodo per recuperare una foto ricercandola nel database locale

Argomenti:

- id : int

Identificativo numerico della foto da recuperare

- - **fromTableToBo(photoTable : PhotoTable) : PhotoRef**

Metodo per la costruzione di oggetto PhotoRef a partire da un PhotoTable

Argomenti:

- photoTable : PhotoTable

Oggetto contenente le informazioni della foto

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

Costruttore della classe PhotoService

Argomenti:

- sqlitePhoto : SQLitePhotoDao

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

- **remotePhoto** : `RemotePhotoDao`
Oggetto di utility per la conversione da JSON a PhotoTable

3.4.69 model::dataaccess::service::PointOfInterestService



Figura 86: Classe PointOfInterestService

Nome: `PointOfInterestService`;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: `public`;

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

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

Attributi:

- - `remoteCategoryDao` : `RemoteCategoryDao`
Oggetto di utility per la conversione da JSON a CategoryTable
- - `remotePointOfInterestDao` : `RemotePointOfInterestDao`
Oggetto di utility per la conversione da JSON a PointOfInterestTable

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

Metodi:

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

Argomenti:

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

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

Argomenti:

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

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

Argomenti:

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

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

Argomenti:

– `id : int`

Identificativo numerico del PointOfInterest da rimuovere

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

Argomenti:

– `major : int`

Major dell'edificio

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

Argomenti:

– `id : int`

Identificativo numerico del PointOfInterest da recuperare

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

Argomenti:

– `pointOfInterestTable : PointOfInterestTable`

Oggetto contenente le informazioni del PointOfInterest

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

Argomenti:

– `sqlitePOI : SQLitePointOfInterestDao`

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

– `remotePOI : RemotePointOfInterestDao`

Oggetto di utility per la conversione da JSON a PointOfInterestTable

– `sqliteRoiPoi : SQLiteRoiPoiDao`

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

- **remoteRoiPoi** : `RemoteRoiPoiDao`
Oggetto di utility per la conversione da JSON a 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

3.4.70 model::dataaccess::service::RegionOfInterestService



Figura 87: Classe RegionOfInterestService

Nome: RegionOfInterestService;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

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

Metodi:

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

Argomenti:

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

- + `deleteRegionOfInterest(id : int) : void`
Metodo per rimuovere una RegionOfInterest dal database locale

Argomenti:

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

- + `findAllRegionsWithMajor(major : int) : Collection<RegionOfInterest>`
Metodo per recuperare le informazioni di tutte le RegionOfInterest di un edificio, dato il major dell'edificio

Argomenti:

- `major : int`
Major dell'edificio

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

Argomenti:

- **id : int**
Identificativo numerico della RegionOfInterest da recuperare
- • - **fromTableToBo(regionOfInterestTable : RegionOfInterestTable) : RegionOfInterest**
Metodo per la costruzione di oggetto RegionOfInterest a partire da un RegionOfInterestTable

Argomenti:

- **regionOfInterestTable : RegionOfInterestTable**
Oggetto contenente le informazioni della RegionOfInterest
- • + **RegionOfInterestService(sqliteROI : SQLiteRegionOfInterestDao, remoteROI : RemoteRegionOfInterestDao, sqliteRoiPoi : SQLiteRoiPoiDao, remoteRoiPoi : RemoteRoiPoiDao)**
Costruttore della classe RegionOfInterestService

Argomenti:

- **sqliteROI : SQLiteRegionOfInterestDao**
Oggetto che rappresenta un DAO per la tabella "ROI" del database locale
- **remoteROI : RemoteRegionOfInterestDao**
Oggetto di utility per la conversione da JSON a RegionOfInterestTable
- **sqliteRoiPoi : SQLiteRoiPoiDao**
Oggetto che rappresenta un DAO per la tabella "ROI-POI" del database locale
- **remoteRoiPoi : RemoteRoiPoiDao**
Oggetto di utility per la conversione da JSON a RoiPoiTable

3.4.71 model::dataaccess::service::ServiceHelper**Figura 88:** 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 :`

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

3.4.72 model::navigator::BuildingInformation

BuildingInformation
<ul style="list-style-type: none"> - <code>name : String {readOnly}</code> - <code>description : String {readOnly}</code> - <code>openingHours : String {readOnly}</code> - <code>address : String {readOnly}</code>
<ul style="list-style-type: none"> + <code>BuildingInformation(name : String, description : String, timetable : String, address : String)</code> + <code>getName() : String</code> + <code>getDescription() : String</code> + <code>getOpeningHours() : String</code> + <code>getAddress() : String</code> + <code>toString() : String</code>

Figura 89: 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

3.4.73 model::navigator::BuildingMap

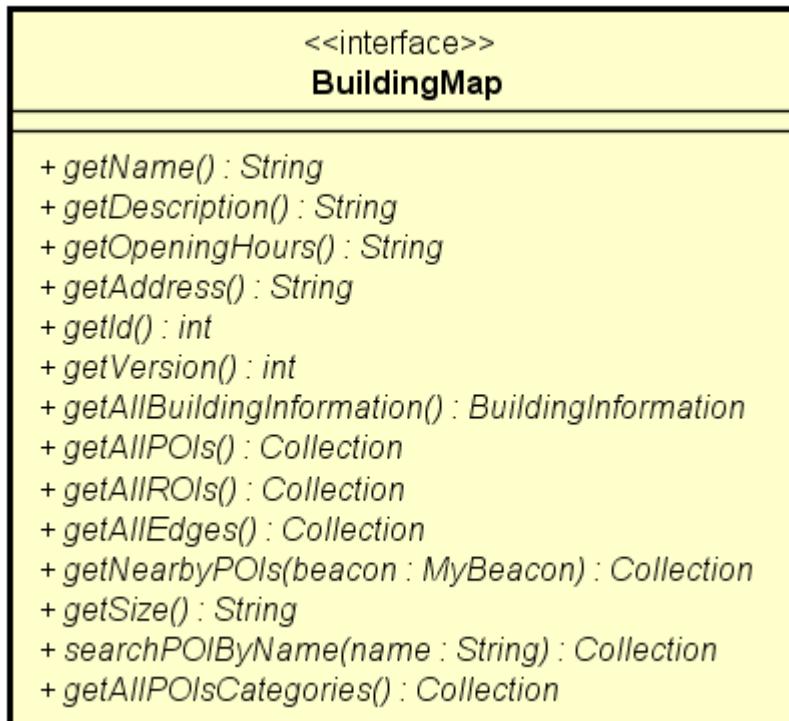


Figura 90: Interfaccia BuildingMap

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

3.4.74 model::navigator::BuildingMapImp

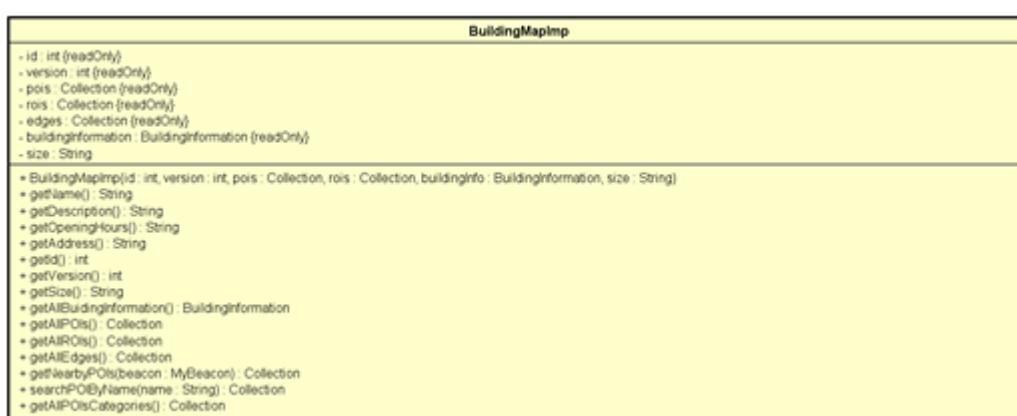


Figura 91: Classe BuildingMapImp

Nome: BuildingMapImp;

Tipo: Classe;

Implementa:

- BuildingMap.

Componenti delle librerie utilizzate:

- android.content.Intent (Android).

Visibilità: public;

Utilizzo: È utilizzata per accedere alle informazioni riguardanti la mappa di un edificio, alle informazioni generali dell'edificio stesso e per accedere alle collezioni di archi, POI e ROI in cui è stato decomposto un edificio;

Descrizione: Classe che rappresenta la mappa di un edificio con tutte le informazioni ad esso associate;

Attributi:

- - buildingInformation : BuildingInformation
Informazioni riguardanti l'edificio rappresentato dalla mappa
- - edges : Collection<EnrichedEdge> {readOnly}
Insieme di archi che indicano i possibili percorsi tra due ROI
- - id : int {readOnly}
Identificativo univoco dell'edificio
- - pois : Collection<PointOfInterest> {readOnly}
Insieme di POI appartenenti all'edificio rappresentato dalla mappa
- - rois : Collection<RegionOfInterest> {readOnly}
Insieme di ROI appartenenti all'edificio rappresentato dalla mappa
- - size : String
Dimensione della mappa dell'edificio (in MB)
- - version : int {readOnly}
Versione corrente della mappa

Metodi:

- + BuildingMapImp(id : int, version : int, pois : Collection<PointOfInterest>, rois : Collection<RegionOfInterest>, buildingInfo : BuildingInformation, size : String)
Costruttore della classe BuildingMapImp

Argomenti:

- **id** : int
Identificativo dell'edificio
- **version** : int
Versione della mappa
- **pois** : Collection<PointOfInterest>
Tutti i POI appartenenti all'edificio
- **rois** : Collection<RegionOfInterest>
Tutte le ROI appartenente all'edificio
- **buildingInfo** : BuildingInformation
Informazioni dell'edificio
- **size** : String
Dimensione della mappa dell'edificio (espressa in MB)
- + **getAddress()** : String
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
- + **getAllNearbyPOIs(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
- + **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
- + `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

3.4.75 model::navigator::NavigationExceptions

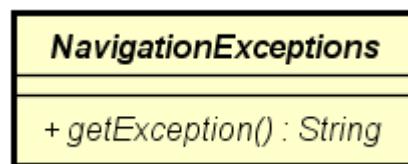


Figura 92: Classe astratta `NavigationExceptions`

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

3.4.76 model::navigator::Navigator

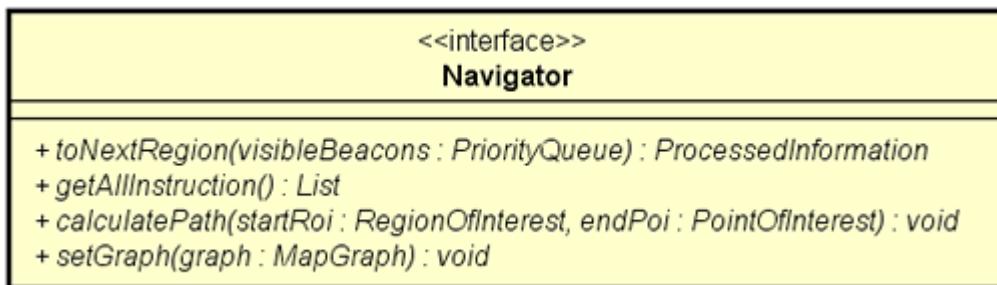


Figura 93: Interfaccia Navigator

Nome: Navigator;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente il modo in cui viene implementata la navigazione dal modo in cui è possibile usufruirne;

Descrizione: Interfaccia che espone i metodi per accedere alle funzionalità di navigazione;

Metodi:

- + *calculatePath(startRoi : RegionOfInterest, endRoi : RegionOfInterest : void)*

Metodo che calcola un percorso tra due RegionOfInterest viste come vertici di un grafo MapGraph utilizzando un oggetto PathFinder. Il punto di partenza e il punto di arrivo sono i parametri richiesti in input dal metodo mentre il grafo è un campo dati. Viene lanciata un'eccezione di tipo NoGraphSetException nel caso in cui non sia settato alcun grafo.

Argomenti:

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

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

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

- + *setGraph(graph : MapGraph) : void*

Metodo per settare il grafo sul quale calcolare il percorso.

Argomenti:

- *graph : MapGraph*
Grafo sul quale si vogliono calcolare dei percorsi.

- + *toNextRegion(visibleBeacons : PriorityQueue<MyBeacon> : ProcessedInformation)*

Metodo che ritorna le informazioni da seguire per raggiungere la prossima RegionOfInterest. Le informazioni fornite dipendono dalla lista di beacon passata come parametro in 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.

3.4.77 model::navigator::NavigatorImp

NavigatorImp	
- path : List	
- progress : Iterator	
- buildingGraph : MapGraph	
- pathFinder : PathFinder	
- compass : Compass	
+ NavigatorImp(compass : Compass)	
+ toNextRegion(visibleBeacons : PriorityQueue) : ProcessedInformation	
+ getAllInstruction() : List	
+ calculatePath(startRoi : RegionOfInterest, endPoi : PointOfInterest) : void	
+ setGraph(graph : MapGraph) : void	
+ getPath() : List	
- isShorter(firstPath : List, secondPath : List) : boolean	
- getStarterInformation() : String	
- getMostPowerfulBeacon(beacons : PriorityQueue) : MyBeacon	
- createInformation(edge : EnrichedEdge) : ProcessedInformation	

Figura 94: Classe NavigatorImp

Nome: NavigatorImp;

Tipo: Classe;

Implementa:

- Navigator.

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per calcolare un percorso da seguire durante la navigazione e per recuperare da quest'ultimo tutte le informazioni necessarie da fornire all'utente per seguire tale percorso;

Descrizione: Classe che si occupa della navigazione;

Attributi:

- - **buildingGraph** : `MapGraph`
Grafo dell'edificio in cui si desidera navigare
- - **compass** : `Compass`
Sensore di tipo Compass utilizzato all'avvio della navigazione
- - **path** : `List<EnrichedEdge>`
Lista di EnrichedEdge rappresentanti le indicazioni da seguire per raggiungere la destinazione
- - **pathFinder** : `PathFinder`
Campo dati che si occupa del calcolo del percorso
- - **progress** : `Iterator`
Iteratore rappresentante il punto che è stato raggiunto durante la navigazione

Metodi:

- + **calculatePath(startRoi : RegionOfInterest, endRoi : RegionOfInterest : void)**
Metodo che calcola un percorso tra due RegionOfInterest viste come vertici di un grafo MapGraph utilizzando un oggetto PathFinder. Il punto di partenza e il punto di arrivo sono i parametri richiesti in input dal metodo mentre il grafo è un campo dati. Viene lanciata un'eccezione di tipo NoGraphSetException nel caso in cui non sia settato alcun grafo

Argomenti:

- **startRoi** : `RegionOfInterest`
Punto di partenza del percorso
- **endRoi** : `RegionOfInterest`
Punto di arrivo del percorso

- - **createInformation(edge : EnrichedEdge) : ProcessedInformation**
Metodo che crea le ProcessedInformation in base al tipo di arco e in base alle informazioni provenienti dal beacon e da eventuali sensori utilizzati

Argomenti:

- **edge** : `EnrichedEdge`
Edge di cui devono essere recuperate le informazioni

- + **getAllInstructions() : List<ProcessedInformation>**
Metodo che ritorna la lista completa delle ProcessedInstruction da seguire per percorrere un percorso calcolato

- - `getMostPowerfullBeacon(beacons : PriorityQueue<MyBeacon> : MyBeacon)`
Metodo che ritorna il beacon con potenza maggiore tra quelli rilevati

Argomenti:

- `beacons : PriorityQueue<MyBeacon>`
Queue dei beacon rilevati

- + `getPath() : List<EnrichedEdge>`
Metodo per ottenere la lista di EnrichedEdge rappresentanti un percorso

- - `getStarterInformation() : String`
Metodo che ritorna le prime informazioni utili alla navigazione
- - `isShorter(firstPath : List<EnrichedEdge>, secondPath : List<EnrichedEdge>) : boolean`
Metodo per determinare se un percorso è più lungo o più breve di un altro percorso

Argomenti:

- `firstPath : List<EnrichedEdge>`
Lista di EnrichedEdge rappresentante un percorso
- `secondPath : List<EnrichedEdge>`
Lista di EnrichedEdge rappresentante un percorso

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

Argomenti:

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

- + `setGraph(graph : MapGraph) : void`
Metodo per settare il grafo sul quale calcolare il percorso

Argomenti:

- `graph : MapGraph`
Grafo sul quale si vogliono calcolare dei percorsi

- + `toNextRegion(visibleBeacons : PriorityQueue<MyBeacon> : ProcessedInformation)`
Metodo che ritorna le informazioni da seguire per raggiungere la prossima RegionOfInterest. Le informazioni fornite dipendono dalla lista di beacon passata come parametro in ingrasso e dal beacon più potente tra quelli in essa contenuti. Viene lanciata una

eccezione di tipo NoNavigationInformationException nel caso in cui si cerchi di accedere a tale metodo senza prima aver calcolato un percorso di navigazione. Viene lanciata una eccezione di tipo PathException nel caso in cui il beacon più potente nella lista di beacon in ingrasso sia associato ad una RegionOfInterest non appartenente ad una di quelle presenti nel percorso calcolato

Argomenti:

- visibleBeacons : PriorityQueue<MyBeacon>
Insieme di beacon visibili al momento della chiamata al metodo

3.4.78 model::navigator::NoGraphSetException

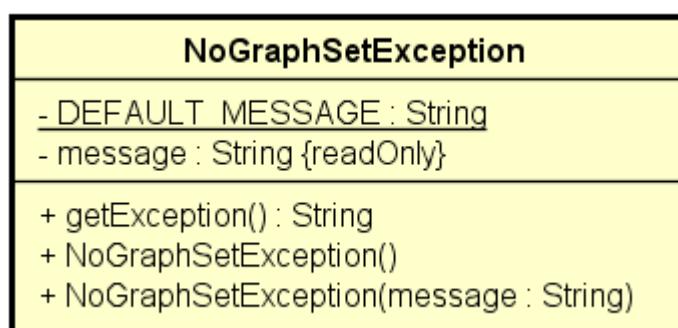


Figura 95: Classe NoGraphSetException

Nome: NoGraphSetException;

Tipo: Classe;

Estende:

- NavigationExceptions.

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per lanciare una eccezione nel caso in cui si cerchi di accedere alle funzioni di navigazione offerte da Navigator senza aver settato un grafo;

Descrizione: Classe che rappresenta una eccezione che può essere lanciata durante l'utilizzo della classe Navigator;

Attributi:

- - DEFAULT_MESSAGE : String {readOnly}
Messaggio di default associato all'eccezione nel caso in cui si avvi la navigazione senza aver settato il grafo
- - message : String {readOnly}
Stringa che rappresenta l'eccezione

Metodi:

- + getException() : String
Metodo che ritorna il messaggio che rappresenta l'eccezione
- + NoGraphSetException()
Costruttore di default della classe NoGraphSetException
- + NoGraphSetException(message : String)
Costruttore della classe NoGraphSetException

Argomenti:

- message : String
Messaggio che rappresenta l'eccezione

3.4.79 model::navigator::NoNavigationInformationException

NoNavigationInformationException
<u>- DEFAULT_MESSAGE : String</u> <u>- message : String {readOnly}</u>
+ <u>getException() : String</u> + <u>NoNavigationInformationException()</u> + <u>NoNavigationInformationException(message : String)</u>

Figura 96: Classe NoNavigationInformationException

Nome: NoNavigationInformationException;

Tipo: Classe;

Estende:

- NavigationExceptions.

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per lanciare un'eccezione nel caso in cui si voglia accedere alle informazioni di navigazione ma non è ancora stato calcolato un percorso;

Descrizione: Classe che rappresenta una eccezione che può essere lanciata durante l'utilizzo della classe Navigator;

Attributi:

- - DEFAULT_MESSAGE : String {readOnly}
Messaggio di default associato all'eccezione nel caso in cui si avvi la navigazione senza aver settato il grafo
- - message : String {readOnly}
Stringa che rappresenta l'eccezione

Metodi:

- + getException() : String
Metodo che ritorna il messaggio che rappresenta l'eccezione
- + NoNavigationInformationException()
Costruttore di default della classe NoNavigationInformationException
- + NoNavigationInformationException(message : String)
Costruttore della classe NoNavigationInformationException

Argomenti:

- message : String
Messaggio che rappresenta l'eccezione

3.4.80 model::navigator::PathException

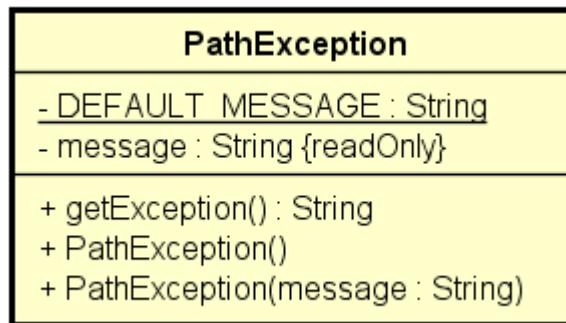


Figura 97: Classe PathException

Nome: PathException;

Tipo: Classe;

Estende:

- NavigationExceptions.

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per lanciare una eccezione nel caso in cui si rilevino dei beacon non previsti dal percorso calcolato;

Descrizione: Classe che rappresenta una eccezione che può essere lanciata durante l'utilizzo della classe Navigator;

Attributi:

- - `DEFAULT_MESSAGE : String {readOnly}`
Messaggio di default associato all'eccezione nel caso in cui si avvia la navigazione senza aver settato il grafo
- - `message : String {readOnly}`
Stringa che rappresenta l'eccezione

Metodi:

- + `getException() : String`
Metodo che ritorna il messaggio che rappresenta l'eccezione
- + `PathException()`
Costruttore di default della classe PathException
- + `PathException(message : String)`
Costruttore della classe PathException

Argomenti:

- `message : String`
Messaggio che rappresenta l'eccezione

3.4.81 model::navigator::ProcessedInformation

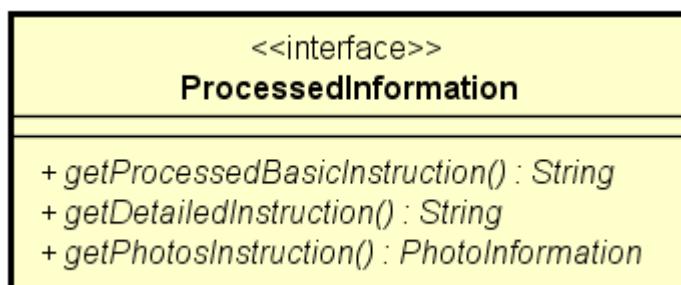


Figura 98: Interfaccia ProcessedInformation

Nome: ProcessedInformation;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione delle informazioni di navigazione e la loro costruzione dall'utilizzo di quest'ultime;

Descrizione: Interfaccia che espone i metodi per l'accesso alle informazioni di navigazione, pronte per essere restituite ad un utilizzatore di tali informazioni;

Metodi:

- + `getDetailedInstruction() : String`
Metodo che ritorna le istruzioni dettagliate per superare un certo arco nel percorso calcolato.

- + *getPhotoInstruction()* : *PhotoInformation*
Metodo che ritorna un oggetto PhotoInformation con il quale è possibile accedere alle fotografie che ritraggono l'arco da superare nel percorso calcolato
- + *getProcessedBasicInstruction()* : *String*
Metodo che ritorna le istruzioni basilari per superare un certo arco nel percorso calcolato.

3.4.82 model::navigator::ProcessedInformationImp

ProcessedInformationImp	
- basic : String	
- detailed : String	
- photos : List	
+ ProcessedInformationImp(edge : EnrichedEdge)	
+ ProcessedInformation(edge : EnrichedEdge, starterInformation : String)	
+ getProcessedBasicInstruction() : String	
+ getDetailedInstruction() : String	
+ getPhotosInstruction() : PhotoInformation	

Figura 99: Classe ProcessedInformationImp

Nome: ProcessedInformationImp;

Tipo: Classe;

Implementa:

- ProcessedInformation.

Componenti delle librerie utilizzate:

- org.altbeacon.beacon.startup.BootstrapNotifier (AltBeacon).

Visibilità: public;

Utilizzo: È utilizzata per accedere ai vari tipi di informazioni forniti quali informazioni di base, informazioni dettagliate e le foto dei luoghi da raggiungere;

Descrizione: Classe che rappresenta le informazioni di navigazione pronte per essere restituite ad un eventuale utilizzatore;

Attributi:

- - `basic` : `String`
Informazioni di base di un Edge
- - `detailed` : `String`
Indicazioni dettagliate di un Edge
- - `photos` : `List<String>`
Lista di foto di un Edge

Metodi:

- + `getDetailedInstruction()` : `String`
Metodo che ritorna le istruzioni dettagliate per superare un certo arco nel percorso calcolato
- + `getPhotoInstruction()` : `PhotoInformation`
Metodo che ritorna un oggetto PhotoInformation con il quale è possibile accedere alle fotografie che ritraggono l'arco da superare nel percorso calcolato
- + `getProcessedBasicInstruction()` : `String`
Metodo che ritorna le istruzioni basilari per superare un certo arco nel percorso calcolato
- + `ProcessedInformationImp(edge : EnrichedEdge)`
Costruttore della classe ProcessedInformationImp

Argomenti:

- `edge` : `EnrichedEdge`
Edge da cui devono essere estratte le informazioni
- + `ProcessedInformationImp(edge : EnrichedEdge, starterInformation : String)`
Costruttore della classe ProcessedInformationImp

Argomenti:

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

3.4.83 model::navigator::algorithm::DijkstraPathFinder



Figura 100: Classe DijkstraPathFinder

Nome: DijkstraPathFinder;

Tipo: Classe;

Estende:

- PathFinder.

Visibilità: public;

Utilizzo: È utilizzata per calcolare il percorso di navigazione sfruttando l'algoritmo di Dijkstra. Per fare questo utilizza la classe org.jgrapht.alg.DijkstraShortestPath<V, E>.

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

Metodi:

- + calculatePath(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
RegionOfInterest di partenza del percorso. Deve appartenere al grafo passato come paramentro.
- endROI : RegionOfInterest
RegionOfInterest di arrivo del percorso. Deve appartenere al grafo passato come paramentro.

- + DijkstraPathFinder()

Costruttore di default della classe DijkstraPathFinder

3.4.84 model::navigator::algorithm::PathFinder



Figura 101: 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.

3.4.85 model::navigator::graph::MapGraph

MapGraph
- graph : SimpleDirectedWeightedGraph
+ addRegionOfInterest(roi : RegionOfInterest) : void
+ addAllRegions(regions : Collection) : void
+ addEdge(edge : EnrichedEdge) : void
+ addAllEdges(edges : Collection) : void
+ getGraph() : SimpleDirectedWeightedGraph

Figura 102: 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 algorith. 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:

- + 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(Collection<RegionOfInterest> regions : 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(EnrichedEdge edge : 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(RegionOfInterest roi : 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

3.4.86 model::navigator::graph::area::PointOfInterest

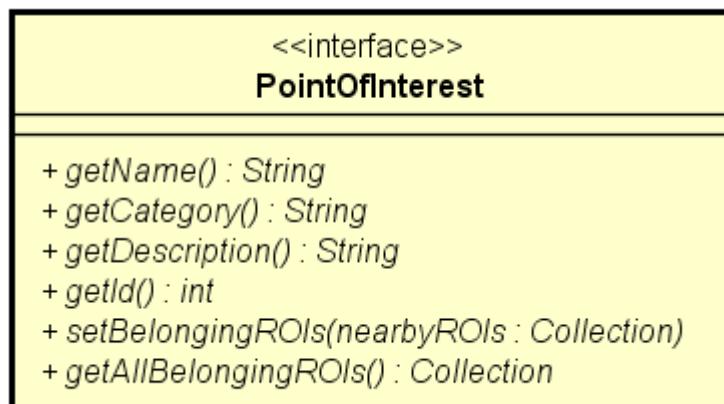


Figura 103: Interfaccia PointOfInterest

Nome: PointOfInterest;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione delle informazioni di un POI dal loro utilizzo;

Descrizione: Interfaccia che rappresenta un'area di interesse all'interno di un edificio. Espone i metodi per accedere alle informazioni di quest'area e per settare a quali aree coperte dal segnale di beacon appartiene;

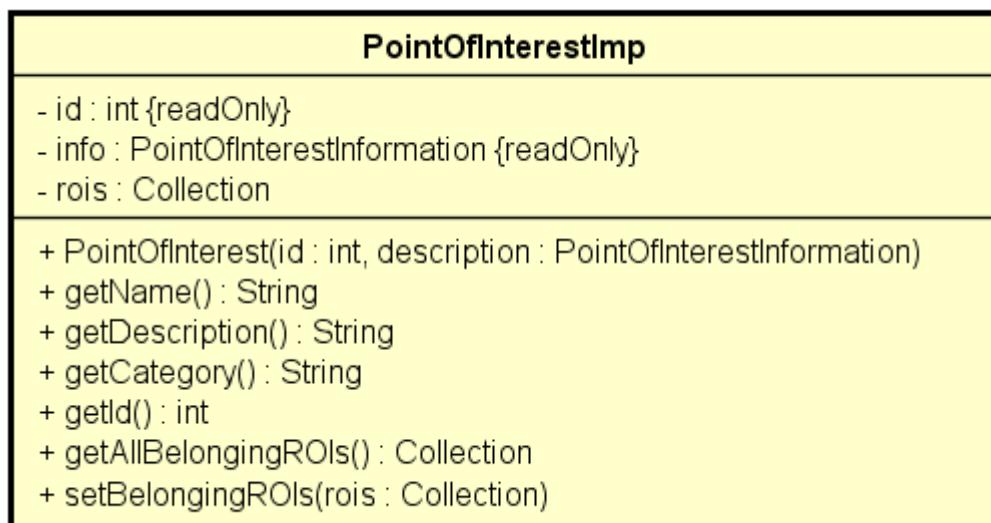
Metodi:

- **+ getBelongingROIs() : Collection<RegionOfInterest>**
Metodo che ritorna la collezione di RegionOfInterest alle quali tale oggetto appartiene
- **+ getCategory() : String**
Metodo che ritorna il nome della categoria di appartenenza del PointOfInterest
- **+ getDescription() : String**
Metodo che ritorna una descrizione del PointOfInterest
- **+ getId() : int**
Metodo che ritorna l'identificativo numerico associato al PointOfInterest

- + *getName()* : *String*
Metodo che ritorna il nome associato al PointOfInterest
- + *setAllBelongingROIs(rois : Collection<RegionOfInterest>)* : *void*
Metodo che permette di settare l'insieme di RegionOfInterest nelle quali tale PointOfInterest è contenuto

Argomenti:

- *rois* : *Collection<RegionOfInterest>*
Insieme di RegionOfInterest alle quali appartiene il PointOfInterest

3.4.87 model::navigator::graph::area::PointOfInterestImp**Figura 104:** Classe PointOfInterestImp**Nome:** PointOfInterestImp;**Tipo:** Classe;**Implementa:**

- PointOfInterest.

Visibilità: public;

Utilizzo: È utilizzata per definire dei punti di partenza e destinazione all'interno di un edificio, permettendo la navigazione tra di essi. Inoltre, viene utilizzata per indicare un punto di interesse nell'edificio e accedere alle sue informazioni;

Descrizione: Classe che rappresenta un POI, ossia un punto all'interno di un edificio ritenuto di possibile interesse;

Attributi:

- - `id` : `int {readOnly}`
Identificativo numerico di un `PointOfInterestImp`
- - `info` : `PointOfInterestInformation {readOnly}`
Informazioni relative ad un `PointOfInterestImp`
- - `rois` : `Collection<RegionOfInterest>`
Collezione degli oggetti `RegionOfInterest` alle quali appartiene l'oggetto

Metodi:

- + `getAllBelongingROIs()` : `Collection<RegionOfInterest>`
Metodo che ritorna la collezione di `RegionOfInterest` alle quali tale oggetto appartiene
- + `getCategory()` : `String`
Metodo che ritorna il nome della categoria di appartenenza del `PointOfInterestImp`.
- + `getDescription()` : `String`
Metodo che ritorna una descrizione del `PointOfInterestImp`.
- + `getId()` : `int`
Metodo che ritorna l'identificativo numerico associato al `PointOfInterestImp`.
- + `getName()` : `String`
Metodo che ritorna il nome associato al `PointOfInterestImp`
- + `PointOfInterestImp(id : int, info : PointOfInterestInformation)`
Costruttore della classe `PointOfInterestImp`

Argomenti:

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

- + setBelongingROIs(rois : Collection<RegionOfInterest>)

: void

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.

3.4.88 model::navigator::graph::area::PointOfInterestInformation

PointOfInterestInformation	
- description : String {readOnly}	
- name : String {readOnly}	
- category : String {readOnly}	
+ PointOfInterestInformation(name : String, description : String, category : String)	
+ getName() : String	
+ getDescription() : String	
+ getCategory() : String	

Figura 105: Classe PointOfInterestInformation

Nome: PointOfInterestInformation;

Tipo: Classe;

Visibilità: public;

Utilizzo: È utilizzata per recuperare le informazioni associate ad un POI, utili per fornire informazioni ad un utente durante la modalità esplorazione;

Descrizione: Classe che rappresenta le informazioni associate ad un POI;

Attributi:

- - category : String {readOnly}

Categoria di appartenenza del POI

- - description : String {readOnly}

Descrizione del POI

- - `name` : `String {readOnly}`
Nome del POI

Metodi:

- + `getCategory()` : `String`
Metodo che ritorna la categoria di appartenenza del PointOfInterest a cui l'oggetto è associato
- + `getDescription()` : `String`
Metodo che ritorna la descrizione del PointOfInterest a cui l'oggetto è associato
- + `getName()` : `String`
Metodo che ritorna il nome del PointOfInterest a cui l'oggetto è associato
- + `PointOfInterestInformation(name : String, description : String, category : String)`
Costruttore della classe PointOfInterestInformation

Argomenti:

- `name` : `String`
Nome del POI
- `description` : `String`
Descrizione del POI
- `category` : `String`
Categoria di appartenenza del POI

3.4.89 model::navigator::graph::area::RegionOfInterest

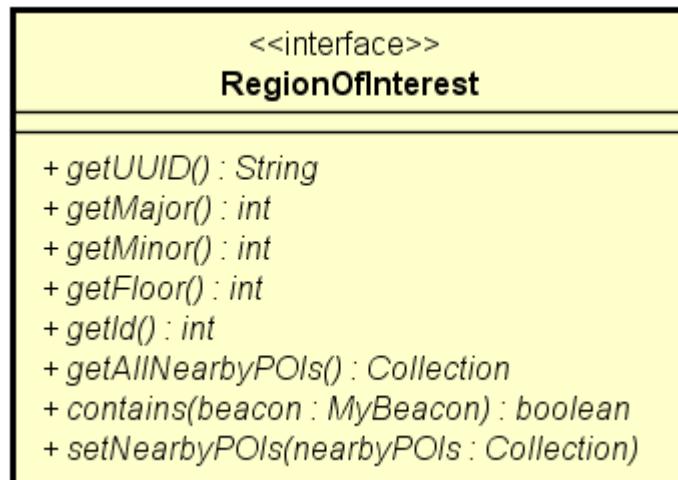


Figura 106: Interfaccia RegionOfInterest

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 da 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

3.4.90 model::navigator::graph::area::RegionOfInterestImp

RegionOfInterestImp	
- uuid : String {readOnly}	
- major : int {readOnly}	
- minor : int {readOnly}	
- id : int	
- pois : Collection	
+ RegionOfInterestImp(id : int, UUID : String, major : int, minor : int)	
+ getUUID() : String	
+ getMajor() : int	
+ getMinor() : int	
+ getFloor() : int	
+ getId() : int	
+ contains(beacon : MyBeacon) : boolean	
+ getAllNearbyPOIs() : Collection	
+ setNearbyPOIs(nearbyPOIs : Collection)	

Figura 107: Classe RegionOfInterestImp

Nome: RegionOfInterestImp;

Tipo: Classe;

Estende:

- VertexImp.

Implementa:

- RegionOfInterest.

Visibilità: public;

Utilizzo: È utilizzata per la navigazione all'interno di un edificio, permettendo di ottenere indicazioni o informazioni a seconda dell'appartenenza o meno ad una determinata ROI;

Descrizione: Classe che rappresenta una ROI, area coperta da un beacon che può contenere uno o più POI. Implementa la classe VertexImp;

Attributi:

- - `id` : `int {readOnly}`
Identificativo numerico di un oggetto RegionOfInterestImp
- - `major` : `int {readOnly}`
Identificativo Major del beacon associato alla ROI rappresentata dall'oggetto
- - `minor` : `int {readOnly}`
Identificativo Minor del beacon associato alla ROI rappresentata dall'oggetto
- - `pois` : `Collection<PointOfInterest>`
Collezione degli oggetti PointOfInterest appartenenti all'oggetto
- - `uuid` : `String {readOnly}`
Identificativo UUID del beacon associato alla ROI rappresentata dall'oggetto

Metodi:

- + `contains(beacon : MyBeacon) : boolean`
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 RegionOfInterestImp
- + `getFloor() : int`
Metodo che ritorna il piano dell'edificio nel quale la ROI rappresentata da tale oggetto si trova
- + `getMajor() : int`
Metodo che ritorna l'identificativo Major del beacon associato al ROI rappresentato da tale RegionOfInterestImp
- + `getMinor() : int`
Metodo che ritorna l'identificativo Minor del beacon associato al ROI rappresentato da tale RegionOfInterestImp

- + `getUUID() : String`
Metodo che ritorna l'identificativo UUID del beacon associato al ROI rappresentato da tale RegionOfInterestImp
 - + `RegionOfInterestImp(id : int, uuid : String, major : int, minor : int)`
Costruttore della classe RegionOfInterestImp
- Argomenti:**
- `id : int`
Identificativo numerico di un RegionOfInterestImp.
 - `uuid : String`
Identificativo UUID del beacon associato alla ROI rappresentata dall'oggetto
 - `major : int`
Identificativo Major del beacon associato alla ROI rappresentata dall'oggetto
 - `minor : int`
Identificativo Minor del beacon associato alla ROI rappresentata dall'oggetto
- + `setNearbyPOIs() : void`
Metodo utilizzato per settare l'insieme di PointOfInterest associato a tale RegionOfInterestImp

3.4.91 model::navigator::graph::edge::AbsEnrichedEdge

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

Figura 108: Classe astratta AbsEnrichedEdge

Nome: *AbsEnrichedEdge*;

Tipo: Classe astratta;

Implementa:

- `EnrichedEdge`.

Visibilità: `public`;

Utilizzo: È utilizzata per costruire il grafo di un edificio insieme a `Vertex`, necessario per la navigazione;

Descrizione: Classe astratta che rappresenta l'implementazione di base utilizzata per gli archi del grafo;

Attributi:

- - `coordinate` : `int {readOnly}`
Angolo rispetto al Nord polare tra il punto di inizio dell'arco e il punto finale dell'arco
- - `distance` : `double {readOnly}`
Lunghezza dell'arco
- - `endROI` : `RegionOfInterest {readOnly}`
Punto di arrivo dell'arco
- - `id` : `int {readOnly}`
Identificativo numerico di un arco
- - `navInfo` : `NavigationInformation {readOnly}`
Informazioni associate ad un arco per il superamento dello stesso
- - `startROI` : `RegionOfInterest {readOnly}`
Punto di partenza dell'arco
- - `userElevatorPreference` : `int {readOnly}`
Preferenze dell'utente rispetto agli archi che prevedono un ascensore
- # `userStairPreference` : `int {readOnly}`
Preferenze dell'utente rispetto agli archi che prevedono delle scale

Metodi:

- + `AbsEnrichedEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)`
Costruttore della classe `AbsEnrichedEdge`

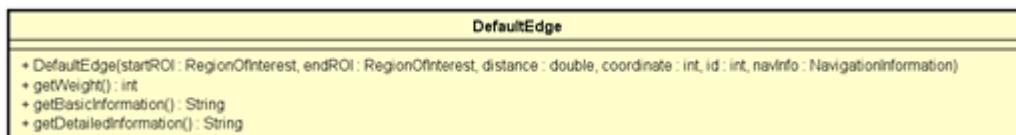
Argomenti:

- `startROI` : `RegionOfInterest`
Punto di partenza dell'arco
 - `endROI` : `RegionOfInterest`
Punto di arrivo dell'arco
 - `distance` : `double`
Lunghezza dell'arco
 - `coordinate` : `int`
Angolo rispetto al Nord polare tra il punto di partenza dell'arco e il punto di arrivo dell'arco
 - `id` : `int`
Identificativo numerico dell'arco
 - `navInfo` : `NavigationInformation`
Informazioni associate ad un arco per il superamento dello stesso
- + `getBasicInformation()` : `String`
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`
Metodo che ritorna la lunghezza di un arco
 - + `getEndPoint()` : `RegionOfInterest`
Metodo che ritorna la RegionOfInterest di arrivo dell'arco
 - + `getId()` : `int`
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`
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

- + `getWeight() : int`
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

3.4.92 model::navigator::graph::edge::DefaultEdge**Figura 109:** Classe DefaultEdge**Nome:** DefaultEdge;**Tipo:** Classe;**Estende:**

- AbsEnrichedEdge.

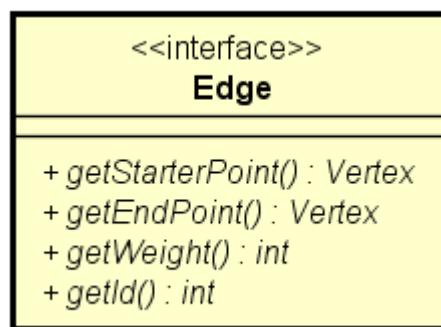
Visibilità: public;**Utilizzo:** È utilizzata per costruire il grafo di un edificio insieme a Vertex, necessario per la navigazione;**Descrizione:** Classe che implementa AbsEnrichedEdge e rappresenta un arco del grafo senza caratteristiche particolari;**Metodi:**

- + `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`
Metodo che ritorna le informazioni di base per attraversare l'arco
 - + **getDetailedInformation()** : `String`
Metodo che ritorna le informazioni di base per attraversare l'arco
 - + **getWeight()** : `int`
Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

3.4.93 model::navigator::graph::edge::Edge



Nome: Edge;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione di un arco dal suo utilizzo;

Descrizione: Interfaccia che serve per descrivere le funzionalità di un arco di un grafo;

Metodi:

- + *getEndPoint()* : *Vertex*
Metodo che restituisce il Vertex di arrivo dell'arco
- + *getId()* : *int*
Metodo che ritorna l'identificativo numerico dell'arco
- + *getStarterPoint()* : *Vertex*
Metodo che restituisce il Vertex di partenza dell'arco
- + *getWeight()* : *int*
Metodo che ritorna il peso dell'arco

3.4.94 model::navigator::graph::edge::ElevatorEdge

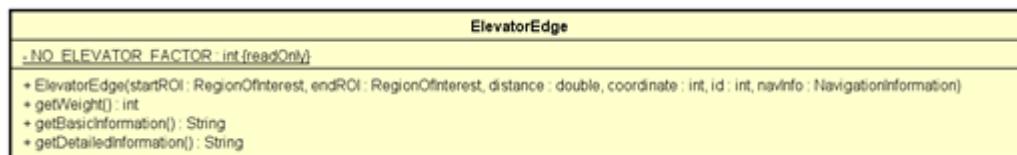


Figura 111: Classe ElevatorEdge

Nome: ElevatorEdge;

Tipo: Classe;

Estende:

- AbsEnrichedEdge.

Visibilità: public;

Utilizzo: È utilizzata per costruire il grafo di un edificio insieme a Vertex, necessario per la navigazione;

Descrizione: Classe che implementa AbsEnrichedEdge e rappresenta un arco del grafo che corrisponde ad un ascensore;

Attributi:

- - NO_ELEVATOR_FACTOR : int {readOnly}
Fattore da aggiungere al peso dell'arco in base alle preferenze di navigazione

Metodi:

- + ElevatorEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)
Costruttore della classe ElevatorEdge

Argomenti:

- startROI : RegionOfInterest
RegionOfInterest di partenza dell'arco
- endROI : RegionOfInterest
RegionOfInterest di arrivo dell'arco
- distance : double
Lunghezza dell'arco
- coordinate : int
Angolo rispetto al Nord polare presente tra il punto iniziale e il punto finale dell'arco
- id : int
Identificativo numerico dell'arco
- navInfo : NavigationInformation
Informazioni di navigazione associate all'arco

- + getBasicInformation() : String
Metodo che ritorna le informazioni di base per attraversare l'arco
- + getDetailedInformation() : String
Metodo che ritorna le informazioni di base per attraversare l'arco
- + getWeight() : int
Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

3.4.95 model::navigator::graph::edge::EnrichedEdge

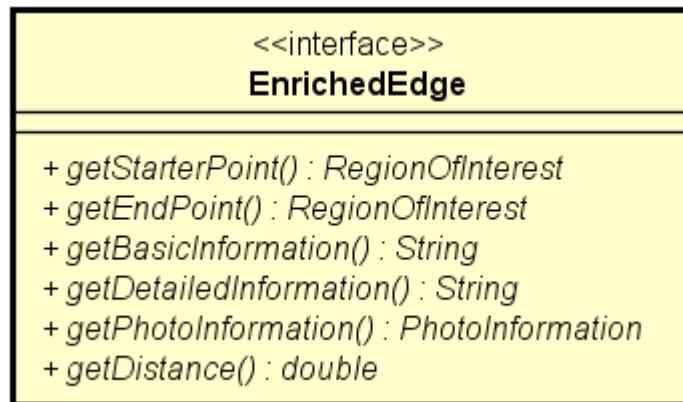


Figura 112: Interfaccia EnrichedEdge

Nome: EnrichedEdge;

Tipo: Interfaccia;

Estende:

- Edge.

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'utilizzo di tale tipo di arco, utile per costruire un insieme di informazioni per far seguire all'utente un determinato percorso, dal modo in cui queste informazioni sono gestite;

Descrizione: Interfaccia che serve per rappresentare un arco completo delle informazioni che possono aiutare a superarlo. Questo può rappresentare per esempio un corridoio di un edificio. Quest'interfaccia quindi espone i metodi per accedere a tali informazioni;

Metodi:

- + *getBasicInformation()* : String

Metodo astratto che ritorna le informazioni di base associate all'arco

- + *getDetailedInformation()* : *String*
Metodo astratto che ritorna le informazioni di base associate all'arco
- + *getDistance()* : *double*
È stata restituita la lunghezza dell'arco
- + *getEndPoint()* : *RegionOfInterest*
Metodo che ritorna la RegionOfInterest di arrivo dell'arco
- + *getPhotoInformation()* : *PhotoInformation*
Metodo che ritorna un oggetto PhotoInformation contenente un riferimento alle fotografie associate all'arco
- + *getStarterPoint()* : *RegionOfInterest*
Metodo che ritorna la RegionOfInterest di partenza dell'arco

3.4.96 model::navigator::graph::edge::StairEdge

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

Figura 113: Classe StairEdge

Nome: StairEdge;

Tipo: Classe;

Estende:

- AbsEnrichedEdge.

Visibilità: public;

Utilizzo: È utilizzata per costruire il grafo di un edificio insieme a Vertex, necessario per la navigazione;

Descrizione: Classe che implementa AbsEnrichedEdge e rappresenta un arco del grafo corrispondente ad una rampa di scale;

Attributi:

- - NO_STAIR_FACTOR : int {readOnly}

Fattore da aggiungere al peso dell'arco in base alle preferenze di navigazione

Metodi:

- + **getBasicInformation() : String**

Metodo che ritorna le informazioni dettagliate per attraversare l'arco

- + **getDetailedInformation() : String**

Metodo che ritorna le informazioni dettagliate per attraversare l'arco

- + **getWeight() : int**

Metodo che ritorna il peso dell'arco calcolato in base al tipo e alle preferenze dell'utente

- + **StairEdge(startROI : RegionOfInterest, endROI : RegionOfInterest, distance : double, coordinate : int, id : int, navInfo : NavigationInformation)**

Costruttore della classe StairEdge

Argomenti:

- **startROI : RegionOfInterest**

RegionOfInterest di partenza dell'arco

- **endROI : RegionOfInterest**

RegionOfInterest di arrivo dell'arco

- **distance : double**

Lunghezza dell'arco

- **coordinate : int**

Angolo rispetto al Nord polare presente tra il punto iniziale e il punto finale dell'arco

- **id : int**

Identificativo numerico dell'arco

- **navInfo : NavigationInformation**

Informazioni di navigazione associate all'arco

3.4.97 model::navigator::graph::navigationinformation::BasicInformation

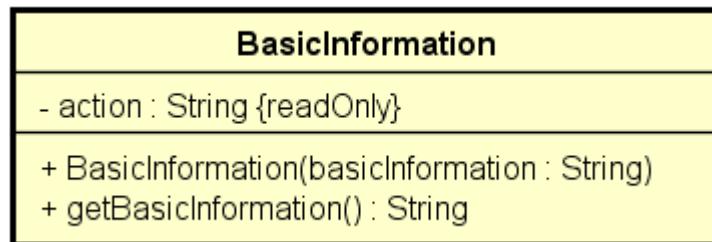


Figura 114: Classe BasicInformation

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

3.4.98 model::navigator::graph::navigationinformation::DetailedInformation

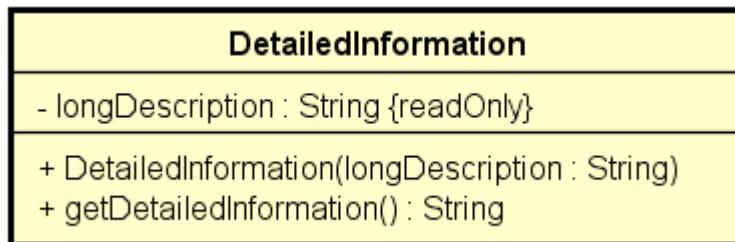


Figura 115: Classe DetailedInformation

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

3.4.99 model::navigator::graph::navigationinformation::NavigationInformation

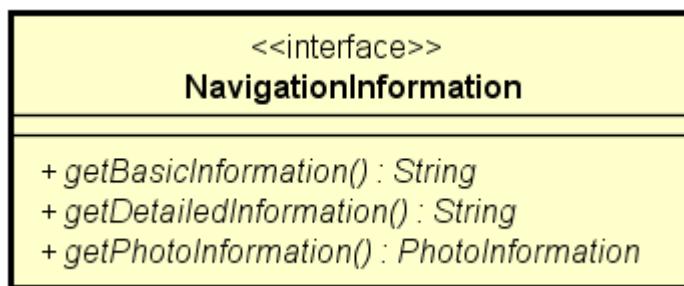


Figura 116: Interfaccia NavigationInformation

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

3.4.100 model::navigator::graph::navigationinformation::NavigationInformationImp

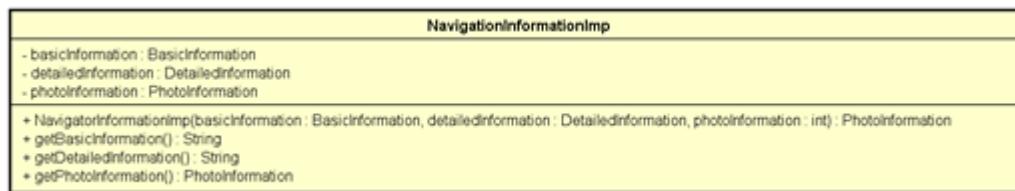


Figura 117: Classe NavigationInformationImp

Nome: NavigationInformationImp;

Tipo: Classe;

Implementa:

- NavigationInformation.

Visibilità: public;

Utilizzo: È utilizzata per recuperare le informazioni testuali, testuali estese e visive per permettere all’utente di navigare tramite dei campi dati di tipo BasicInformation, DetailedInformation e PhotoInformation;

Descrizione: Classe utilizzata per recuperare le informazioni da fornire all’utente per la navigazione;

Attributi:

- - basicInformation : BasicInformation {readOnly}
Informazioni di base associate ad un EnrichedEdge
- - detailedInformation : DetailedInformation {readOnly}
Informazioni di dettagliate associate ad un EnrichedEdge
- - photoInformation : PhotoInformation {readOnly}
Fotografie associate ad un EnrichedEdge

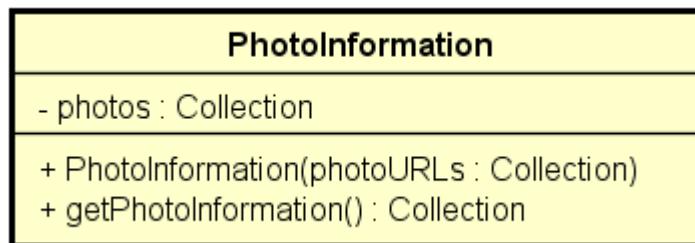
Metodi:

- + getBasicInformation() : String
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
- + `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

3.4.101 model::navigator::graph::navigationinformation::PhotoInformation**Figura 118:** Classe PhotoInformation**Nome:** PhotoInformation;**Tipo:** Classe;**Visibilità:** public;**Utilizzo:** È utilizzata per restituire le informazioni visive utilizzate durante la navigazione;

Descrizione: Classe che contiene le informazioni visive (sotto forma di URI a foto) utilizzate per la navigazione;

Attributi:

- - photos : Collection<PhotoRef> {readOnly}
Collezione di oggetti PhotoRef rappresentanti le fotografie dell'arco a cui l'oggetto è associato

Metodi:

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

3.4.102 model::navigator::graph::navigationinformation::PhotoRef

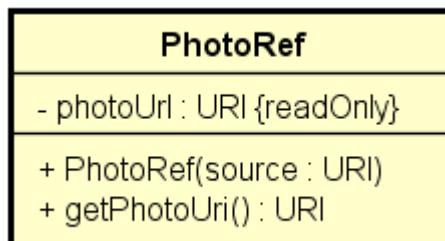


Figura 119: Classe PhotoRef

Nome: PhotoRef;

Tipo: Classe;

Visibilità: public;

Utilizzo: È utilizzata per fornire l'URI di una foto, necessario per il recupero della foto durante la navigazione;

Descrizione: Classe che rappresenta l'URI di una foto;

Attributi:

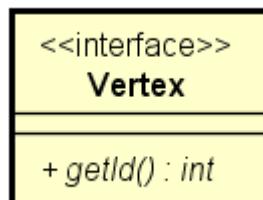
- - `photoUrl : URI {readOnly}`
URL di una fotografia

Metodi:

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

Argomenti:

- `source : URI`
URL di una fotografia

3.4.103 model::navigator::graph::vertex::Vertex**Figura 120:** 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

3.4.104 model::navigator::graph::vertex::VertexImp



Figura 121: Classe VertexImp

Nome: VertexImp;

Tipo: Classe;

Implementa:

- Vertex.

Visibilità: public;

Utilizzo: È utilizzata per costruire il grafo di un edificio, necessario per ricavare il percorso da seguire durante la navigazione;

Descrizione: Classe rappresentante il vertice (o nodo) di un grafo;

Attributi:

- - id : int {readOnly}
Identificativo numerico di un VertexImp

Metodi:

- + getId() : int
Metodo che ritorna l'identificativo numerico associato all'oggetto VertexImp
- + VertexImp(id : int)
Costruttore della classe VertexImp

Argomenti:

- id : int
Identificativo numerico di un oggetto VertexImp

3.4.105 model::usersetting::DeveloperCodeManager

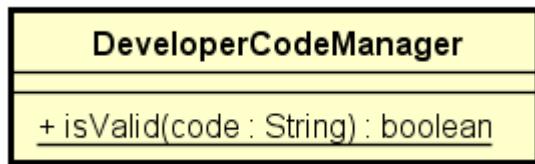


Figura 122: Classe DeveloperCodeManager

Nome: DeveloperCodeManager;

Tipo: Classe;

Visibilità: public;

Utilizzo: È utilizzata per verificare che un codice sviluppatore sia valido o meno;

Descrizione: Classe che per la verifica dei codici sviluppatore;

Metodi:

- + isValid(code : String) : boolean

Questo metodo permette di verificare se il codice inserito è valido per attivare la modalità sviluppatore

Argomenti:

- code : String

Questo parametro richiede il codice per attivare la modalità sviluppatore

3.4.106 model::usersetting::InstructionPreference

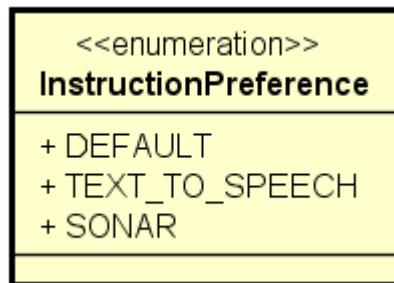


Figura 123: Classe InstructionPreference

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;

3.4.107 model::usersetting::PathPreference

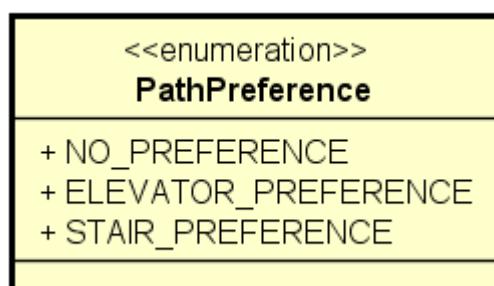


Figura 124: Classe PathPreference

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;

3.4.108 model::usersetting::Setting

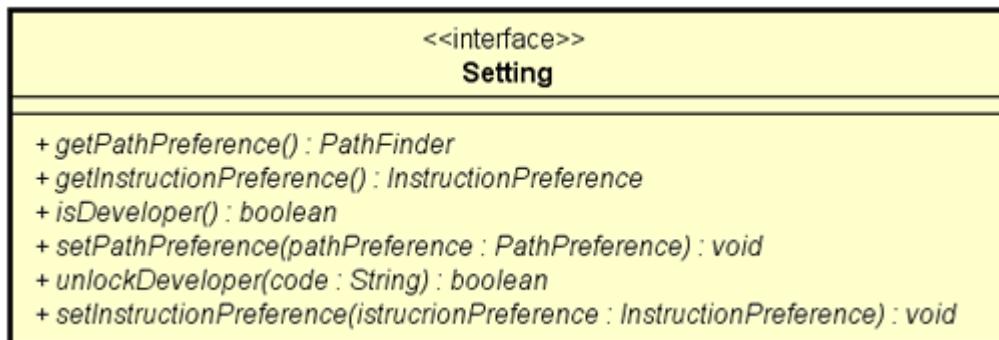


Figura 125: Interfaccia Setting

Nome: Setting;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'accesso e la gestione delle impostazioni di un utente dall'implementazione dei metodi e delle classi che si occupano di tale gestione;

Descrizione: Interfaccia che espone i metodi per accedere alle preferenze di un utente riguardo il percorso di navigazione e le istruzioni di navigazione. Espone inoltre i metodi per verificare se è stato inserito un codice sviluppatore valido e per verificare i codici inseriti;

Metodi:

- + *getInstructionPreference()* : *InstructionPreference*
Metodo che ritorna le preferenze riguardanti la modalità di fruizione delle informazioni
- + *getPathPreference()* : *PathPreference*
Metodo che ritorna le preferenze di percorso
- + *isDeveloper()* : *boolean*
Metodo che verifica se è stato inserito in precedenza un codice sviluppatore valido
- + *setInstructionPreference(instructionPreference : InstructionPreference : void)*
Metodo che permette di modificare le impostazioni riguardanti le preferenze di fruizione delle istruzioni di navigazione

Argomenti:

- *instructionPreference* : *InstructionPreference*
Preferenza riguardante la fruizione delle istruzioni di navigazione.
- + *setPathPreference(pathPreference : PathPreference) : void*
Metodo che permette di modificare le impostazioni riguardanti le preferenze sul percorso di navigazione

Argomenti:

- *pathPreference* : *PathPreference*
Preferenza riguardante il percorso di navigazione.
- + *unlockDeveloper() : boolean*
Metodo che passato una stringa in ingresso controlla se tale stringa rappresenta un codice sviluppatore valido

3.4.109 model::usersetting::SettingImp

SettingImp
- pathPreference : PathPreference - instructionPreference : InstructionPreference
+ getPathPreference() : PathPreference + getInstructionPreference() : InstructionPreference + isDeveloper() : boolean + unlockDeveloper(code : String) : boolean + setPathPreference(pathPreference : PathPreference) : void + setInstructionPreference(instructionPreference : InstructionPreference) : void

Figura 126: Classe SettingImp

Nome: SettingImp;

Tipo: Classe;

Implementa:

- Setting.

Componenti delle librerie utilizzate:

- android.content.SharedPreferences (Android);
- android.content.SharedPreferences.Editor (Android).

Visibilità: public;

Utilizzo: È utilizzata per accedere alle impostazioni relative a preferenze e alle opzioni di sviluppatore salvate sul dispositivo utilizzando la classe android.content.SharedPreferences. Ha inoltre il compito di modificare tali preferenze e impostazioni utilizzando le classi android.content.SharedPreferences e android.content.SharedPreferences.Editor;

Descrizione: Classe che implementa l'interfaccia Setting. Implementa tutti i metodi per la gestione delle impostazioni dell'utente;

Attributi:

- - `instructionPreference` : `InstructionPreference`
Preferenze dell'utente riguardanti le modalità di fruizione delle informazioni per la navigazione
- - `pathPreference` : `PathPreference`
Preferenze di percorso dell'utente

Metodi:

- + `getInstructionPreference()` : `InstructionPreference`
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`
Questo parametro richiede il tipo di istruzioni che si vuole ricevere durante la navigazione
- + `setPathPreference(pathPreference : PathPreference) : void`
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`
Metodo che passato una stringa in ingresso controlla se tale stringa rappresenta un codice sviluppatore valido

Argomenti:

- `code` : `String`
Questo parametro richiede il codice per attivare la modalità sviluppatore

3.4.110 presenter::Checker

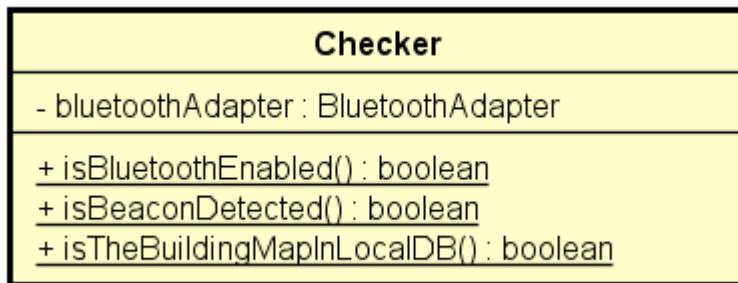


Figura 127: Classe Checker

Nome: Checker;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.bluetooth.BluetoothAdapter(Android).

Visibilità: public;

Utilizzo: È utilizzata per effettuare controlli sullo stato del dispositivo e dell'applicativo;

Descrizione: Classe che utilizza BluetoothAdapter e permette di effettuare controlli sullo stato del dispositivo e dell'applicativo;

Attributi:

- - bluetoothAdapter : BluetoothAdapter
Utilizzato per accedere al Bluetooth del dispositivo

Metodi:

- + isBeaconDetected() : boolean
Metodo che controlla se sono stati rilevati dei beacon
- + isBluetoothEnabled() : boolean
Metodo che controla se è stato attivato il Bluetooth del dispositivo
- + isTheBuildingMapInLocalDB() : boolean
Metodo che controlla se una mappa è presente sul database locale

3.4.111 presenter::DatabaseServiceManager



Figura 128: Classe DatabaseServiceManager

Nome: DatabaseServiceManager;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.app.Application(Android).

Visibilità: public;

Utilizzo: È utilizzata per fornire un riferimento al DatabaseService del model;

Descrizione: Classe che estende Application e recupera il riferimento al DatabaseService;

Attributi:

- - dbService : DatabaseService
Riferimento per accedere al database locale

Metodi:

- + getDatabaseService() : DatabaseService
Metodo che restituisce un riferimento al database locale

3.4.112 presenter::DetailedInformationActivity

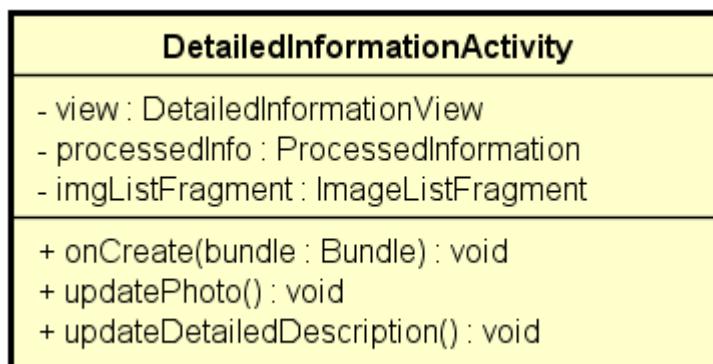


Figura 129: Classe DetailedInformationActivity

Nome: DetailedInformationActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per recuperare le informazioni dettagliate riguardo ad una certa istruzione di navigazione. Gestisce anche tutte le richieste che vengono fatte dalla DetailedInformationView;

Descrizione: Classe che estende AppCompatActivity per la gestione delle informazioni dettagliate riguardo alla navigazione;

Attributi:

- - imgListFragment : ImageListFragment
Contiene la lista di tutte le anteprime delle immagini associate ad un certo Edge
- - processedInfo : ProcessedInformation
ProcessedInformation associata all'Edge corrente
- - view : DetailedInformationView
View associata a tale Activity

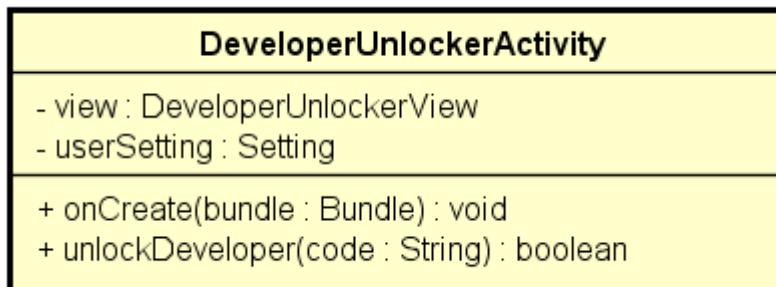
Metodi:

- + `onCreate(bundle : Bundle) : void`
Metodo che inizializza la DetailedInformationView

Argomenti:

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

- + `updateDetailedDescription() : void`
Metodo che aggiorna le informazioni testuali estese visualizzate sulla View
- + `updatePhoto() : void`
Metodo che aggiorna la foto visualizzata sulla View

3.4.113 presenter::DeveloperUnlockerActivity**Figura 130:** 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
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

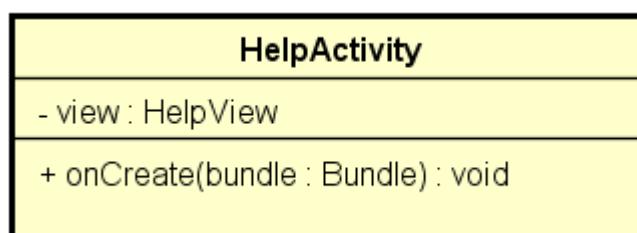
3.4.114 presenter::HelpActivity

Figura 131: Classe HelpActivity

Nome: HelpActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È usata per recuperare la guida dell'applicativo dal model e metterla a disposizione di HelpView. Gestisce tutte le possibili richieste effettuare da HelpView;

Descrizione: È una classe che estende AppCompatActivity che gestisce consente di gestire l'interazione tra HelpView ed il model;

Attributi:

- - view : HelpView
View associata a tale Activity

Metodi:

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

Argomenti:

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

3.4.115 presenter::HomeActivity

HomeActivity	
- view : HomeView	
- informationManager : InformationManager	
+ onCreate(bundle : Bundle) : void	
+ updateBuildingName() : void	
+ updateBuildingDescription() : void	
+ updateBuildingOpeningHours() : void	
+ updatePoiCategoryList() : void	
+ updateBuildingAddress() : void	
+ enableSuggestions() : void	
+ showPoisCategory(categoryName : String) : void	
+ showExplorer() : void	
+ showLocalMaps() : void	
+ showPreferences() : void	
+ showHelp() : void	
+ startNavigation(poiPosition : int) : void	

Figura 132: Classe HomeActivity

Nome: HomeActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per recuperare tutte le informazioni necessarie dal model al fine di popolare la HomeView. Gestisce anche tutte le richieste che vengono fatte dalla HomeView;

Descrizione: Classe che estende AppCompcatActivity per la gestione dell'interazione tra HomeView ed il model;

Attributi:

- - **informationManager** : `InformationManager`
Riferimento utilizzato per accedere alle informazioni trattate dal model
- - **view** : `HomeView`
View associata a tale Activity

Metodi:

- + **enableSuggestions()** : `void`
Metodo che permette di attivare la lista dei possibili POI raggiungibili a partire da una stringa
- + **onCreate(bundle : Bundle)** : `void`
Metodo che inizializza la View associata e recupera un riferimento all'InformationManager

Argomenti:

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

- + **showExplorer()** : `void`
Metodo che viene invocato a seguito della richiesta di visualizzazione della modalità esplora
- + **showHelp()** : `void`
Metodo che viene invocato a seguito della richiesta di visualizzazione della guida
- + **showLocalMaps()** : `void`
Metodo che viene invocato a seguito della richiesta di visualizzazione delle mappe salvate nel database locale
- + **showPoisCategory(categoryName : String)** : `void`
Metodo che viene invocato a seguito della richiesta di visualizzazione di tutti i POI appartenenti ad un certa categoria

Argomenti:

- **categoryName** : `String`
Nome della categoria di cui visualizzare l'insieme di POI appartenente

- + **showPreferences()** : `void`
Metodo che viene invocato a seguito della richiesta di visualizzazione delle preferenze dell'utente
- + **startNavigation(poiPosition : int)** : `void`
Metodo che viene invocato a seguito della richiesta di inizio della navigazione

Argomenti:

- poiPosition : int
Identificativo del POI verso il quale si vuole effettuare una navigazione
- + updateBuildingAddress() : void
Metodo che recupera l'indirizzo dell'edificio e lo passa alla View corrispondente
- + updateBuildingDescription() : void
Metodo che recupera la descrizione dell'edificio e lo passa alla View corrispondente
- + updateBuildingName() : void
Metodo che recupera il nome dell'indirizzo dell'edificio e lo passa alla View corrispondente
- + updateBuildingOpeningHours() : void
Metodo che recupera l'orario di apertura dell'edificio e lo passa alla View corrispondente
- + updatePoiCategoryList() : void
Metodo che recupera la lista di categorie di POI nell'edificio e lo passa alla View corrispondente

3.4.116 presenter::ImageAdapter

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

Figura 133: Classe ImageAdapter**Nome:** ImageAdapter;**Tipo:** Classe;**Componenti delle librerie utilizzate:**

- android.widget.BaseAdapter(Android).

Visibilità: public;

Utilizzo: È usata per gestire la lista di immagini associate ad una certa istruzione di navigazione;

Descrizione: Classe che estende FragmentStatePagerAdapter per gestire la lista di immagini associate ad una certa istruzione di navigazione;

Attributi:

- - context : Context
Contesto dell'applicativo

Metodi:

- + getCount() : int
Metodo che viene utilizzato per ottenere il numero di immagini relative a quel POI
- + getItem(position : int) : Object
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
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
Metodo che restituisce la foto in una certa posizione

Argomenti:

- position : int
Posizione della foto che si vuole recuperare
- convertView : View
Il layout dell'anteprima della foto restituita

- parent : ViewGroup
Il layout della lista di anteprime
- + ImageAdapter()
Costruttore della classe ImageAdapter

3.4.117 presenter::ImageDetailActivity

ImageDetailActivity	
- view : ImageDetailView	
- listPhotos : List	
- startItem : int	
- imgPgAdpt : ImagePageAdapter	
+ onCreate(bundle : Bundle) : void	

Figura 134: Classe ImageDetailActivity

Nome: ImageDetailActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per recuperare l'insieme di immagini associate ad una certa istruzione di navigazione ed esporle alla view che le utilizza. Gestisce anche tutte le richieste che vengono fatte dalla ImageDetailView;

Descrizione: Classe che implementa AppCompatActivity per la gestione dell'interazione tra ImageDetailView ed il model;

Attributi:

- - imgPgAdpt : ImagePageAdapter
ImagePageAdapter per la gestione dello slideshow delle immagini relative ad una certa istruzione di navigazione

- - `listPhotos : List<String>`
Lista contenenti gli URI delle foto
- - `startItem : int`
Elemento di partenza sul quale l'utente ha cliccato
- - `view : ImageDetailView`
View associata a tale Activity

Metodi:

- + `onCreate(bundle : Bundle) : void`
Metodo che inizializza ImageDetailView

Argomenti:

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

3.4.118 presenter::ImageListFragment

ImageListFragment	
-	<code>imgAdapter : ImageAdapter</code>
-	<code>photosUrls : List</code>
+	<code>ImageListFragment()</code>
+	<code>newInstance(photosUrls : List) : ImageListFragment</code>
+	<code>onCreate(bundle : Bundle) : void</code>
+	<code>onCreateView(inflater : LayoutInflater, viewGroup : ViewGroup, bundle : Bundle) : View</code>
+	<code>onItemClick() : void</code>

Figura 135: Classe ImageListFragment

Nome: ImageListFragment;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per la gestione dell'interazione tra lista di immagini associate ad una istruzione e le possibili azioni che è possibile effettuare su di esse;

Descrizione: Classe che estende BaseAdapter per la gestione della lista di immagini relative ad una istuzione 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`
Metodo che inizializza ImageView

Argomenti:

- `bundle : Bundle`
Componente per salvare lo stato dell'applicazione
- + `onCreateView(lay : LayoutInflator, viewGr : ViewGroup, bundle : Bundle) : View`
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() : void`
Metodo che viene invocato alla pressione di un bottone

3.4.119 presenter::ImagePagerAdapter

ImagePagerAdapter	
-	size : int
+	ImagePagerAdapter(fm : FragmentManager, size : int)
+	getCount() : int
+	getItem(position : int) : Fragment

Figura 136: Classe ImagePagerAdapter

Nome: ImagePagerAdapter;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.support.v4.app.FragmentStatePagerAdapter(Android).

Visibilità: public;

Utilizzo: È utilizzata per la gestione dello slideshow delle immagini relative ad una certa indicazione di navigazione;

Descrizione: Classe che estende FragmentStatePagerAdapter per la gestione dello slideshow delle immagini relative ad una certa istruzione di navigazione;

Attributi:

- - size : int
Numero di elementi della slideshow

Metodi:

- + getCount() : int
Metodo che fornisce il numero di elementi all'interno dello slideshow
- + getItem(position : int) : Fragment
Metodo che permette di ottenere un'immagine all'intero dello slideshow

Argomenti:

– position : int

Intero che indica la posizione dell'immagine da recuperare all'interno dello slideshow

- + ImagePagerAdapter(fm : FragmentManager, size : int)
Costruttore della classe ImagePagerAdapter

Argomenti:

– fm : FragmentManager

!

– size : int

!

3.4.120 presenter::InformationManagerPresenter

InformationManagerPresenter
- informationManager : InformationManager
+ getInformationManager() : InformationManager

Figura 137: Classe InformationManagerPresenter

Nome: InformationManagerPresenter;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.app.Application(Android).

Visibilità: public;

Utilizzo: È utilizzata per restituire un riferimento al NavigationManager del model;

Descrizione: Classe che estende Application e recupera il riferimento a InformationManager del model;

Attributi:

- - `informationManager : InformationManager`
Riferimento utilizzato per accedere alle informazioni trattate dal model

Metodi:

- + `getInformationManager() : InformationManager`
Metodo che restituisce un InformationManger per accedere alle informazioni del presenter

3.4.121 presenter::LocalMapActivity

LocalMapActivity	
-	<code>view : LocalMapManagerView</code>
+	<code>onCreate(bundle : Bundle) : void</code>
+	<code>updateMap(mapPosition : int) : void</code>
+	<code>deleteMap(mapPosition : int) : void</code>

Figura 138: Classe LocalMapActivity

Nome: LocalMapActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per recuperare le mappe salvate in locale dal model ed esporle a LocalMapView. Gestisce tutte le possibili richieste fatte da LocalMapView;

Descrizione: Classe che estende AppCompatActivity e per la gestione dell'interazione tra LocalMapView ed il model;

Attributi:

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

Metodi:

- + deleteMap(mapPosition : int) : void
Metodo che permette di rimuovere una mappa del database locale

Argomenti:

- mapPosition : int
Posizione occupata dalla mappa da rimuovere

- + onCreate(bundle : Bundle) : void
Metodo che inizializza la View associata a tale Activity

Argomenti:

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

- + updateMap(mapPosition : int) : void
Metodo che permette di aggiornare una mappa del database locale

Argomenti:

- mapPosition : int
Posizione della mappa da aggiornare

3.4.122 presenter::LoggingActivity

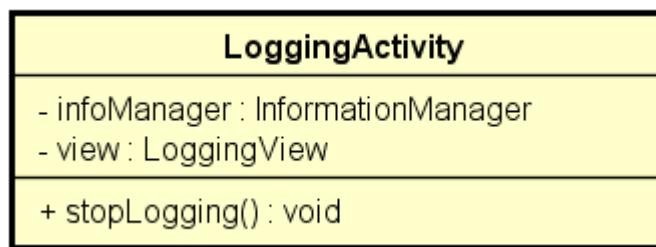


Figura 139: Classe LoggingActivity

Nome: LoggingActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È usata per arrestare un'attività di logging avviata e per recuperare i beacon circostanti. Gestisce tutte le possibili richieste effettuate da LoggingView;

Descrizione: Classe che estende AppCompatActivity per la gestione dell'interazione tra il model e LoggingView;

Attributi:

- - infoManager : InformationManager
Riferimento utile per gestire i log
- - view : LoggingView
View associata a tale Activity

Metodi:

- + stopLogging() : void
Metodo che viene utilizzato per interrompere l'attività di log

3.4.123 presenter::LogInformationActivity

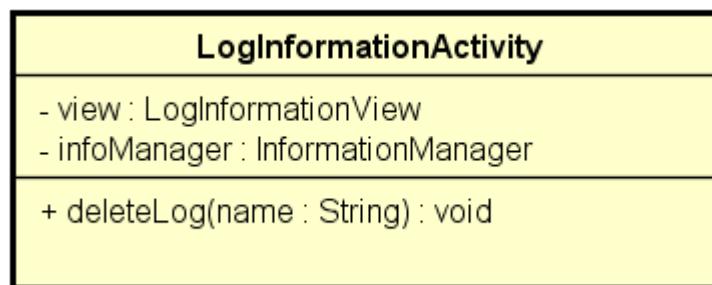


Figura 140: Classe LogInformationActivity

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

3.4.124 presenter::MainActivity

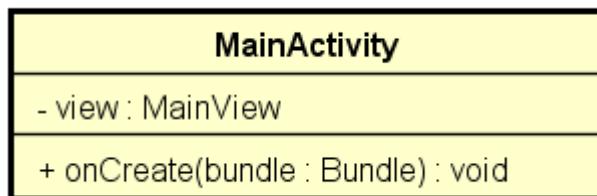


Figura 141: Classe MainActivity

Nome: MainActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per la gestione delle MainView;

Descrizione: Classe che implementa AppCompatActivity per la gestione dell'interazione tra MainView ed il model;

Attributi:

- - view : MainView
View associata a tale Activity

Metodi:

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

Argomenti:

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

3.4.125 presenter::MainDeveloperActivity

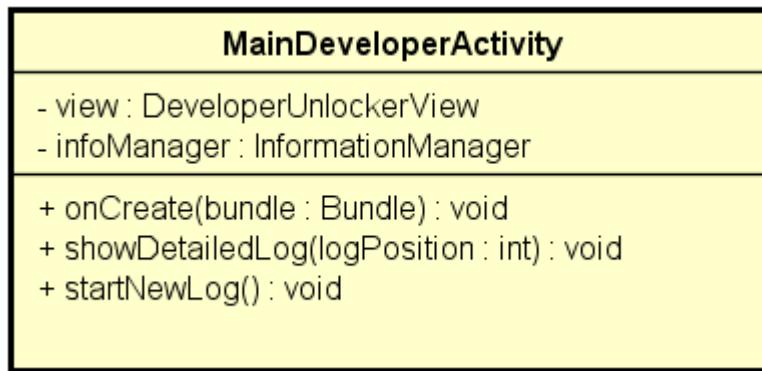


Figura 142: Classe MainDeveloperActivity

Nome: MainDeveloperActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È usata per recuperare i log dal model e avviare la registrazione di un nuovo log. Gestisce tutte le possibili richieste effettuate da MainDeveloperView;

Descrizione: È una classe che estende AppCompatActivity e consente di gestire l'interazione tra MainDeveloperView ed il model;

Attributi:

- - `infoManager` : `InformationManager`
Oggetto del Model per la gestione delle informazioni
- - `view` : `DeveloperUnlockerView`
View associata a tale Activity

Metodi:

- + `onCreate(bundle : Bundle) : void`
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

3.4.126 presenter::MainDeveloperPresenter

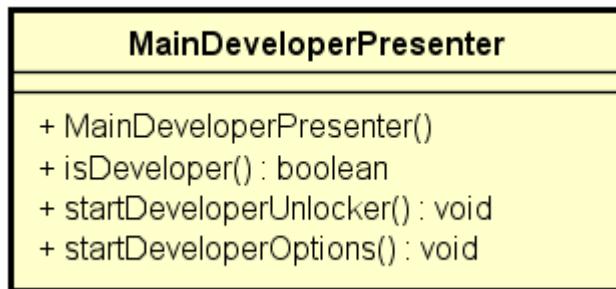


Figura 143: Classe MainDeveloperPresenter

Nome: MainDeveloperPresenter;

Tipo: Classe;

Visibilità: public;

Utilizzo: È utilizzata per discriminare la visualizzazione delle funzionalità sviluppatore tra un utente sviluppatore ed un utente che non lo è;

Descrizione: Classe che estende AppCompatActivity e controlla utilizzando il model, se l'utente è sviluppatore o meno;

Metodi:

- + isDeveloper() : boolean
Metodo che permette di verificare se un utente è sviluppatore
- + MainDeveloperPresenter()
Costruttore della classe MainDeveloperPresenter
- + startDeveloperOptions() : void
Metodo che consente di avviare la gestione delle funzionalità sviluppatore
- + startDeveloperUnlocker() : void
Metodo che consente di avviare la gestione dello sblocco delle funzionalità sviluppatore

3.4.127 presenter::MapDownloaderActivity

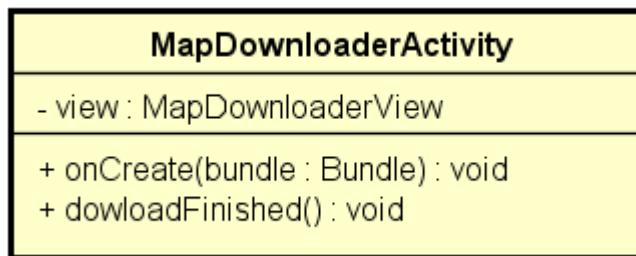


Figura 144: Classe MapDownloaderActivity

Nome: MapDownloaderActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per gestire il download o l'aggiornamento di una mappa. Gestisce anche tutte le possibili richieste effettuate da MapDownloaderView;

Descrizione: È una classe che estende AppCompatActivity che consente di gestire il download o l'aggiornamento delle mappe ;

Attributi:

- - view : MapDownloaderView
View associata a tale Activity

Metodi:

- + downloadFinished() : void
Metodo che gestisce la View per visualizzare il completamento del download di una mappa
- + onCreate(bundle : Bundle) : void
Metodo che inizializza la View associata

Argomenti:

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

3.4.128 presenter::NavigationActivity

NavigationActivity	
- view : NavigationView	
- navigationAdapter : NavigationAdapter	
+ onCreate(bundle : Bundle) : void	
+ pathError() : void	
+ informationUpdate(info : ProcessedInformation) : void	
+ showDetailedInformation(instructionPosition : int) : void	
+ stopNavigation() : void	

Figura 145: Classe NavigationActivity

Nome: NavigationActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per recuperare tutte le informazioni necessarie alla navigazione e renderle disponibili alla NavigationView. Gestisce anche tutte le richieste che vengono fatte dalla NavigationView;

Descrizione: Classe che estende AppCompatActivity per la gestione dell’interazione tra NavigationView ed il model;

Attributi:

- - navigationAdapter : NavigationAdapter
Consente di gestire l’insieme di funzioni per la navigazione
- - view : NavigationView
View associata a tale Activity

Metodi:

- + informationUpdate(info : ProcessedInformation) : void

Metodo che permette di aggiornare le informazioni

Argomenti:

- info : ProcessedInformation

Informazioni utili alla navigazione

- + onCreate(bundle : Bundle) : void

Metodo che inizializza NavigationView

Argomenti:

- bundle : Bundle

Componente per salvare lo stato dell'applicazione

- + pathError() : void

Metodo che viene invocato dal Model per segnalare un errore durante la navigazione

- + showDetailedInformation(instructionPosition : int) : void

Metodo che permette la visualizzazione delle informazioni dettagliate

Argomenti:

- instructionPosition : int

Intero rappresentante la posizione dell'informazione a cui accedere

- + stopNavigation() : void

Metodo che permette di interrompere la navigazione

3.4.129 presenter::NavigationAdapter

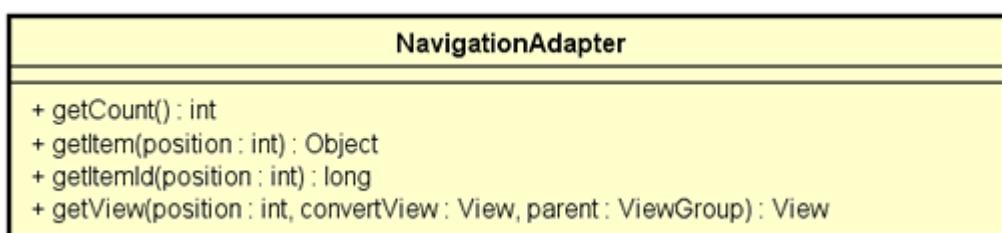


Figura 146: Classe NavigationAdapter

Nome: NavigationAdapter;

Tipo: Classe;

Componenti delle librerie utilizzate:

- `android.widget.BaseAdapter(Android)`.

Visibilità: public;

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

Descrizione: Classe che estende BaseAdapter per la gestione della lista di istruzioni di navigazione;

Metodi:

- `+ getCount() : int`

Metodo che ritorna il numero di istruzioni relative alla tua destinazione

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

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`

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`

Metodo che viene utilizzato per recuperare un'istruzione in una certa posizione

Argomenti:

- `position : int`

Posizione dell'istruzione da recuperare

- `convertView : View`

Layout dell'istruzione recuperata

- parent : ViewGroup
Layout della lista di istruzioni
- + NavigationAdapter()
Costruttore della classe NavigationAdapter

3.4.130 presenter::NavigationManagerPresenter

NavigationManagerPresenter	
-	navigationManager : NavigationManager
+	<u>getNavigationManager() : NavigationManager</u>

Figura 147: Classe NavigationManagerPresenter

Nome: NavigationManagerPresenter;

Tipo: Classe;

Componenti delle librerie utilizzate:

- android.app.Application(Android).

Visibilità: public;

Utilizzo: È utilizzata per fornire un riferimento al NavigationManager del model;

Descrizione: Classe che estende Application e recupera il riferimento al NavigationMager;

Attributi:

- - navigationManager : NavigationManager
Oggetto del Model utilizzato per gestire la navigazione

Metodi:

- + getNavigationManager() : NavigationManager
Metodo per ottenere l'oggetto NavigationManager

3.4.131 presenter::NearbyPoiActivity

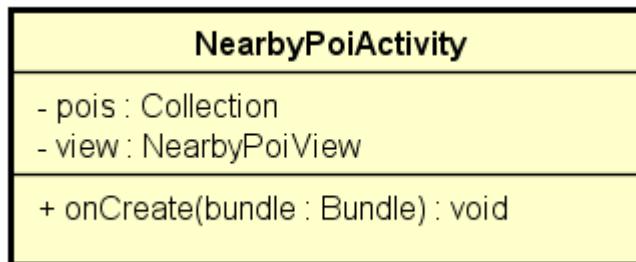


Figura 148: Classe NearbyPoiActivity

Nome: NearbyPoiActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per recuperare tutti i POI rilevati dal dispositivo e gestire tutte le richieste effettuate da NearbyPoiView;

Descrizione: Classe che estende AppCompatActivity è gestisce tutte le possibili interazioni tra NearbyPoiView ed il model;

Attributi:

- - pois : Collection<PointOfInterest>
Insieme di POI rilevati nelle circostanze
- - view : NearbyPoiView
View associata a tale Activity

Metodi:

- + onCreate(bundle : Bundle) : void
Metodo che istanzia NearbyPoiView

Argomenti:

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

3.4.132 presenter::PoiCategoryActivity

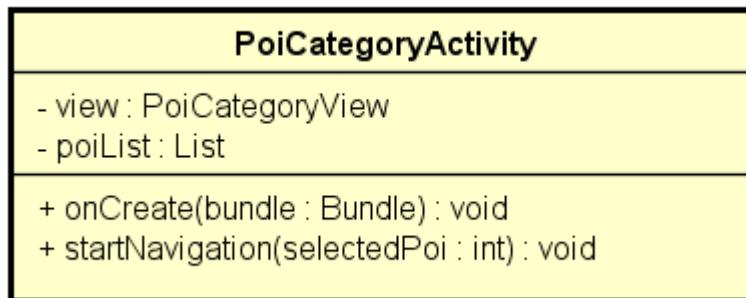


Figura 149: Classe PoiCategoryActivity

Nome: PoiCategoryActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per recuperare tutte le informazioni di tutti i POI associati ad una certa categoria dal model al fine di popolare la `PoiCategoryView`. Gestisce anche tutte le richieste che vengono fatte dalla `PoiCategoryView`;

Descrizione: Classe che implementa `AppCompatActivity` per la gestione dell'interazione tra `PoiCategoryView` ed il model;

Attributi:

- - `poiList : List<PointOfInterest>`
Lista di POI associati ad una certa categoria
- - `view : PoiCategoryView`
View associata a tale Activity

Metodi:

- + `onCreate(bundle : Bundle) : void`
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

3.4.133 presenter::PreferencesActivity

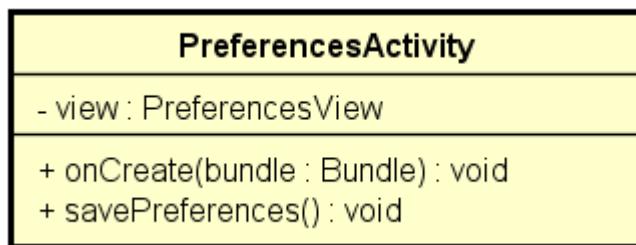


Figura 150: Classe PreferencesActivity

Nome: PreferencesActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per gestire l'interazione tra PreferencesView ed il model. Gestisce tutte le possibili richieste di PreferencesView;

Descrizione: È una classe che estende AppCompatActivity che consente di gestire le preferenze utente recuperandole dal model;

Attributi:

- - view : PreferencesView
View associata a tale Activity

Metodi:

- + `onCreate(bundle : Bundle) : void`

Metodo che inizializza la View associata

Argomenti:

- `bundle : Bundle`

Componente per salvare lo stato dell'applicazione

- + `savePreferences() : void`

Metodo che permette di salvare le preferenze utente

3.4.134 presenter::RemoteMapManagerActivity

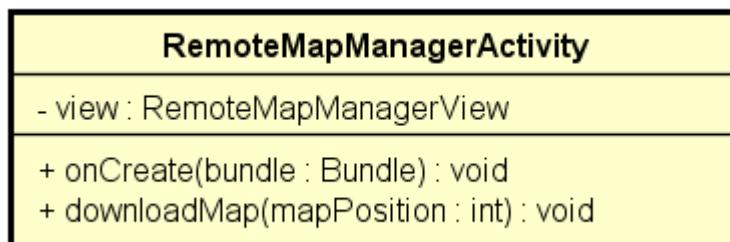


Figura 151: Classe RemoteMapManagerActivity

Nome: RemoteMapManagerActivity;

Tipo: Classe;

Componenti delle librerie utilizzate:

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

Visibilità: public;

Utilizzo: È utilizzata per gestire le mappe in remoto dal model per poterle esporre in RemoteMapManagerView. Gestisce anche tutte le possibili richieste effettuate da RemoteMapManagerView;

Descrizione: È una classe che estende AppCompatActivity che recupera tutte le mappe in remoto dal model e gestisce RemoteMapManagerView;

Attributi:

- - view : RemoteMapManagerView
View associata a tale Activity

Metodi:

- + downloadMap(mapPosition : int) : void
Metodo che permette di eseguire il download di una mappa da un database remoto

Argomenti:

- mapPosition : int
Posizione della mappa di cui fare il download

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

Argomenti:

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

3.4.135 presenter::SearchSuggestionsProvider

SearchSuggestionsProvider
<pre>-COLUMNS : String - homeActivity : HomeActivity</pre>
<pre>+ query(Uri : uri, projection : String[], selection : String, selectionArgs : String[], sortOrder : String) : Cursor + getType(Uri : uri) : String + insert(Uri : uri, values : ContentValues) : Uri + update(uri : Uri, values : ContentValues, selection : String, selectionArgs : String[]) : int + onCreate() : void</pre>

Figura 152: Classe SearchSuggestionsProvider**Nome:** SearchSuggestionsProvider;**Tipo:** Classe;**Componenti delle librerie utilizzate:**

- android.content.ContentProvider(Android).

Visibilità: public;**Utilizzo:** È usata per la gestione suggerimenti di ricerca per la navigazione;**Descrizione:** Classe che estende content Provider e si occupa della gestione suggerimenti di ricerca per la navigazione;

Attributi:

- - **COLUMNS** : String {readOnly}
Identifica la struttura di un singolo suggerimento
- - **homeActivity** : HomeActivity
Riferimento all'activity che gestisce la ricerca dei POI in base al nome

Metodi:

- + **getType(uri : Uri) : String**
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**
Metodo che serve per inserire i suggerimenti nel content provider

Argomenti:

- **uri** : Uri
URI in cui fare l'inserimento
- **values** : ContentValues
Insieme di coppie nome-colonna/valore da inserire nel content provider

- + **onCreate() : void**
Metodo che inizializza un oggetto di tipo SearchSuggestionProvider

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

Metodo che serve per popolare i suggerimenti della SearchView in base al testo inserito

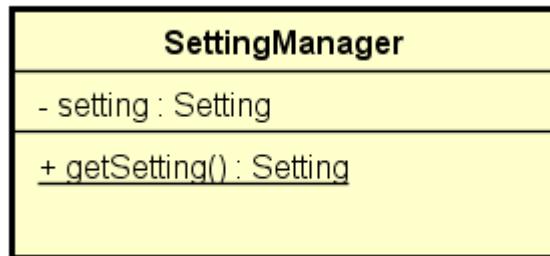
Argomenti:

- **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**
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

3.4.136 presenter::SettingManager**Figura 153:** Classe SettingManager**Nome:** SettingManager;**Tipo:** Classe;**Visibilità:** public;**Utilizzo:** È usata per fornire un punto di accesso alle impostazioni utente nel model;

Descrizione: È una classe che estende AppCompatActivity e consente di recuperare un riferimento verso le impostazioni utente nel model;

Attributi:

- - `setting : Setting`
Le impostazioni e preferenze dell'utente

Metodi:

- + `getSetting() : Setting`
Metodo che viene utilizzato per ottenere le impostazioni e le preferenze dell'utente

3.4.137 view::DetailedInformationView

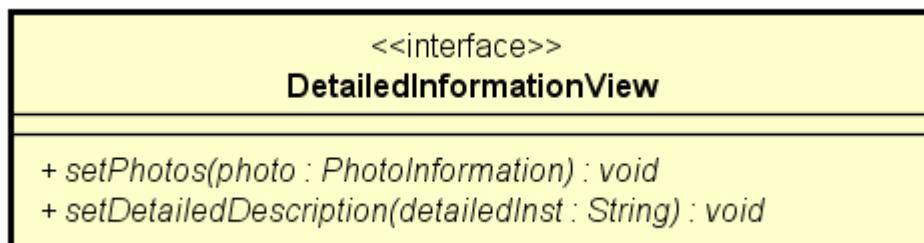


Figura 154: Interfaccia DetailedInformationView

Nome: `DetailedInformationView`;

Tipo: Interfaccia;

Visibilità: `public`;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente la versione dettagliata di una certa istruzione e le foto relative al POI destinazione di tale istruzione;

Metodi:

- + `setDetailedDescription(detailedInst : String) : void`
Metodo utilizzato per visualizzare la descrizione dettagliata relativa ad una certa istruzione

Argomenti:

– detailedInst : String

 Descrizione dettagliata relativa ad una certa istruzione di navigazione

- + *setPhotos(photo : PhotoInformation) : void*

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

Argomenti:

– photo : PhotoInformation

 Lista degli URI delle immagini relative ad un certo POI

3.4.138 view::DetailedInformationViewImp

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

Figura 155: Classe DetailedInformationViewImp

Nome: DetailedInformationViewImp;

Tipo: Classe;

Implementa:

- DetailedInformationView.

Visibilità: public;

Utilizzo: La lista di anteprime deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista possa reagire alla pressione su di esso. Per recuperare un riferimento alla lista è sufficiente utilizzare la classe R.id di Android;

Descrizione: Classe che si occupa di mostrare: le anteprime delle foto rappresentanti il prossimo POI, la descrizione dettagliata di una certa informazione di navigazione. La UI legata a questa classe permette all'utente di accedere alle foto rappresentanti il prossimo POI;

Attributi:

- - presenter : DetailedInformationActivity
Presenter della View

Metodi:

- + DetailedInformationViewImp(presenter : DetailedInformationActivity)
Costruttore della classe DetailedInformationViewImp

Argomenti:

- presenter : DetailedInformationActivity
Presenter della View che viene creata

- + setDetailedDescription(detailedInst : String) : void
Metodo utilizzato per visualizzare la descrizione dettagliata relativa ad una certa istruzione

Argomenti:

- detailedInst : String
Descrizione dettagliata relativa ad una certa istruzione di navigazione

- + setPhotos(urls : List<String>) : void
Metodo utilizzato per visualizzare la lista delle anteprime delle foto relative ad un certo POI

Argomenti:

- urls : List<String>
Lista degli URI delle immagini relative ad un certo POI

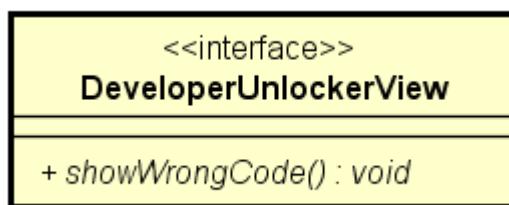
3.4.139 view::DeveloperUnlockerView

Figura 156: 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'u-tente, 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

3.4.140 view::DeveloperUnlockerViewImp

DeveloperUnlockerViewImp	
- presenter : DeveloperUnlockerActivity	
- plainTxtCode : EditText	
- txtInsertCode : TextView	
- btnInsertCode : Button	
+ DeveloperUnlockerViewImp(presenter : DeveloperUnlockerActivity)	
+ showWrongCode() : void	

Figura 157: Classe DeveloperUnlockerViewImp

Nome: DeveloperUnlockerViewImp;

Tipo: Classe;

Implementa:

- DeveloperUnlockerView.

Visibilità: public;

Utilizzo: Mantiene i riferimenti agli elementi di layout che compongono la UI dell'isenrimento. Tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso;

Descrizione: Classe che si occupa di mostrare i campi utili all'inserimento del codice sviluppatore ;

Attributi:

- - `btnInsertCode` : `Button`
Bottone per confermare l'inserimento del codice sviluppatore
- - `plainTxtCode` : `EditText`
`EditText` in cui è possibile inserire il codice sviluppatore
- - `presenter` : `DeveloperUnlockerActivity`
Presenter della View
- - `txtInsertCode` : `TextView`
`TextView` all'interno della quale viene visualizzato un suggerimento per l'inserimento del codice sviluppatore

Metodi:

- + `DeveloperUnlockerViewImp(presenter : DeveloperUnlokerActivity)`
Costruttore della classe `DeveloperUnlockerViewImp`

Argomenti:

- `presenter` : `DeveloperUnlokerActivity`
Presenter della View che viene creata
- + `showWrongCode() : void`
Metodo utilizzato per visualizzare un errore relativo all'errato inserimento del codice sviluppatore

3.4.141 view::HelpView

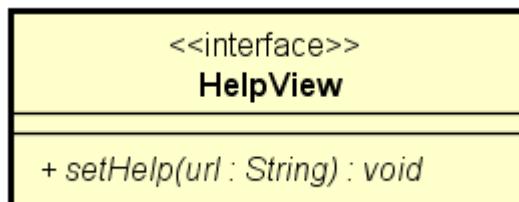


Figura 158: Interfaccia HelpView

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

3.4.142 view::HelpViewImp

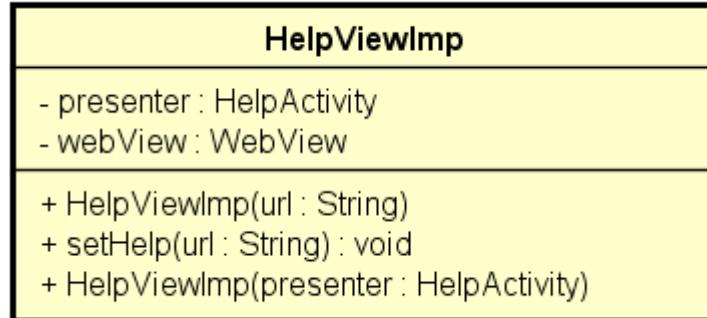


Figura 159: Classe HelpViewImp

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`
Metodo utilizzato per visualizzare la guida dell'applicazione

Argomenti:

- `url : String`
Stringa rappresentante l'URL a cui recuperare la guida dell'applicazione

3.4.143 view::HomeView

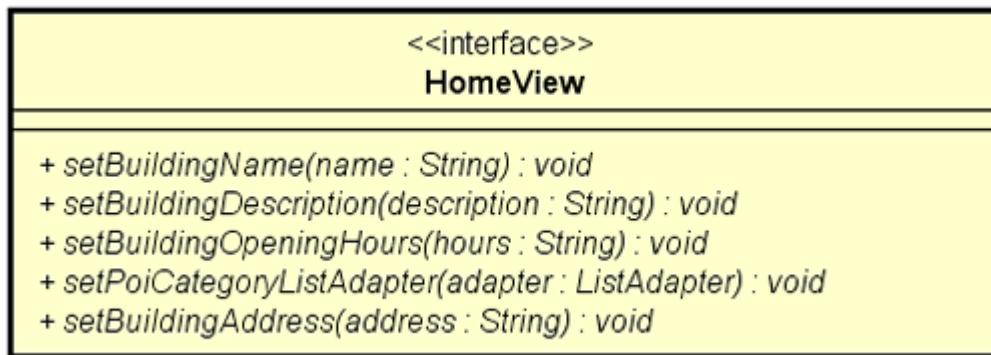


Figura 160: Interfaccia HomeView

Nome: HomeView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI della Home;

Metodi:

- **+ setBuildingAddress(address : String) : void**
Metodo utilizzato per visualizzare l'indirizzo di un edificio

Argomenti:

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

- **+ setBuildingDescription(description : String) : void**
Metodo utilizzato per visualizzare la descrizione di un edificio

Argomenti:

- **description : String**
Descrizione dell'edificio

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

Metodo utilizzato per visualizzare il nome di un edificio

Argomenti:

– name : String

Nome dell'edificio

- + *setBuildingOpeningHours(hours : String) : void*

Metodo utilizzato per visualizzare gli orari di apertura di un edificio

Argomenti:

– hours : String

Orari di apertura

- + *setPoiCategoryListAdapter(adapter : ListAdapter) : void*

Metodo utilizzato per visualizzare la lista di categorie dei POI dell'edificio

Argomenti:

– adapter : ListAdapter

Collegamento tra la lista delle categorie di POI dell'edificio e la view in cui esse devono essere mostrate

3.4.144 view::HomeViewImp

HomeViewImp	
- presenter : HomeActivity	
- layoutId : int	
- toolbar : Toolbar	
- drawer : DrawerLayout	
- navigationView : NavigationView	
- searchView : SearchView	
- buildingAddress : TextView	
- buildingName : TextView	
- buildingDescription : TextView	
- buildingOpeningHours : TextView	
- poiCategories : ListView	
- exploreButton : FloatingActionButton	
+ HomeViewImp(presenter : HomeActivity)	
+ setBuildingName(name : String) : void	
+ setBuildingDescription(description : String) : void	
+ setBuildingOpeningHours(hours : String) : void	
+ setPoiCategoryListAdapter(adapter : ListAdapter) : void	
+ setBuildingAddress(address : String) : void	

Figura 161: Classe HomeViewImp

Nome: HomeViewImp;

Tipo: Classe;

Implementa:

- HomeView.

Visibilità: public;

Utilizzo: Mantiene i riferimenti agli elementi di layout che compongono la UI della Home. Tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso;

Descrizione: Classe che si occupa di mostrare: informazioni relative all'edificio, categorie di POI presenti nell'edificio. La UI legata a questa classe permette all'utente di accedere alle sezioni principali dell'applicazione e alla lista dei POI che si trovano nelle vicinanze dell'utente;

Attributi:

- - `buildingAddress` : `TextView`
TextView all'interno della quale viene visualizzato l'indirizzo dell'edificio
- - `buildingDescription` : `TextView`
TextView all'interno della quale viene visualizzata la descrizione dell'edificio
- - `buildingName` : `TextView`
TextView all'interno della quale viene visualizzato il nome dell'edificio
- - `buildingOpeningHours` : `TextView`
TextView all'interno della quale viene visualizzato l'orario di apertura dell'edificio
- - `drawer` : `DrawerLayout`
Permette di estrarre delle view dagli angoli dello schermo
- - `exploreButton` : `FloatingActionButton`
Bottone che permette di trovare i POI intorno all'utente
- - `navigationView` : `NavigationView`
View che gestisce la navigazione
- - `poiCategories` : `ListView`
View che si occupa di mostrare tutte le categorie dei POI
- - `presenter` : `HomeActivity`
Presenter della View
- - `searchView` : `SearchView`
View che permette la ricerca delle destinazioni di navigazione
- - `toolbar` : `Toolbar`
Barra degli strumenti che permette la visualizzazione del menu

Metodi:

- + `HomeViewImp(presenter : HomeActivity)`
Costruttore della classe HomeViewImp

Argomenti:

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

- + `setBuildingAddress(address : String) : void`
Metodo utilizzato per visualizzare l'indirizzo di un edificio

Argomenti:

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

- + `setBuildingDescription(description : String) : void`
Metodo utilizzato per visualizzare la descrizione di un edificio

Argomenti:

- `description` : `String`
Descrizione dell'edificio

- + `setBuildingName(name : String) : void`
Metodo utilizzato per visualizzare il nome di un edificio

Argomenti:

- `name` : `String`
Nome dell'edificio

- + `setBuildingOpeningHours(hours : String) : void`
Metodo utilizzato per visualizzare gli orari di apertura di un edificio

Argomenti:

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

- + `setPoiCategoryListAdapter(adapter : ListAdapter) : void`
Metodo utilizzato per visualizzare la lista di categorie dei POI dell'edificio

Argomenti:

- `adapter` : `ListAdapter`
Collegamento tra la lista delle categorie di POI dell'edificio e la view in cui esse devono essere mostrate

3.4.145 view::ImageDetailView



Figura 162: Interfaccia ImageDetailView

Nome: ImageDetailView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente lo slideshow delle immagini relative ad un certo POI;

Metodi:

- + *setAdapter(adp : Adapter) : void*

Metodo utilizzato per visualizzare una slideshow di immagini relative al POI da raggiungere

Argomenti:

- *adp : Adapter*

Collegamento tra la lista delle immagini e la view in cui esse devono essere mostrate

3.4.146 view::ImageDetailViewImp

ImageDetailViewImp	
- presenter : ImageDetailActivity	
- pager : ViewPager	
+ setAdapter(adp : Adapter) : void	
+ ImageDetailViewImp(presenter : ImageDetailActivity) : void	

Figura 163: Classe ImageDetailViewImp

Nome: ImageDetailViewImp;

Tipo: Classe;

Implementa:

- ImageDetailView.

Visibilità: public;

Utilizzo: Mantiene un riferimento all'elemento di layout utile per passare da una foto alla successiva. alla pressione su di esso.;

Descrizione: Classe che si occupa di mostrare una foto relativa ad un certo POI. La UI legata a questa classe permette all'utente di accedere a tutte le foto relative allo stesso POI;

Attributi:

- - pager : ViewPager
Manager del layout che permette di capovolgere la view
- - presenter : ImageDetailActivity
Presenter della View

Metodi:

- + ImageDetailViewImp(presenter : ImageDetailActivity) : void
Costruttore della classe ImageDetailViewImp

Argomenti:

- **presenter** : `ImageDetailActivity`
Presenter della View che viene creata
- + `setAdapter(adp : Adapter) : void`
Metodo utilizzato per visualizzare una slideshow di immagini relative al POI da raggiungere

Argomenti:

- `adp : Adapter`
Collegamento tra la lista delle immagini e la view in cui esse devono essere mostrate

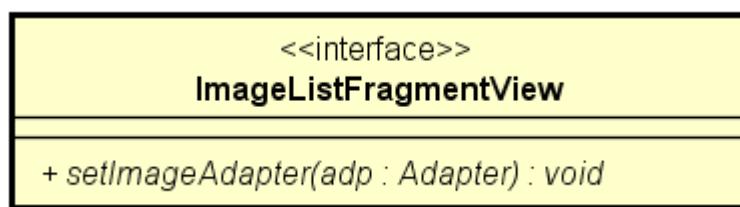
3.4.147 view::ImageListView

Figura 164: Interfaccia ImageListView

Nome: `ImageListView`;

Tipo: Interfaccia;

Componenti delle librerie utilizzate:

- `android.widget.AdapterView.OnItemClickListener`(Android).

Visibilità: public;

Utilizzo: Utilizzata per visualizzare le immagini di un arco necessarie per passare dal nodo di partenza a quello di arrivo;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente le thumbnail delle immagini relative al prossimo POI;

Metodi:

- + `setImageAdapter(adp : Adapter) : void`
Metodo utilizzato per visualizzare la lista delle anteprime delle immagine relative ad un certo POI

Argomenti:

- adp : Adapter
Collegamento tra la lista delle immagini e la view in cui esse devono essere mostrate

3.4.148 view::ImageListFragmentViewImp

ImageListFragmentViewImp	
- presenter : ImageListFragment	
- listViewImages : ListView	
+ ImageListFragmentViewImp(presenter : ImageListFragment)	
+ setImageAdapter(adp : Adapter) : void	

Figura 165: Classe ImageListFragmentViewImp**Nome:** ImageListFragmentViewImp;**Tipo:** Classe;**Implementa:**

- ImageListFragmentView.

Visibilità: public;**Utilizzo:** Mantiene i riferimenti all'elemento di layout che rappresenta la lista di immagini. La lista di immagini deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso;**Descrizione:** Classe che si occupa di mostrare la lista delle anteprime delle immagini relative ad un certo POI. La UI legata a questa classe permette all'utente di accedere alle immagini relative ad un certo POI;**Attributi:**

- - listViewImages : ListView
Lista delle anteprime delle foto
- - presenter : ImageListFragment
Presenter della View

Metodi:

- + `ImageListFragmentViewImp(presenter : ImageListFragment)`
Costruttore della classe `ImageListFragmentViewImp`

Argomenti:

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

- + `setImageAdapter(adp : Adapter) : void`
Metodo utilizzato per visualizzare la lista delle anteprime delle immagine relative ad un certo POI

Argomenti:

- `adp : Adapter`
Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate

3.4.149 view::LocalMapManagerView

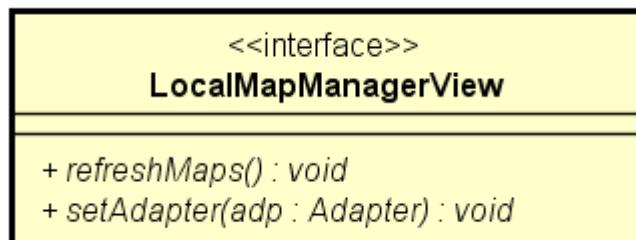


Figura 166: Interfaccia LocalMapManagerView

Nome: `LocalMapManagerView`;

Tipo: Interfaccia;

Visibilità: `public`;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente le mappe salvate nel database locale. ;

Metodi:

- + *refreshMaps()* : void

Metodo che aggiorna la lista delle mappe salvate nel database locale

- + *setAdapter(adp : Adapter)* : void

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

Argomenti:

- adp : Adapter

Collegamento tra la lista delle mappe salvate nel database locale e la view in cui esse devono essere mostrate

3.4.150 view::LocalMapManagerViewImp

LocalMapManagerViewImp	
- presenter : LocalMapActivity	
+ refreshMaps() : void	
+ setAdapter(adp : Adapter) : void	
+ LocalMapManagerViewImp(presenter : LocalMapActivity)	

Figura 167: Classe LocalMapManagerViewImp

Nome: LocalMapManagerViewImp;

Tipo: Classe;

Implementa:

- LocalMapManagerView.

Visibilità: public;

Utilizzo: La lista delle mappe deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso.;

Descrizione: Classe che si occupa di mostrare le mappe degli edifici salvate nel database locale. La UI legata a questa classe permette all'utente di accedere alle funzionalità di aggiornamento e rimozione di una certa mappa;

Attributi:

- - presenter : LocalMapActivity
Presenter della View

Metodi:

- + LocalMapManagerViewImp(presenter : LocalMapActivity)
Costruttore della classe LocalMapManagerViewImp

Argomenti:

- presenter : LocalMapActivity
Presenter della View che viene creata

- + refreshMaps() : void
Metodo che aggiorna la lista delle mappe salvate nel database locale
- + setAdapter(adp : Adapter) : void
Metodo utilizzato per visualizzare la lista delle mappe salvate nel database locale

Argomenti:

- adp : Adapter
Collegamento tra la lista delle mappe salvate nel database locale e la view in cui esse devono essere mostrate

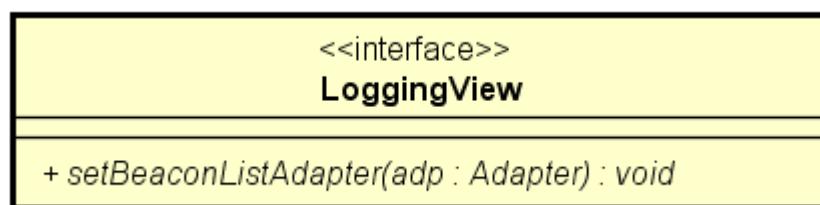
3.4.151 view::LoggingView

Figura 168: Interfaccia LoggingView

Nome: LoggingView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente gli identificativi dei beacon rilevati;

Metodi:

- + *setBeaconListAdapter(adp : Adapter) : void*
Metodo utilizzato per visualizzare la lista dei beacon rilevati

Argomenti:

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

3.4.152 view::LoggingViewImp

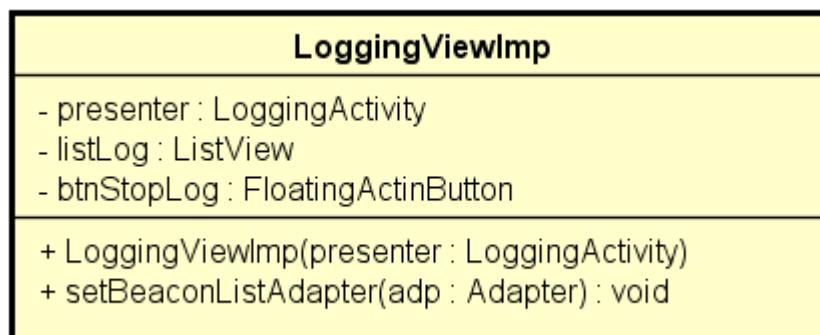


Figura 169: Classe LoggingViewImp

Nome: LoggingViewImp;

Tipo: Classe;

Implementa:

- LoggingView.

Visibilità: public;

Utilizzo: Mantiene i riferimenti agli elementi di layout che compongono la UI, tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad

un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso.;

Descrizione: Classe che permette di visualizzare il log in corso e di salvarlo.;

Attributi:

- - btnStopLog : FloatingActionButton
Bottone per interrompere un log in corso
- - listLog : ListView
Lista di log salvati sul dispositivo
- - presenter : LoggingActivity
Presenter della View

Metodi:

- + LoggingViewImp(presenter : LoggingActivity)
Costruttore della classe LoggingViewImp

Argomenti:

- presenter : LoggingActivity
Presenter della View che viene creata

- + setBeaconListAdapter(adp : Adapter) : void
Metodo utilizzato per visualizzare la lista dei beacon rilevati

Argomenti:

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

3.4.153 view::LogInformationView

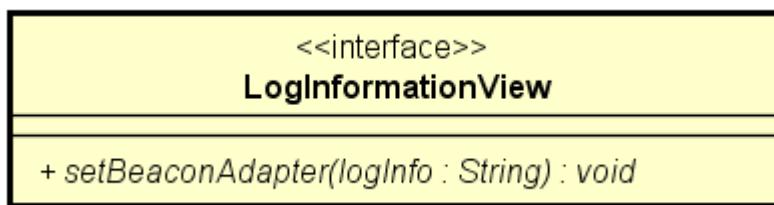


Figura 170: Interfaccia LogInformationView

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

3.4.154 view::LogInformationViewImp

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

Figura 171: 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`
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

3.4.155 view::MainDeveloperView



Figura 172: Interfaccia MainDeveloperView

Nome: MainDeveloperView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente la lista dei log salvati e disponibili ad essere consultati;

Metodi:

- + *setLogsAdapter(adp : Adapter) : void*

Metodo utilizzato per visualizzare la lista di tutti i log salvati in locale

Argomenti:

- *adp : Adapter*

Collegamento tra la lista dei log e la view in cui essi devono essere mostrati

3.4.156 view::MainDeveloperViewImp

MainDeveloperViewImp
- presenter : MainDeveloperActivity - logStartBtn : FloatingActionButton - logList : ListView
+ MainDeveloperViewImp(presenter : MainDeveloperActivity) + setLogsAdapter(adp : Adapter) : void

Figura 173: Classe MainDeveloperViewImp

Nome: MainDeveloperViewImp;

Tipo: Classe;

Implementa:

- MainDeveloperView.

Visibilità: public;

Utilizzo: Mantiene i riferimenti agli elementi del layout, tramite questi riferimenti è possibile invocare i metodi propri dei vari elementi di layout. Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso. Per lo stesso motivo, la lista dei log salvati deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener;

Descrizione: Classe che si occupa di mostrare la lista di log salvati. La UI legata a questa classe permette all'utente di: accedere alle informazioni di un certo log o avviare un nuovo log;

Attributi:

- - logList : ListView
View che mostra la lista dei log
- - logStartBtn : FloatingActionButton
Bottone che permette di attivare un log
- - presenter : MainDeveloperActivity
Presenter della View

Metodi:

- + MainDeveloperViewImp(presenter : MainDeveloperActivity)
Costruttore della classe MainDeveloperViewImp

Argomenti:

- presenter : MainDeveloperActivity
Presenter della View che viene creata

- + setLogsAdapter(adp : Adapter) : void
Metodo utilizzato per visualizzare la lista di tutti i log salvati in locale

Argomenti:

- adp : Adapter
Collegamento tra la lista dei log e la view in cui essi devono essere mostrati

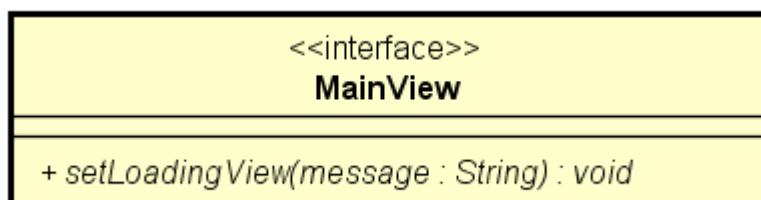
3.4.157 view::MainView

Figura 174: Interfaccia MainView

Nome: MainView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI relativa alla schermata di avvio dell'applicazione;

Metodi:

- + *setLoadingView(message : String) : void*

Metodo utilizzato per visualizzare la schermata di caricamento dell'applicazione

Argomenti:

- message : String
Messaggio di caricamento

3.4.158 view::MainViewImp

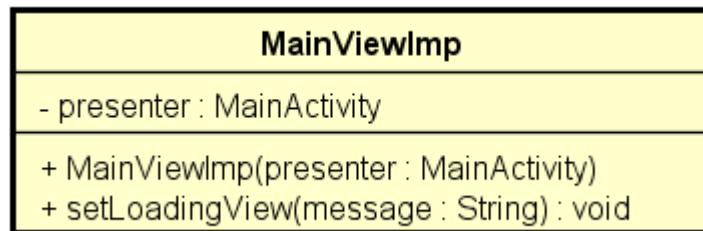


Figura 175: Classe MainViewImp

Nome: MainViewImp;

Tipo: Classe;

Implementa:

- MainView.

Visibilità: public;

Utilizzo: Questa classe viene utilizzata per eseguire tutte le operazioni CPU-intensive necessarie all'avvio dell'applicazione.;

Descrizione: Classe che si occupa di mostrare la schermata di caricamento dell'applicazione, eventuali operazioni CPU-intensive possono essere eseguite qui.;

Attributi:

- - presenter : MainActivity
Presenter della View

Metodi:

- + MainViewImp(presenter : MainActivity)

Costruttore della classe MainViewImp

Argomenti:

- presenter : MainActivity

Presenter della View che viene creata

- + setLoadingView(message : String) : void

Metodo utilizzato per visualizzare la schermata di caricamento dell'applicazione

Argomenti:

- message : String

Messaggio di caricamento

3.4.159 view::MapDownloaderView

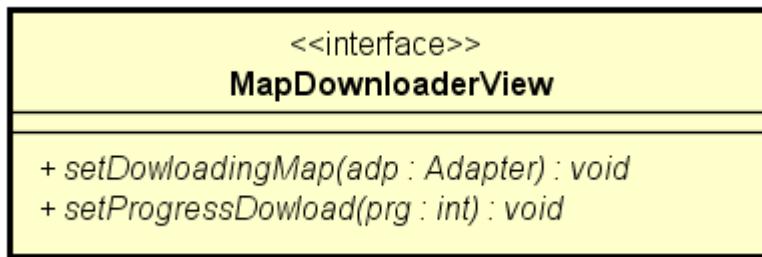


Figura 176: Interfaccia MapDownloaderView

Nome: MapDownloaderView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI che mostra il progresso durante il download della mappa di un edificio dal server;

Metodi:

- + *setDowloadingMap(adp : Adapter) : void*

Metodo utilizzato per visualizzare la mappa che si sta scaricando

Argomenti:

– adp : Adapter

Collegamento tra la mappa che si sta scaricando e la view
in cui essa deve essere mostrata

- + *setProgressDowload(prg : int) : void*

Metodo utilizzato per visualizzare il progresso nel download di una
mappa

Argomenti:

– prg : int

Attuale progresso del download

3.4.160 view::MapDownloaderViewImp

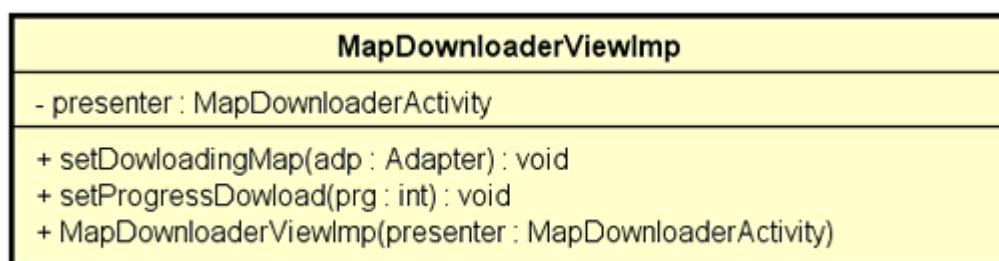


Figura 177: Classe MapDownloaderViewImp

Nome: MapDownloaderViewImp;

Tipo: Classe;

Implementa:

- MapDownloaderView.

Visibilità: public;

Utilizzo: I dettagli della mappa in download ed il progresso del download
stesso possono essere mostrati utilizzando i metodi esposti da questa
classe;

Descrizione: Classe che si occupa di mostrare il progresso del download di
una mappa.;

Attributi:

- - **presenter** : MapDownloaderActivity
Presenter della View

Metodi:

- + **MapDownloaderViewImp(presenter : MapDownloaderActivity)**
Costruttore della classe MapDownloaderViewImp

Argomenti:

- **presenter** : MapDownloaderActivity
Presenter della View che viene creata

- + **setDownloadingMap(adp : Adapter) : void**

Metodo utilizzato per visualizzare la mappa che si sta scaricando

Argomenti:

- **adp** : Adapter
Collegamento tra la mappa che si sta scaricando e la view
in cui essa deve essere mostrata

- + **setProgressDownload(prg : int) : void**

Metodo utilizzato per visualizzare il progresso nel download di una mappa

Argomenti:

- **prg** : int
Attuale progresso del download

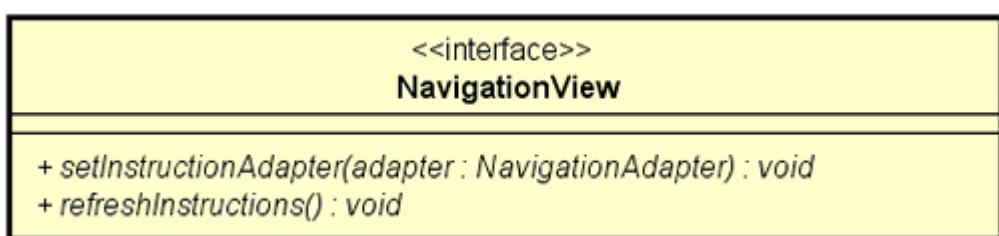
3.4.161 view::NavigationView

Figura 178: Interfaccia NavigationView

Nome: NavigationView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente le istruzioni di navigazione per raggiungere una certa destinazione;

Metodi:

- + *refreshInstructions()* : void

Metodo utilizzato per aggiornare la lista di istruzioni

- + *setInstructionAdapter(adapter : NavigationAdapter)* : void

Metodo utilizzato per visualizzare la lista di istruzioni utili a raggiungere un certo POI

Argomenti:

- adapter : NavigationAdapter

Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate

3.4.162 view::NavigationViewImp

NavigationViewImp	
- presenter : NavigationActivity	
- instructionAdapter : Adapter	
+ setInstructionAdapter(adapter : NavigationAdapter) : void	
+ refreshInstructions() : void	
+ NavigationViewImp(presenter : NavigationActivity)	

Figura 179: Classe NavigationViewImp

Nome: NavigationViewImp;

Tipo: Classe;

Implementa:

- NavigationView.

Visibilità: public;

Utilizzo: La lista di istruzioni deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista possa reagire alla pressione su di esso. Per recuperare un riferimento alla lista è sufficiente utilizzare la classe R.id di Android;

Descrizione: Classe che si occupa di mostrare la lista di istruzioni di navigazione utili per raggiungere un determinato POI. La UI legata a questa classe permette all'utente di accedere alle descrizioni dettagliate delle varie istruzioni;

Attributi:

- - `instructionAdapter : Adapter`
Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate
- - `presenter : NavigationActivity`
Presenter della View

Metodi:

- + `NavigationViewImp(presenter : NavigationActivity)`
Costruttore della classe NavigationViewImp

Argomenti:

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

- + `refreshInstructions() : void`
Metodo utilizzato per aggiornare la lista di istruzioni
- + `setInstructionAdapter(adapter : NavigationAdapter) : void`
Metodo utilizzato per visualizzare la lista di istruzioni utili a raggiungere un certo POI

Argomenti:

- `adapter : NavigationAdapter`
Collegamento tra la lista delle informazioni e la view in cui esse devono essere mostrate

3.4.163 view::NearbyPoiView

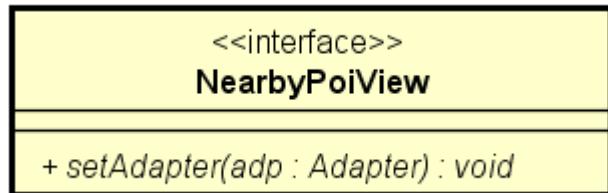


Figura 180: Interfaccia NearbyPoiView

Nome: NearbyPoiView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente la lista dei POI nelle vicinanze dell'utente;

Metodi:

- + *setAdapter(adp : Adapter) : void*

Metodo utilizzato per visualizzare tutti i POI nelle circostanze dell'utente

Argomenti:

- adp : Adapter

Collegamento tra la lista dei POI circostanti l'utente e la view in cui essi devono essere mostrati

3.4.164 view::NearbyPoiViewImp

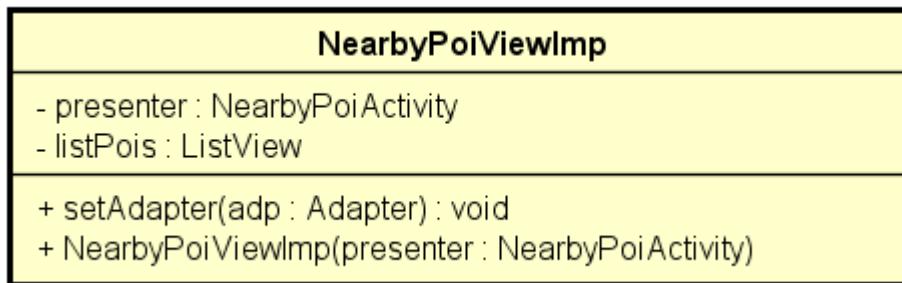


Figura 181: Classe NearbyPoiViewImp

Nome: NearbyPoiViewImp;

Tipo: Classe;

Implementa:

- `NearbyPoiView`.

Visibilità: public;

Utilizzo: Mantiene i riferimenti all'elemento di layout che rappresenta la lista di POI. La lista di POI deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso;

Descrizione: Classe che si occupa di mostrare i POI situati nelle vicinanze dell'utente. La UI legata a questa classe permette all'utente di accedere alle informazioni di un certo POI;

Attributi:

- `- listPois : ListView`
View che mostra la lista di POI nelle vicinanze dell'utente
- `- presenter : NearbyPoiActivity`
Presenter della View

Metodi:

- `+ NearbyPoiViewImp(presenter : NearbyPoiActivity)`
Costruttore della classe NearbyPoiViewImp

Argomenti:

- presenter : NearbyPoiActivity
Presenter della View che viene creata
- + setAdapter(adp : Adapter) : void
Metodo utilizzato per visualizzare tutti i POI nelle circostanze dell'utente

Argomenti:

- adp : Adapter
Collegamento tra la lista dei POI circostanti l'utente e la view in cui essi devono essere mostrati

3.4.165 view::PoiCategoryView



Figura 182: Interfaccia PoiCategoryView

Nome: PoiCategoryView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

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

Metodi:

- + setPoiListAdapter(adapter : ListAdapter) : void
Metodo utilizzato per visualizzare tutti i POI appartenenti ad una certa categoria

Argomenti:

- **adapter : ListAdapter**
Collegamento tra la lista delle categorie dei POI e la view in cui essi devono essere mostrati

3.4.166 view::PoiCategoryViewImp

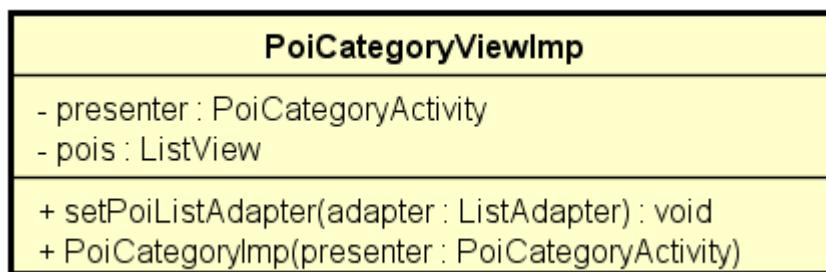


Figura 183: Classe PoiCategoryViewImp

Nome: PoiCategoryViewImp;

Tipo: Classe;

Implementa:

- `PoiCategoryView`.

Visibilità: public;

Utilizzo: Mantiene i riferimenti all'elemento di layout che rappresenta la lista di POI. La lista di POI deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso;

Descrizione: Classe che si occupa di mostrare la lista dei POI relativi ad una certa categoria. La UI legata a questa classe permette all'utente di accedere alle informazioni di un certo POI appartenente alla categoria.;

Attributi:

- `-pois : ListView`
View che permette di visualizzare la lista delle categorie di POI
- `-presenter : PoiCategoryActivity`
Presenter della View

Metodi:

- + PoiCategoryViewImp(presenter : PoiCategoryActivity)
Costruttore della classe PoiCategoryViewImp

Argomenti:

- presenter : PoiCategoryActivity
Presenter della View che viene creata

- + setPoiListAdapter(adapter : ListAdapter) : void
Metodo utilizzato per visualizzare tutti i POI appartenenti ad una certa categoria

Argomenti:

- adapter : ListAdapter
Collegamento tra la lista delle categorie dei POI e la view in cui essi devono essere mostrati

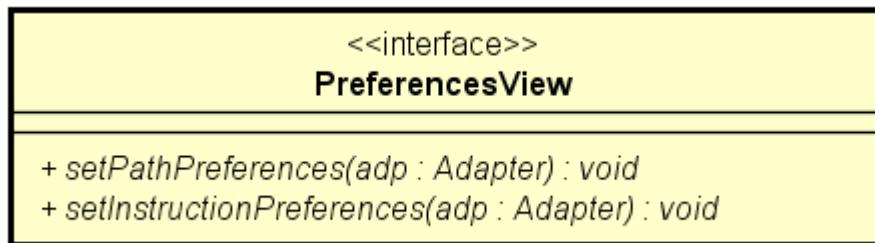
3.4.167 view::PreferencesView

Figura 184: Interfaccia PreferencesView

Nome: PreferencesView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente le preferenze dell'utente rispetto al percorso consigliato e alla modalità di fruizione delle istruzioni di navigazione;

Metodi:

- + *setInstructionPreferences(adp : Adapter) : void*
Metodo utilizzato per visualizzare le preferenze dell'utente riguardo la fruizione delle istruzioni di navigazione

Argomenti:

- adp : Adapter

Collegamento tra le preferenze riguardanti la fruizione delle istruzioni di navigazione e la view in cui esse devono essere mostrate

- + *setPathPreferences(adp : Adapter) : void*

Metodo utilizzato per visualizzare le preferenze dell'utente relative al percorso proposto

Argomenti:

- adp : Adapter

Collegamento tra le preferenze riguardanti le preferenze del percorso di navigazione e la view in cui esse devono essere mostrate

3.4.168 view::PreferencesViewImp

PreferencesViewImp
- presenter : PreferencesActivity
+ PreferencesViewImp(presenter : PreferencesActivity)
+ setPathPreferences(adp : Adapter) : void
+ setInstructionPreferences(adp : Adapter) : void

Figura 185: Classe PreferencesViewImp

Nome: PreferencesViewImp;

Tipo: Classe;

Implementa:

- PreferencesView.

Visibilità: public;

Utilizzo: Ogni bottone presente deve essere legato ad un oggetto anonimo di tipo View.OnClickListener, in modo da poter reagire alla pressione su di esso. Per recuperare un riferimento alla lista è sufficiente utilizzare la classe R.id di Android;

Descrizione: Classe che si occupa di mostrare la UI utile alla modifica delle preferenze dell'utente;

Attributi:

- - presenter : PreferencesActivity
Presenter della View

Metodi:

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

Argomenti:

- presenter : PreferencesActivity
Presenter della View che viene creata
- presenter : PreferencesActivity
Presenter della View che viene creata

- + setInstructionPreferences(adp : Adapter) : void
Metodo utilizzato per visualizzare le preferenze dell'utente riguardo la fruizione delle istruzioni di navigazione

Argomenti:

- adp : Adapter
Collegamento tra le preferenze riguardanti la fruizione delle istruzioni di navigazione e la view in cui esse devono essere mostrate

- + setPathPreferences(adp : Adapter) : void
Metodo utilizzato per visualizzare le preferenze dell'utente relative al percorso proposto

Argomenti:

- adp : Adapter
Collegamento tra le preferenze riguardanti le preferenze del percorso di navigazione e la view in cui esse devono essere mostrate

3.4.169 view::RemoteMapView

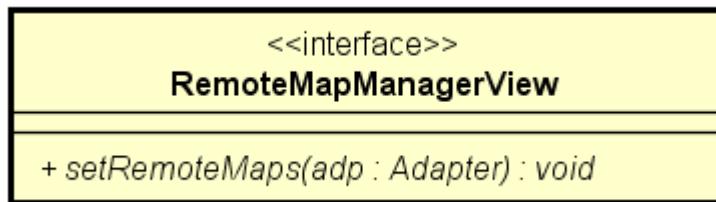


Figura 186: Interfaccia RemoteMapView

Nome: RemoteMapView;

Tipo: Interfaccia;

Visibilità: public;

Utilizzo: È utilizzata per rendere indipendente l'implementazione della UI dalle modalità attraverso le quali è possibile aggiornarla;

Descrizione: Interfaccia che espone i metodi per aggiornare la UI contenente le mappe delle quali è possibile effettuare il download dal server;

Metodi:

- + *setRemoteMaps(adp : Adapter) : void*

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

Argomenti:

- adp : Adapter

Collegamento tra la lista delle mappe che è possibile scaricare e la view in cui esse devono essere mostrate

3.4.170 view::RemoteMapViewImp

RemoteMapViewImp
- presenter : RemoteMapManagerActivity
+ setRemoteMaps(adp : Adapter) : void
+ RemoteMapViewImp(presenter : RemoteMapManagerActivity)

Figura 187: Classe RemoteMapViewImp

Nome: RemoteMapViewImp;

Tipo: Classe;

Implementa:

- RemoteMapView.

Visibilità: public;

Utilizzo: La lista delle mappe deve essere legata ad un oggetto anonimo di tipo AdapterView.OnItemClickListener, in modo che ogni item della lista di POI possa reagire alla pressione su di esso.;

Descrizione: Classe che si occupa di mostrare le mappe degli edifici disponibili al download. La UI legata a questa classe permette all'utente di accedere alle funzionalità di download di una certa mappa;

Attributi:

- - presenter : RemoteMapManagerActivity
Presenter della View

Metodi:

- + RemoteMapViewImp(presenter : RemoteMapManagerActivity)
Costruttore della classe RemoteMapViewImp

Argomenti:

- presenter : RemoteMapManagerActivity
Presenter della View che viene creata

- + setRemoteMaps(adp : Adapter) : void
Metodo utilizzato per visualizzare le mappe che è possibile scaricare da un server remoto

Argomenti:

– adp : Adapter

Collegamento tra la lista delle mappe che è possibile scaricare e la view in cui esse devono essere mostrate

4 Schema base di dati

Di seguito viene presentata lo schema in UML della base di dati implementata nell'applicativo con SQLite e gestito dal componente **DataManager** e implementata nel server remoto. Lo schema illustra le relazioni tra le entità che costituiscono il grafo rappresentato l'edificio di interesse. Si fa notare che la base di dati non memorizza separatamente gli elementi che compongono i grafici.

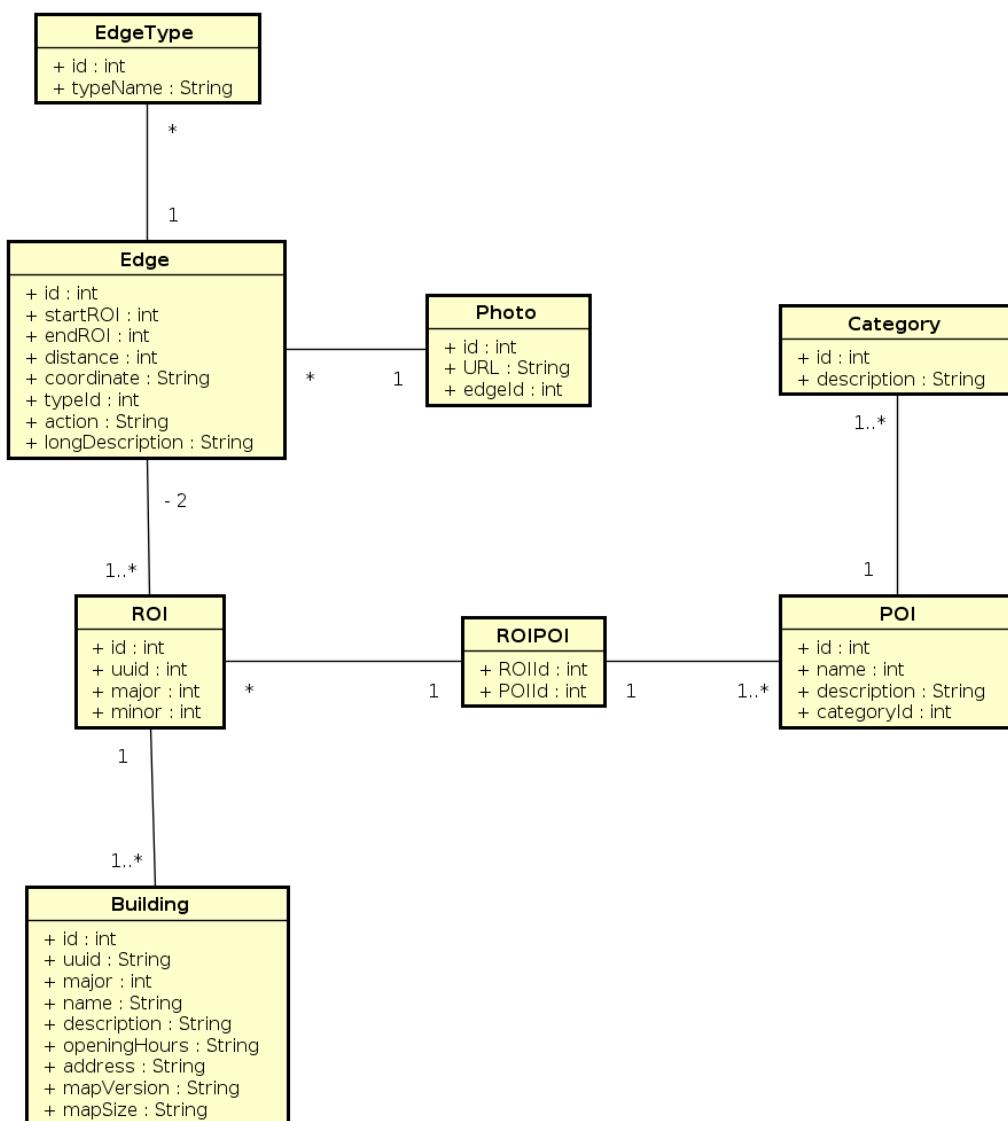


Figura 188: Schema UML - base di dati

5 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.

5.1 Avvio Service per il rilevamento beacon

Il diagramma in figura 189 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.

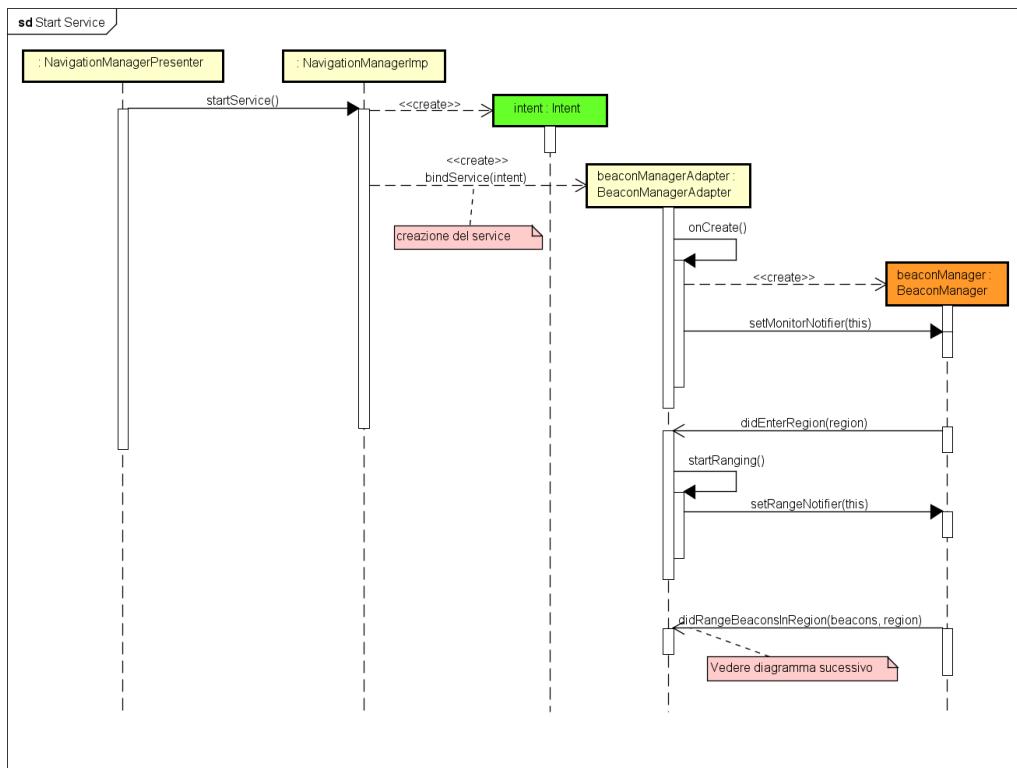


Figura 189: Diagramma di sequenza - Avvio di un service_g per il rilevamento beacon

5.2 Elaborazione beacon rilevati e comunicazione broadcast

Il diagramma in figura 190 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.

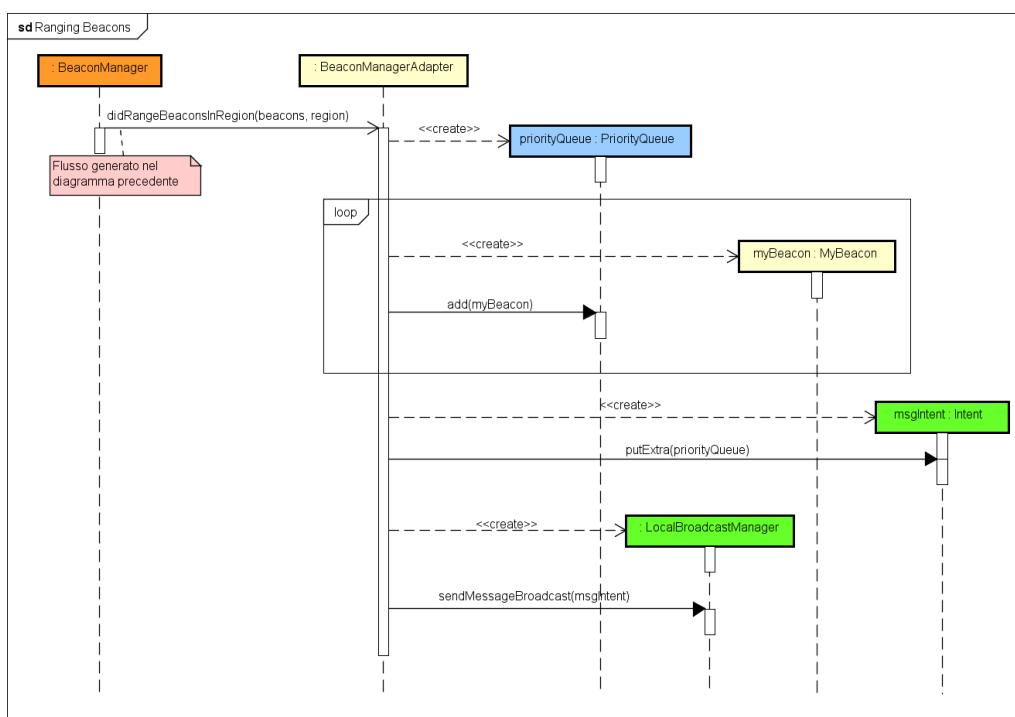


Figura 190: Diagramma di sequenza - Elaborazione beacon rilevati e comunicazione broadcast

5.3 Avvio navigazione

Il diagramma in figura 191 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 `NavigatorManagerImp` 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 scompatterà e le restituirà alla view e quindi all'utente.

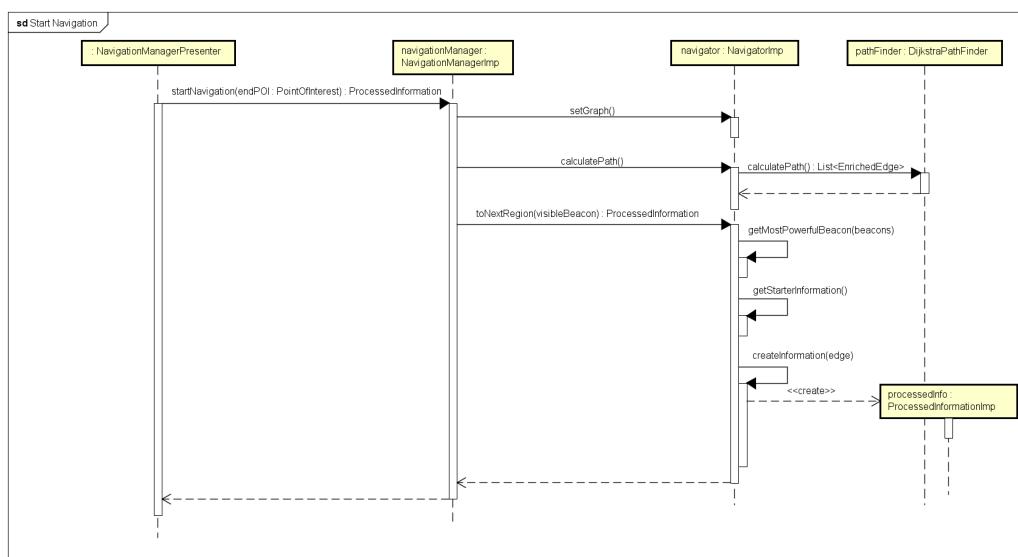


Figura 191: Diagramma di sequenza - Avvio navigazione

6 Tracciamento

6.1 Tracciamento Classi-Requisiti

6.2 Requisiti-Classi

A Diagrammi riassuntivi package significativi

Di seguito sono riportati tutti i package dell'applicativo ad eccezione della `view`, essendo applicato il pattern MVP, per chiarire la relazione tra le componenti e le classi al suo interno. Per chiarezza ed esigenza di spazio le classi rappresentate all'interno dei package sono senza metodi e attributi.

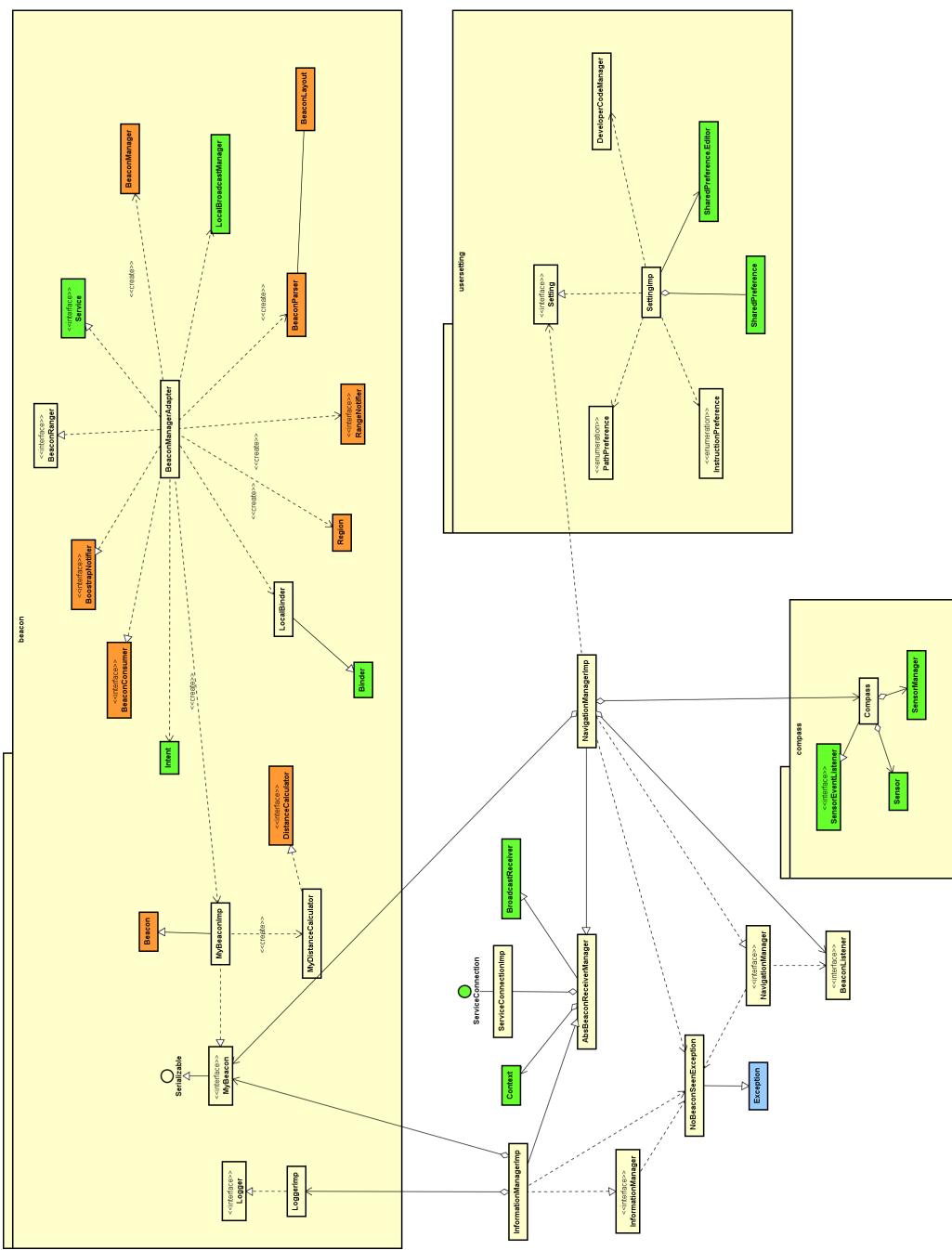
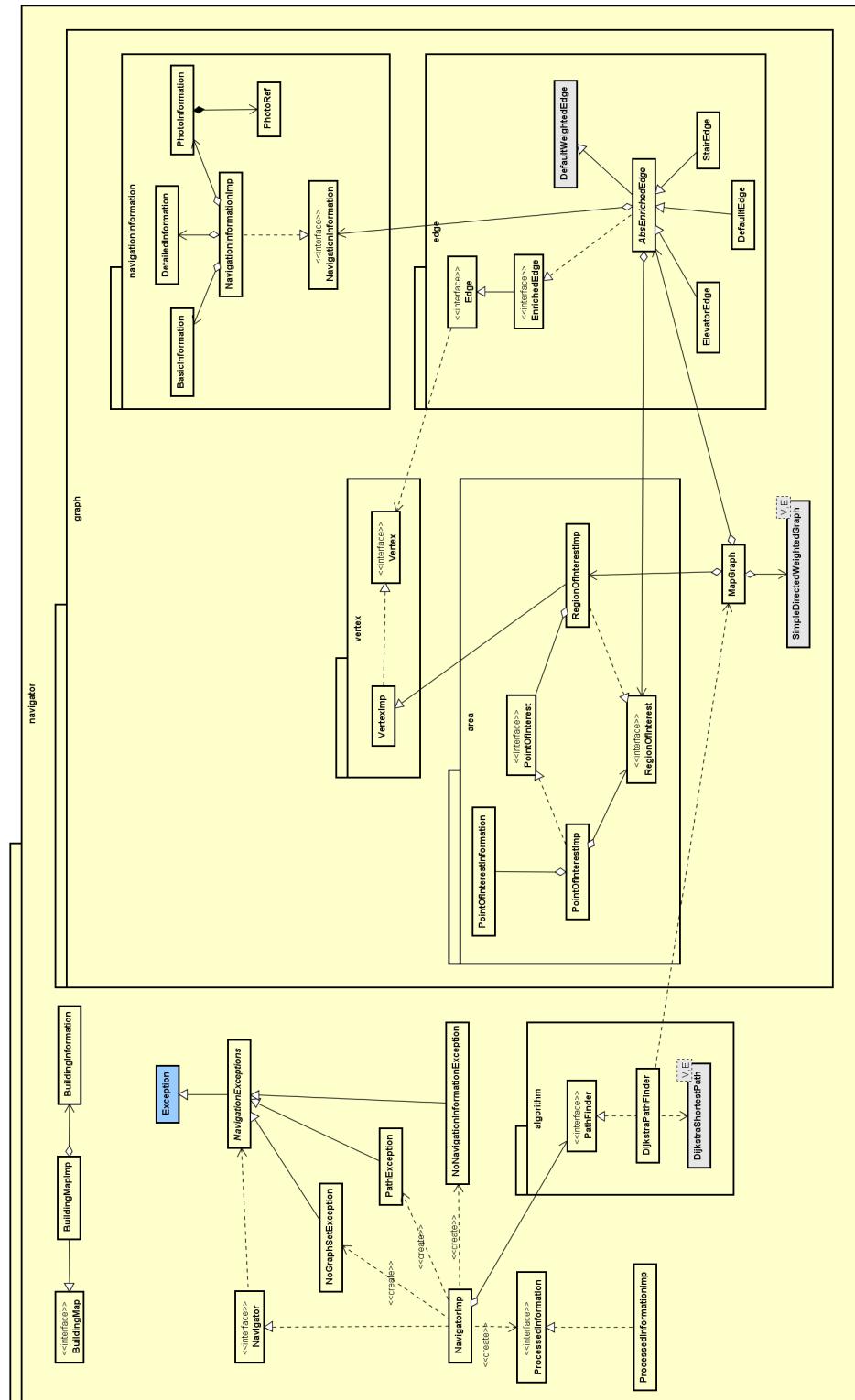


Figura 192: Diagramma delle classi - model


Figura 193: Diagramma delle classi - model::navigator

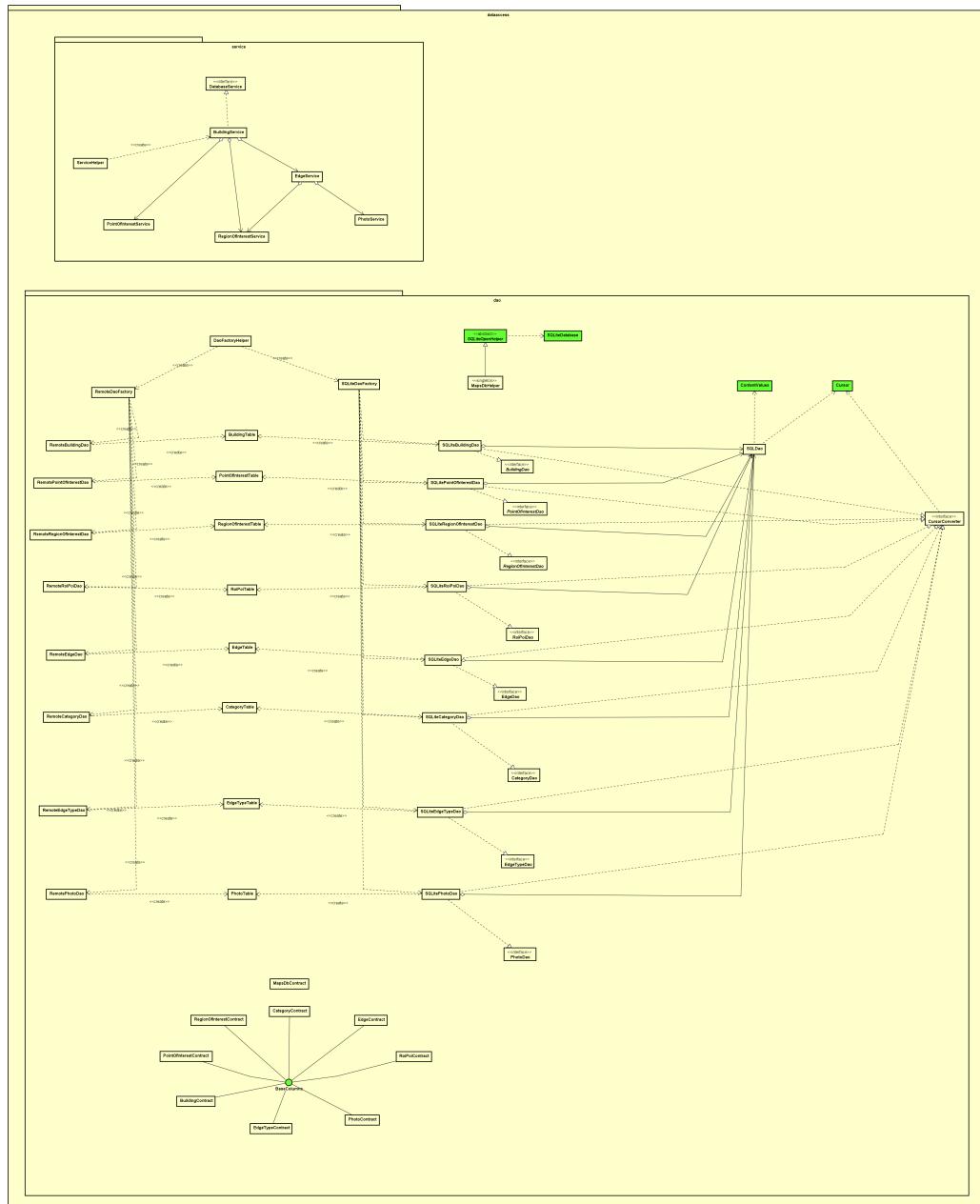


Figura 194: Diagramma delle classi - model::dataaccess

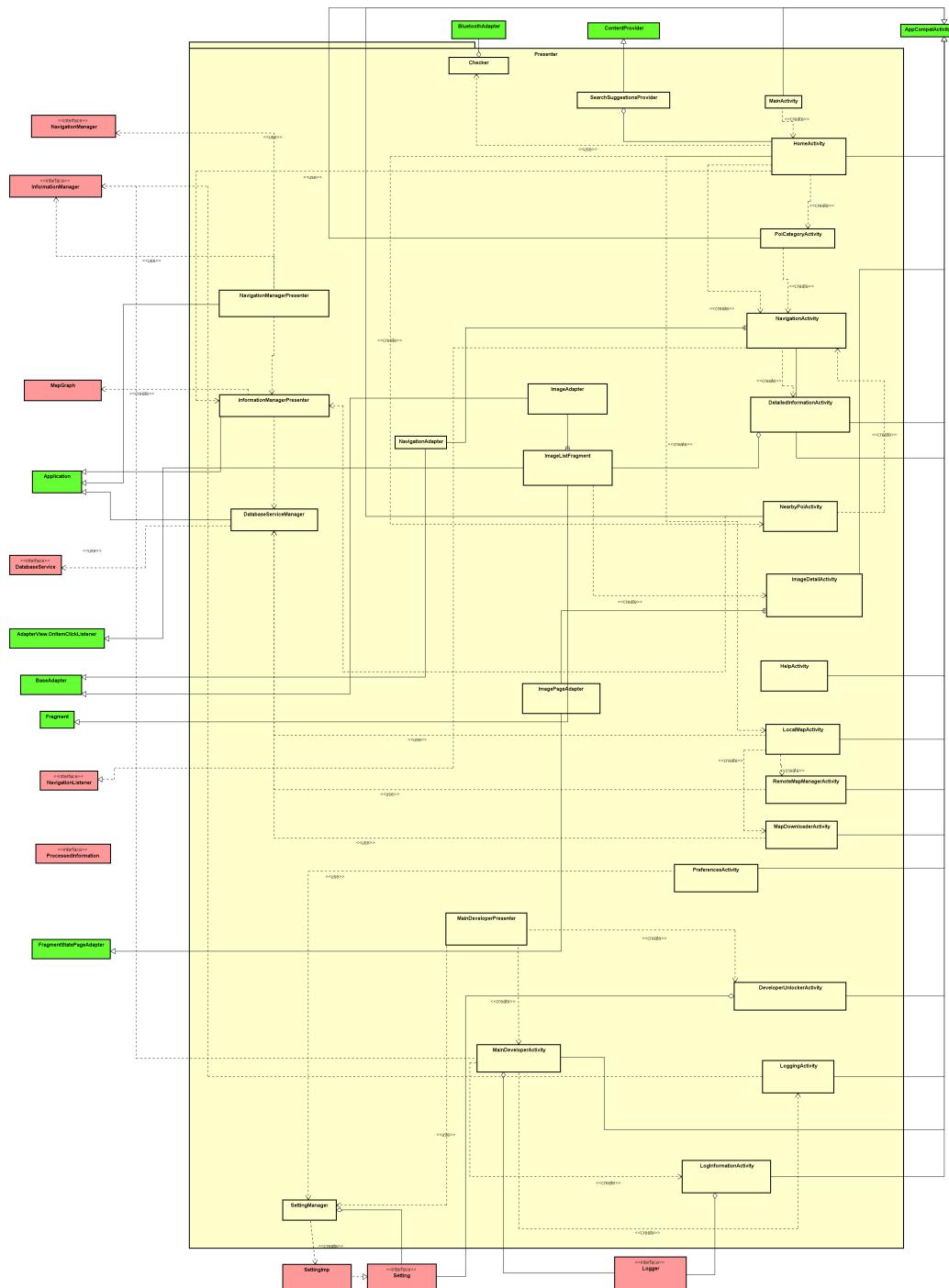


Figura 195: Diagramma delle classi - presenter