



## Definizione di Prodotto

### Informazioni sul documento

---

<b>Nome file:</b>	definizione_di_prodotto_v2.0.pdf
<b>Versione:</b>	2.0
<b>Data creazione:</b>	2015-05-04
<b>Data ultima modifica:</b>	2015-06-16
<b>Lista di distribuzione:</b>	Prof. Tullio Vardanega Prof. Riccardo Cardin Si14 S.p.a.
<b>Redattori:</b>	Andrea Boscolo Nata Andrea Costa Tiziano Longo
<b>Approvato da:</b>	Mario Garavello
<b>Verificatori:</b>	Jacopo Cavallarin
<b>Stato:</b>	Preliminare
<b>Uso:</b>	Esterno

---

## Storia delle modifiche

Versione	Descrizione intervento	Autore	Ruolo	Data
2.0	Approvato	Mario Garavello	Responsabile	2015-06-16
1.12	Verifica	Jacopo Cavallarin	Verificatore	2015-06-16
1.11	Aggiornata sezione <b>Schema base di dati</b>	Andrea Costa	Progettista	2015-06-15
1.10	Sistemati errori riscontrati	Davide Canal	Progettista	2015-06-14
1.9	Aggiornata sezione <b>shike::web::view</b>	Andrea Boscolo Nata	Progettista	2015-06-13
1.8	Conclusa sezione <b>shike::app::model::sync</b>	Andrea Costa	Progettista	2015-06-12
1.7	Iniziata sezione <b>shike::app::helper</b>	Andrea Boscolo Nata	Progettista	2015-06-11
1.6	Iniziata sezione <b>shike::app::model::sync</b>	Andrea Costa	Progettista	2015-06-11
1.5	Aggiornata sezione <b>shike::web::helper</b>	Andrea Boscolo Nata	Progettista	2015-06-09
1.4	Stesura sezione <b>shike::app::helper</b>	Andrea Costa	Progettista	2015-06-09
1.3	Stesura sezioni Sync: DAO, Controller e Model	Tiziano Longo	Progettista	2015-06-09
1.2	Corretti errori grammaticali	Tiziano Longo	Progettista	2015-06-09
1.1	Aggiornati diagrammi <b>243</b> e <b>244</b> secondo il modello ER	Andrea Costa	Progettista	2015-06-08
1.0	Approvato	Andrea Costa	Responsabile	2015-05-26
0.19	Verifica	Tiziano Longo	Verificatore	2015-05-25
0.18	Aggiornato <b>Tracciamento</b>	Jacopo Cavallarin	Progettista	2015-05-22
0.17	Sistemati errori <b>shike::web::model::dao</b>	Davide Canal	Progettista	2015-05-22
0.16	Aggiornati grafici uml	Jacopo Cavallarin	Progettista	2015-05-22
0.15	Stesura sezione <b>Diagrammi di sequenza</b>	Andrea Boscolo Nata	Progettista	2015-05-22
0.14	Aggiornato <b>Tracciamento</b>	Davide Canal	Progettista	2015-05-21
0.13	Conclusa sezione <b>shike::web::model</b>	Davide Canal	Progettista	2015-05-20
0.12	Sistemati errori <b>shike::web::controller</b>	Andrea Boscolo Nata	Progettista	2015-05-20
0.11	Stesura sezione <b>shike::app::view</b>	Jacopo Cavallarin	Progettista	2015-05-18
0.10	Sistemati errori riscontrati	Davide Canal	Progettista	2015-05-18
0.9	Conclusa sezione <b>shike::app::model</b>	Jacopo Cavallarin	Progettista	2015-05-15
0.8	Conclusa sezione <b>shike::web::controller</b>	Andrea Boscolo Nata	Progettista	2015-05-14
0.7	Iniziata sezione <b>shike::web::controller</b>	Andrea Boscolo Nata	Progettista	2015-05-12
0.6	Iniziata sezione <b>shike::app::model</b>	Jacopo Cavallarin	Progettista	2015-05-11
0.5	Stesura sezione <b>shike::web::helper</b>	Davide Canal	Progettista	2015-05-08

0.4	Stesura sezione <b>Schema base di dati</b>	Jacopo Cavallarin	Progettista	2015-05-07
0.3	Stesura sezione <b>shike::web::model::dao</b>	Davide Canal	Progettista	2015-05-07
0.2	Definizione della sezione <b>Standard di progetto</b>	Andrea Boscolo Nata	Progettista	2015-05-06
0.1	Scheletro documento	Andrea Boscolo Nata	Progettista	2015-05-04

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento . . . . .	1
1.2	Scopo del prodotto . . . . .	1
1.3	Glossario . . . . .	1
1.4	Riferimenti . . . . .	1
1.4.1	Normativi . . . . .	1
1.4.2	Informativi . . . . .	1
<b>2</b>	<b>Standard di progetto</b>	<b>3</b>
2.1	Standard di progettazione architetturale . . . . .	3
2.2	Standard di documentazione del codice . . . . .	3
2.3	Standard di denominazione di entità e relazioni . . . . .	3
2.4	Standard di programmazione . . . . .	3
2.5	Strumenti di lavoro . . . . .	3
<b>3</b>	<b>Specifiche dei componenti</b>	<b>4</b>
3.1	Metodo e formalismo di specifica . . . . .	4
3.2	shike . . . . .	4
3.2.1	Informazioni sul package . . . . .	4
3.3	shike::app . . . . .	5
3.3.1	Informazioni sul package . . . . .	5
3.4	shike::app::presenter . . . . .	6
3.4.1	Informazioni sul package . . . . .	6
3.4.2	Classi . . . . .	7
3.4.2.1	shike::app::presenter:: <b>NavigationActivity</b> . . . . .	7
3.4.2.2	shike::app::presenter:: <b>WeatherPresenter</b> . . . . .	9
3.4.2.3	shike::app::presenter:: <b>DashboardPresenter</b> . . . . .	11
3.4.2.4	shike::app::presenter:: <b>CompassPresenter</b> . . . . .	13
3.4.2.5	shike::app::presenter:: <b>SyncPresenter</b> . . . . .	16
3.4.2.6	shike::app::presenter:: <b>PoiPresenter</b> . . . . .	16
3.4.2.7	shike::app::presenter:: <b>SettingPresenter</b> . . . . .	18
3.4.2.8	shike::app::presenter:: <b>DashboardPresenter::TimerThread</b> . . . . .	19
3.4.2.9	shike::app::presenter:: <b>HomeActivity</b> . . . . .	20
3.4.2.10	shike::app::presenter:: <b>HomePresenter</b> . . . . .	21
3.4.2.11	shike::app::presenter:: <b>NavigationActivity::MyPagerAdapter</b> . . . . .	22
3.4.2.12	shike::app::presenter:: <b>SyncPresenter</b> . . . . .	23
3.4.2.13	shike::app::presenter:: <b>DashboardPresenter::ButtonThread</b> . . . . .	24
3.4.2.14	shike::app::presenter:: <b>HelpNumberPresenter</b> . . . . .	25
3.5	shike::app::model . . . . .	27
3.5.1	Informazioni sul package . . . . .	27
3.6	shike::app::model::weather . . . . .	29
3.6.1	Informazioni sul package . . . . .	29
3.6.2	Classi . . . . .	30
3.6.2.1	shike::app::model::weather:: <b>Weather</b> . . . . .	30
3.6.2.2	shike::app::model::weather:: <b>TrackWeather</b> . . . . .	31
3.6.2.3	shike::app::model::weather:: <b>Weather::ForecastType</b> . . . . .	32
3.6.2.4	shike::app::model::weather:: <b>Weather::Wind</b> . . . . .	33
3.7	shike::app::model::session . . . . .	34
3.7.1	Informazioni sul package . . . . .	34
3.8	shike::app::model::session::track . . . . .	35
3.8.1	Informazioni sul package . . . . .	35

3.8.2	Classi . . . . .	36
3.8.2.1	shike::app::model::session::track:: <b>Poi</b> . . . . .	36
3.8.2.2	shike::app::model::session::track:: <b>Track</b> . . . . .	37
3.8.2.3	shike::app::model::session::track:: <b>VirtualTrack</b> . . . . .	38
3.8.2.4	shike::app::model::session::track:: <b>RecordedTrack</b> . . . . .	39
3.8.2.5	shike::app::model::session::track:: <b>Poi</b> :: <b>PoiType</b> . . . . .	40
3.9	shike::app::model::session::performance . . . . .	41
3.9.1	Informazioni sul package . . . . .	41
3.9.2	Classi . . . . .	42
3.9.2.1	shike::app::model::session::performance:: <b>Performance</b> . . . . .	42
3.10	shike::app::model::user . . . . .	44
3.10.1	Informazioni sul package . . . . .	44
3.10.2	Classi . . . . .	45
3.10.2.1	shike::app::model::user:: <b>Account</b> . . . . .	45
3.10.2.2	shike::app::model::user:: <b>HelpNumber</b> . . . . .	47
3.10.2.3	shike::app::model::user:: <b>Account</b> :: <b>Gender</b> . . . . .	48
3.11	shike::app::model::dao . . . . .	49
3.11.1	Informazioni sul package . . . . .	49
3.11.2	Classi . . . . .	50
3.11.2.1	shike::app::model::dao:: <b>GeneralDao</b> . . . . .	50
3.11.2.2	shike::app::model::dao:: <b>GeneralDaoImpl</b> . . . . .	51
3.12	shike::app::model::dao::account . . . . .	51
3.12.1	Informazioni sul package . . . . .	51
3.12.2	Classi . . . . .	52
3.12.2.1	shike::app::model::dao::account:: <b>AccountDao</b> . . . . .	52
3.12.2.2	shike::app::model::dao::account:: <b>AccountDaoImpl</b> . . . . .	53
3.13	shike::app::model::dao::helpnumber . . . . .	54
3.13.1	Informazioni sul package . . . . .	54
3.13.2	Classi . . . . .	55
3.13.2.1	shike::app::model::dao::helpnumber:: <b>HelpNumberDao</b> . . . . .	55
3.13.2.2	shike::app::model::dao::helpnumber:: <b>HelpNumberDaoImpl</b> . . . . .	55
3.14	shike::app::model::dao::performance . . . . .	56
3.14.1	Informazioni sul package . . . . .	56
3.14.2	Classi . . . . .	57
3.14.2.1	shike::app::model::dao::performance:: <b>PerformanceDaoImpl</b> . . . . .	57
3.14.2.2	shike::app::model::dao::performance:: <b>PerformanceDao</b> . . . . .	58
3.14.2.3	shike::app::model::dao::performance:: <b>PerformanceDaoImpl</b> . . . . .	59
3.14.2.4	shike::app::model::dao::performance:: <b>PerformanceDao</b> . . . . .	59
3.15	shike::app::model::dao::poi . . . . .	61
3.15.1	Informazioni sul package . . . . .	61
3.15.2	Classi . . . . .	62
3.15.2.1	shike::app::model::dao::poi:: <b>PoiDaoImpl</b> . . . . .	62
3.15.2.2	shike::app::model::dao::poi:: <b>PoiDao</b> . . . . .	62
3.16	shike::app::model::dao::track . . . . .	64
3.16.1	Informazioni sul package . . . . .	64
3.16.2	Classi . . . . .	65
3.16.2.1	shike::app::model::dao::track:: <b>VirtualTrackDaoImpl</b> . . . . .	65
3.16.2.2	shike::app::model::dao::track:: <b>VirtualTrackDao</b> . . . . .	66
3.17	shike::app::model::dao::weather . . . . .	67
3.17.1	Informazioni sul package . . . . .	67
3.17.2	Classi . . . . .	67
3.17.2.1	shike::app::model::dao::weather:: <b>TrackWeatherDao</b> . . . . .	67

3.17.2.2 shike::app::model::dao::weather:: <b>TrackWeatherDaoImpl</b>	68
3.18 shike::app::model::dao::db	69
3.18.1 Informazioni sul package	69
3.18.2 Classi	70
3.18.2.1 shike::app::model::dao::db:: <b>AccountTable</b>	70
3.18.2.2 shike::app::model::dao::db:: <b>GendersTable</b>	71
3.18.2.3 shike::app::model::dao::db:: <b>DbContract</b>	72
3.18.2.4 shike::app::model::dao::db:: <b>DbUtil</b>	73
3.18.2.5 shike::app::model::dao::db:: <b>HelpNumbersTable</b>	74
3.18.2.6 shike::app::model::dao::db:: <b>RecordedTracksTable</b>	75
3.18.2.7 shike::app::model::dao::db:: <b>RecordedTrackLocationsTable</b>	76
3.18.2.8 shike::app::model::dao::db:: <b>VirtualTracksTable</b>	76
3.18.2.9 shike::app::model::dao::db:: <b>VirtualTrackLocationsTable</b>	77
3.18.2.10 shike::app::model::dao::db:: <b>WeatherForecastsTable</b>	78
3.18.2.11 shike::app::model::dao::db:: <b>ForecastTypesTable</b>	79
3.18.2.12 shike::app::model::dao::db:: <b>PoisTable</b>	80
3.18.2.13 shike::app::model::dao::db:: <b>PoiTypesTable</b>	81
3.19 shike::app::model::service	82
3.19.1 Informazioni sul package	82
3.19.2 Classi	83
3.19.2.1 shike::app::model::service:: <b>AccountService</b>	83
3.19.2.2 shike::app::model::service:: <b>GeneralService</b>	84
3.19.2.3 shike::app::model::service:: <b>HelpNumberService</b>	84
3.19.2.4 shike::app::model::service:: <b>PerformanceService</b>	85
3.19.2.5 shike::app::model::service:: <b>PoiService</b>	86
3.19.2.6 shike::app::model::service:: <b>TrackWeatherService</b>	87
3.19.2.7 shike::app::model::service:: <b>VirtualTrackService</b>	88
3.20 shike::app::model::sync	89
3.20.1 Informazioni sul package	89
3.20.2 Classi	90
3.20.2.1 shike::app::model::sync:: <b>SyncDataApp</b>	90
3.20.2.2 shike::app::model::sync:: <b>AccountLinkData</b>	91
3.20.2.3 shike::app::model::sync:: <b>SyncDataWeb</b>	92
3.20.2.4 shike::app::model::sync:: <b>SyncDataWeb::Error</b>	93
3.21 shike::app::view	94
3.21.1 Informazioni sul package	94
3.22 shike::app::view::config	95
3.22.1 Informazioni sul package	95
3.22.2 Classi	95
3.22.2.1 shike::app::view::config:: <b>ConfigView</b>	95
3.22.2.2 shike::app::view::config:: <b>SyncNumberDialog</b>	96
3.23 shike::app::view::helpnumber	97
3.23.1 Informazioni sul package	97
3.23.2 Classi	98
3.23.2.1 shike::app::view::helpnumber:: <b>HelpNumberSelectionView</b>	98
3.23.2.2 shike::app::view::helpnumber:: <b>HelpNumberViewAdapter</b>	99
3.24 shike::app::view::home	100
3.24.1 Informazioni sul package	100
3.24.2 Classi	101
3.24.2.1 shike::app::view::home:: <b>TrackView</b>	101
3.24.2.2 shike::app::view::home:: <b>HomeView</b>	101
3.24.2.3 shike::app::view::home:: <b>ListViewAdapter</b>	102

3.24.2.4 shike::app::view::home:: <b>TrackSelectionView</b>	103
3.25 shike::app::view::poi	105
3.25.1 Informazioni sul package	105
3.25.2 Classi	105
3.25.2.1 shike::app::view::poi:: <b>PoiSelectionView</b>	105
3.25.2.2 shike::app::view::poi:: <b>PoiView</b>	106
3.25.2.3 shike::app::view::poi:: <b>PoiViewAdapter</b>	107
3.26 shike::app::view::session	109
3.26.1 Informazioni sul package	109
3.26.2 Classi	109
3.26.2.1 shike::app::view::session:: <b>DashboardView</b>	109
3.26.2.2 shike::app::view::session:: <b>Compass View</b>	111
3.26.2.3 shike::app::view::session:: <b>WidgetSelectionView</b>	111
3.26.2.4 shike::app::view::session:: <b>DialogConfirmSaveTrack</b>	112
3.27 shike::app::view::weather	113
3.27.1 Informazioni sul package	113
3.27.2 Classi	113
3.27.2.1 shike::app::view::weather:: <b>WeatherView</b>	113
3.27.2.2 shike::app::view::weather:: <b>WeatherSelectionView</b>	115
3.27.2.3 shike::app::view::weather:: <b>WeatherViewAdapter</b>	116
3.28 shike::app::helper	117
3.28.1 Informazioni sul package	117
3.28.2 Classi	118
3.28.2.1 shike::app::helper:: <b>WebConnectionManager</b>	118
3.28.2.2 shike::app::helper:: <b>WebConnectionManager::Result</b>	119
3.29 shike::app::helper::json	120
3.29.1 Informazioni sul package	120
3.29.2 Classi	120
3.29.2.1 shike::app::helper::json:: <b>JsonDateSerializer</b>	120
3.29.2.2 shike::app::helper::json:: <b>JsonDateDeserializer</b>	121
3.29.2.3 shike::app::helper::json:: <b>JsonConverter</b>	122
3.29.2.4 shike::app::helper::json:: <b>JsonLocationSerializer</b>	123
3.29.2.5 shike::app::helper::json:: <b>JsonLocationDeserializer</b>	123
3.30 shike::web	124
3.30.1 Informazioni sul package	124
3.31 shike::web::controller	125
3.31.1 Informazioni sul package	125
3.31.2 Classi	125
3.31.2.1 shike::web::controller:: <b>MainController</b>	125
3.31.2.2 shike::web::controller:: <b>RecordedTrackController</b>	126
3.31.2.3 shike::web::controller:: <b>PoiController</b>	128
3.31.2.4 shike::web::controller:: <b>AuthController</b>	129
3.31.2.5 shike::web::controller:: <b>AccountController</b>	131
3.31.2.6 shike::web::controller:: <b>SyncController</b>	133
3.31.2.7 shike::web::controller:: <b>HelpNumberController</b>	133
3.31.2.8 shike::web::controller:: <b>VirtualTrackController</b>	135
3.32 shike::web::view	137
3.32.1 Informazioni sul package	137
3.32.2 Classi	138
3.32.2.1 shike::web::view:: <b>Home</b>	138
3.33 shike::web::view::account	138
3.33.1 Informazioni sul package	138

3.33.2	Classi . . . . .	139
3.33.2.1	shike::web::view::account::ResetPassword . . . . .	139
3.33.2.2	shike::web::view::account::Login . . . . .	139
3.33.2.3	shike::web::view::account::AddCommonAccount . . . . .	140
3.33.2.4	shike::web::view::account::EditPassword . . . . .	140
3.34	shike::web::view::account::user . . . . .	141
3.34.1	Informazioni sul package . . . . .	141
3.34.2	Classi . . . . .	141
3.34.2.1	shike::web::view::account::user::Dashboard . . . . .	141
3.34.2.2	shike::web::view::account::user::AddHelpNumber . . . . .	142
3.34.2.3	shike::web::view::account::user::EditHelpNumber . . . . .	142
3.34.2.4	shike::web::view::account::user::EditUser . . . . .	143
3.34.2.5	shike::web::view::account::user::ListHelpNumbers . . . . .	143
3.35	shike::web::view::account::admin . . . . .	144
3.35.1	Informazioni sul package . . . . .	144
3.35.2	Classi . . . . .	145
3.35.2.1	shike::web::view::account::admin::Dashboard . . . . .	145
3.35.2.2	shike::web::view::account::admin::ListCommonAccount . . . . .	145
3.36	shike::web::view::track . . . . .	146
3.36.1	Informazioni sul package . . . . .	146
3.36.2	Classi . . . . .	147
3.36.2.1	shike::web::view::track::Stats . . . . .	147
3.37	shike::web::view::track::poi . . . . .	148
3.37.1	Informazioni sul package . . . . .	148
3.37.2	Classi . . . . .	148
3.37.2.1	shike::web::view::track::poi::DetailsPoi . . . . .	148
3.37.2.2	shike::web::view::track::poi::AddPoi . . . . .	149
3.37.2.3	shike::web::view::track::poi::ListPoi . . . . .	149
3.38	shike::web::view::track::virtual . . . . .	150
3.38.1	Informazioni sul package . . . . .	150
3.38.2	Classi . . . . .	150
3.38.2.1	shike::web::view::track::virtual::ListTrack . . . . .	150
3.38.2.2	shike::web::view::track::virtual::EditTrack . . . . .	151
3.38.2.3	shike::web::view::track::virtual::DetailsTrack . . . . .	151
3.39	shike::web::view::track::recorded . . . . .	152
3.39.1	Informazioni sul package . . . . .	152
3.39.2	Classi . . . . .	153
3.39.2.1	shike::web::view::track::recorded::DetailsTrack . . . . .	153
3.39.2.2	shike::web::view::track::recorded::ShareTrack . . . . .	153
3.39.2.3	shike::web::view::track::recorded::ListTrack . . . . .	154
3.40	shike::web::view::inc . . . . .	154
3.40.1	Informazioni sul package . . . . .	154
3.40.2	Classi . . . . .	155
3.40.2.1	shike::web::view::inc::Footer . . . . .	155
3.40.2.2	shike::web::view::inc::NotFound . . . . .	155
3.40.2.3	shike::web::view::inc::Header . . . . .	155
3.40.2.4	shike::web::view::inc::Menu . . . . .	156
3.40.2.5	shike::web::view::inc::Message . . . . .	156
3.41	shike::web::helper . . . . .	157
3.41.1	Informazioni sul package . . . . .	157
3.41.2	Classi . . . . .	157
3.41.2.1	shike::web::helper::MailManager . . . . .	157

3.42 shike::web::helper::sharing . . . . .	158
3.42.1 Informazioni sul package . . . . .	158
3.42.2 Classi . . . . .	159
3.42.2.1 shike::web::helper::sharing::Share . . . . .	159
3.42.2.2 shike::web::helper::sharing::ShareFacebook . . . . .	159
3.42.2.3 shike::web::helper::sharing::ShareTwitter . . . . .	160
3.43 shike::web::helper::validator . . . . .	161
3.43.1 Informazioni sul package . . . . .	161
3.43.2 Classi . . . . .	161
3.43.2.1 shike::web::helper::validator::HelpNumberValidator . . . . .	161
3.43.2.2 shike::web::helper::validator::CommonAccountValidator . . . . .	162
3.44 shike::web::model . . . . .	163
3.44.1 Informazioni sul package . . . . .	163
3.44.2 Classi . . . . .	164
3.44.2.1 shike::web::model::Message::Alert . . . . .	164
3.44.2.2 shike::web::model::Message . . . . .	165
3.45 shike::web::model::weather . . . . .	167
3.45.1 Informazioni sul package . . . . .	167
3.45.2 Classi . . . . .	168
3.45.2.1 shike::web::model::weather::TrackWeather . . . . .	168
3.45.2.2 shike::web::model::weather::Weather . . . . .	169
3.45.2.3 shike::web::model::weather::Weather::Wind . . . . .	171
3.45.2.4 shike::web::model::weather::Weather::ForecastType . . . . .	172
3.46 shike::web::model::user . . . . .	174
3.46.1 Informazioni sul package . . . . .	174
3.46.2 Classi . . . . .	175
3.46.2.1 shike::web::model::user::Account . . . . .	175
3.46.2.2 shike::web::model::user::CommonAccount . . . . .	177
3.46.2.3 shike::web::model::user::HelpNumber . . . . .	179
3.46.2.4 shike::web::model::user::CommonAccount::Gender . . . . .	180
3.47 shike::web::model::session . . . . .	181
3.47.1 Informazioni sul package . . . . .	181
3.48 shike::web::model::session::track . . . . .	182
3.48.1 Informazioni sul package . . . . .	182
3.48.2 Classi . . . . .	183
3.48.2.1 shike::web::model::session::track::Track . . . . .	183
3.48.2.2 shike::web::model::session::track::RecordedTrack . . . . .	186
3.48.2.3 shike::web::model::session::track::VirtualTrack . . . . .	187
3.48.2.4 shike::web::model::session::track::Poi . . . . .	189
3.48.2.5 shike::web::model::session::track::Location . . . . .	191
3.48.2.6 shike::web::model::session::track::Poi::PoiType . . . . .	193
3.48.2.7 shike::web::model::session::track::VirtualTrack::Level . . . . .	194
3.49 shike::web::model::session::performance . . . . .	196
3.49.1 Informazioni sul package . . . . .	196
3.49.2 Classi . . . . .	197
3.49.2.1 shike::web::model::session::performance::Performance . . . . .	197
3.49.2.2 shike::web::model::session::performance::Stats . . . . .	200
3.50 shike::web::model::dao . . . . .	202
3.50.1 Informazioni sul package . . . . .	202
3.51 shike::web::model::dao::account . . . . .	204
3.51.1 Informazioni sul package . . . . .	204
3.51.2 Classi . . . . .	205

3.51.2.1	shike::web::model::dao::account:: <b>CommonAccountMapper</b>	205
3.51.2.2	shike::web::model::dao::account:: <b>AccountDao</b>	206
3.51.2.3	shike::web::model::dao::account:: <b>AccountMapper</b>	208
3.51.2.4	shike::web::model::dao::account:: <b>AccountDaoImpl</b>	209
3.52	shike::web::model::dao::helpnumber	210
3.52.1	Informazioni sul package	210
3.52.2	Classi	210
3.52.2.1	shike::web::model::dao::helpnumber:: <b>HelpNumberDao</b>	210
3.52.2.2	shike::web::model::dao::helpnumber:: <b>HelpNumberMapper</b>	211
3.52.2.3	shike::web::model::dao::helpnumber:: <b>HelpNumberDaoImpl</b>	212
3.53	shike::web::model::dao::performance	212
3.53.1	Informazioni sul package	212
3.54	shike::web::model::dao::poi	213
3.54.1	Informazioni sul package	213
3.54.2	Classi	214
3.54.2.1	shike::web::model::dao::poi:: <b>PoiDao</b>	214
3.54.2.2	shike::web::model::dao::poi:: <b>PoiMapper</b>	215
3.54.2.3	shike::web::model::dao::poi:: <b>PoiDaoImpl</b>	215
3.55	shike::web::model::dao::recordtrack	217
3.55.1	Informazioni sul package	217
3.55.2	Classi	218
3.55.2.1	shike::web::model::dao::recordtrack:: <b>RecordedTrackDao</b>	218
3.55.2.2	shike::web::model::dao::recordtrack:: <b>PerformanceMapper</b>	219
3.55.2.3	shike::web::model::dao::recordtrack:: <b>RecordedTrackDaoImpl</b>	220
3.55.2.4	shike::web::model::dao::recordtrack:: <b>RecordedTrackMapper</b>	220
3.55.2.5	shike::web::model::dao::recordtrack:: <b>RecordedTrackLocationMapper</b>	221
3.55.2.6	shike::web::model::dao::recordtrack:: <b>StatsMapper</b>	221
3.56	shike::web::model::dao::weather	222
3.56.1	Informazioni sul package	222
3.56.2	Classi	223
3.56.2.1	shike::web::model::dao::weather:: <b>WeatherDao</b>	223
3.56.2.2	shike::web::model::dao::weather:: <b>WeatherMapper</b>	223
3.56.2.3	shike::web::model::dao::weather:: <b>WeatherDaoImpl</b>	224
3.57	shike::web::model::dao::virtualtrack	225
3.57.1	Informazioni sul package	225
3.57.2	Classi	226
3.57.2.1	shike::web::model::dao::virtualtrack:: <b>VirtualTrackDao</b>	226
3.57.2.2	shike::web::model::dao::virtualtrack:: <b>VirtualTrackMapper</b>	227
3.57.2.3	shike::web::model::dao::virtualtrack:: <b>VirtualTrackLocationMapper</b>	227
3.57.2.4	shike::web::model::dao::virtualtrack:: <b>VirtualTrackDaoImpl</b>	228
3.58	shike::web::model::service	229
3.58.1	Informazioni sul package	229
3.58.2	Classi	230
3.58.2.1	shike::web::model::service:: <b>VirtualTrackService</b>	230
3.58.2.2	shike::web::model::service:: <b>WeatherService</b>	231
3.58.2.3	shike::web::model::service:: <b>AccountService</b>	232
3.58.2.4	shike::web::model::service:: <b>HelpNumberService</b>	234
3.58.2.5	shike::web::model::service:: <b>PoiService</b>	236
3.58.2.6	shike::web::model::service:: <b>RecordedTrackService</b>	237
3.59	shike::web::model::sync	239
3.59.1	Informazioni sul package	239
3.59.2	Classi	240

3.59.2.1	shike::web::model::sync:: <b>AccountLinkData</b>	240
3.59.2.2	shike::web::model::sync:: <b>SyncDataApp</b>	241
3.59.2.3	shike::web::model::sync:: <b>SyncDataWeb</b>	242
3.59.2.4	shike::web::model::sync:: <b>SyncDataWeb::Error</b>	243
<b>4</b>	<b>Schema base di dati</b>	<b>244</b>
4.1	Parte applicazione	244
4.1.1	Lista delle tabelle	244
4.1.1.1	account	244
4.1.1.2	genders	245
4.1.1.3	helpnumbers	245
4.1.1.4	recordedtracks	245
4.1.1.5	recordedtracklocations	246
4.1.1.6	virtualtracks	246
4.1.1.7	virtualtracklocations	246
4.1.1.8	weatherforecasts	247
4.1.1.9	forecasttypes	247
4.1.1.10	pois	247
4.1.1.11	poitypes	248
4.2	Parte web	248
4.2.1	Lista delle tabelle	249
4.2.1.1	accounts	249
4.2.1.2	commonaccounts	249
4.2.1.3	adminaccounts	249
4.2.1.4	genders	249
4.2.1.5	helpnumbers	250
4.2.1.6	recordedtracks	250
4.2.1.7	recordedtracklocations	250
4.2.1.8	virtualtracks	251
4.2.1.9	virtualtracklocations	251
4.2.1.10	weatherforecasts	251
4.2.1.11	forecasttypes	252
4.2.1.12	stats	252
4.2.1.13	pois	252
4.2.1.14	poitypes	253
4.2.1.15	tokens	253
4.2.1.16	tracktosync	253
<b>5</b>	<b>Diagrammi di sequenza</b>	<b>254</b>
5.1	Gestione di una richiesta di registrazione	254
5.2	Gestione dei tracciati comuni a tutti gli utenti	255
5.3	Selezione percorso	256
<b>6</b>	<b>Tracciamento</b>	<b>257</b>

## Elenco delle figure

1	Diagramma di shike . . . . .	4
2	Diagramma di shike::app . . . . .	5
3	Diagramma di shike::app::presenter . . . . .	6
4	Diagramma di NavigationActivity . . . . .	7
5	Diagramma di WeatherPresenter . . . . .	9
6	Diagramma di DashboardPresenter . . . . .	11
7	Diagramma di CompassPresenter . . . . .	13
8	Diagramma di SyncPresenter . . . . .	16
9	Diagramma di PoiPresenter . . . . .	16
10	Diagramma di SettingPresenter . . . . .	18
11	Diagramma di DashboardPresenter::TimerThread . . . . .	19
12	Diagramma di HomeActivity . . . . .	20
13	Diagramma di HomePresenter . . . . .	21
14	Diagramma di NavigationActivity::MyPagerAdapter . . . . .	22
15	Diagramma di SyncPresenter . . . . .	23
16	Diagramma di DashboardPresenter::ButtonThread . . . . .	24
17	Diagramma di HelpNumberPresenter . . . . .	25
18	Diagramma di shike::app::model . . . . .	27
19	Diagramma di shike::app::model::weather . . . . .	29
20	Diagramma di Weather . . . . .	30
21	Diagramma di TrackWeather . . . . .	31
22	Diagramma di Weather::ForecastType . . . . .	32
23	Diagramma di Weather::Wind . . . . .	33
24	Diagramma di shike::app::model::session . . . . .	34
25	Diagramma di shike::app::model::session::track . . . . .	35
26	Diagramma di Poi . . . . .	36
27	Diagramma di Track . . . . .	37
28	Diagramma di VirtualTrack . . . . .	38
29	Diagramma di RecordedTrack . . . . .	39
30	Diagramma di Poi::PoiType . . . . .	40
31	Diagramma di shike::app::model::session::performance . . . . .	41
32	Diagramma di Performance . . . . .	42
33	Diagramma di shike::app::model::user . . . . .	44
34	Diagramma di Account . . . . .	45
35	Diagramma di HelpNumber . . . . .	47
36	Diagramma di Account::Gender . . . . .	48
37	Diagramma di shike::app::model::dao . . . . .	49
38	Diagramma di GeneralDao . . . . .	50
39	Diagramma di GeneralDaoImpl . . . . .	51
40	Diagramma di shike::app::model::dao::account . . . . .	51
41	Diagramma di AccountDao . . . . .	52
42	Diagramma di AccountDaoImpl . . . . .	53
43	Diagramma di shike::app::model::dao::helpnumber . . . . .	54
44	Diagramma di HelpNumberDao . . . . .	55
45	Diagramma di HelpNumberDaoImpl . . . . .	55
46	Diagramma di shike::app::model::dao::performance . . . . .	56
47	Diagramma di PerformanceDaoImpl . . . . .	57
48	Diagramma di PerformanceDao . . . . .	58
49	Diagramma di PerformanceDaoImpl . . . . .	59
50	Diagramma di PerformanceDao . . . . .	59

51	Diagramma di shike::app::model::dao::poi . . . . .	61
52	Diagramma di PoiDaoImpl . . . . .	62
53	Diagramma di PoiDao . . . . .	62
54	Diagramma di shike::app::model::dao::track . . . . .	64
55	Diagramma di VirtualTrackDaoImpl . . . . .	65
56	Diagramma di VirtualTrackDao . . . . .	66
57	Diagramma di shike::app::model::dao::weather . . . . .	67
58	Diagramma di TrackWeatherDao . . . . .	67
59	Diagramma di TrackWeatherDaoImpl . . . . .	68
60	Diagramma di shike::app::model::dao::db . . . . .	69
61	Diagramma di AccountTable . . . . .	70
62	Diagramma di GendersTable . . . . .	71
63	Diagramma di DbContract . . . . .	72
64	Diagramma di DbUtil . . . . .	73
65	Diagramma di HelpNumbersTable . . . . .	74
66	Diagramma di RecordedTracksTable . . . . .	75
67	Diagramma di RecordedTrackLocationsTable . . . . .	76
68	Diagramma di VirtualTracksTable . . . . .	76
69	Diagramma di VirtualTrackLocationsTable . . . . .	77
70	Diagramma di WeatherForecastsTable . . . . .	78
71	Diagramma di ForecastTypesTable . . . . .	79
72	Diagramma di PoisTable . . . . .	80
73	Diagramma di PoiTypesTable . . . . .	81
74	Diagramma di shike::app::model::service . . . . .	82
75	Diagramma di AccountService . . . . .	83
76	Diagramma di GeneralService . . . . .	84
77	Diagramma di HelpNumberService . . . . .	84
78	Diagramma di PerformanceService . . . . .	85
79	Diagramma di PoiService . . . . .	86
80	Diagramma di TrackWeatherService . . . . .	87
81	Diagramma di VirtualTrackService . . . . .	88
82	Diagramma di shike::app::model::sync . . . . .	89
83	Diagramma di SyncDataApp . . . . .	90
84	Diagramma di AccountLinkData . . . . .	91
85	Diagramma di SyncDataWeb . . . . .	92
86	Diagramma di SyncDataWeb::Error . . . . .	93
87	Diagramma di shike::app::view . . . . .	94
88	Diagramma di shike::app::view::config . . . . .	95
89	Diagramma di ConfigView . . . . .	95
90	Diagramma di SyncNumberDialog . . . . .	96
91	Diagramma di shike::app::view::helpnumber . . . . .	97
92	Diagramma di HelpNumberSelectionView . . . . .	98
93	Diagramma di HelpNumberViewAdapter . . . . .	99
94	Diagramma di shike::app::view::home . . . . .	100
95	Diagramma di TrackView . . . . .	101
96	Diagramma di HomeView . . . . .	101
97	Diagramma di ListViewAdapter . . . . .	102
98	Diagramma di TrackSelectionView . . . . .	103
99	Diagramma di shike::app::view::poi . . . . .	105
100	Diagramma di PoiSelectionView . . . . .	105
101	Diagramma di PoiView . . . . .	106
102	Diagramma di PoiViewAdapter . . . . .	107

103	Diagramma di shike::app::view::session . . . . .	109
104	Diagramma di DashboardView . . . . .	109
105	Diagramma di CompassView . . . . .	111
106	Diagramma di WidgetSelectionView . . . . .	111
107	Diagramma di DialogConfirmSaveTrack . . . . .	112
108	Diagramma di shike::app::view::weather . . . . .	113
109	Diagramma di WeatherView . . . . .	113
110	Diagramma di WeatherSelectionView . . . . .	115
111	Diagramma di WeatherViewAdapter . . . . .	116
112	Diagramma di shike::app::helper . . . . .	117
113	Diagramma di WebConnectionManager . . . . .	118
114	Diagramma di WebConnectionManager::Result . . . . .	119
115	Diagramma di shike::app::helper::json . . . . .	120
116	Diagramma di JsonDateSerializer . . . . .	120
117	Diagramma di JsonDateDeserializer . . . . .	121
118	Diagramma di JsonConverter . . . . .	122
119	Diagramma di JsonLocationSerializer . . . . .	123
120	Diagramma di JsonLocationDeserializer . . . . .	123
121	Diagramma di shike::web . . . . .	124
122	Diagramma di shike::web::controller . . . . .	125
123	Diagramma di MainController . . . . .	125
124	Diagramma di RecordedTrackController . . . . .	126
125	Diagramma di PoiController . . . . .	128
126	Diagramma di AuthController . . . . .	129
127	Diagramma di AccountController . . . . .	131
128	Diagramma di SyncController . . . . .	133
129	Diagramma di HelpNumberController . . . . .	133
130	Diagramma di VirtualTrackController . . . . .	135
131	Diagramma di shike::web::view . . . . .	137
132	Diagramma di Home . . . . .	138
133	Diagramma di shike::web::view::account . . . . .	138
134	Diagramma di ResetPassword . . . . .	139
135	Diagramma di Login . . . . .	139
136	Diagramma di AddCommonAccount . . . . .	140
137	Diagramma di EditPassword . . . . .	140
138	Diagramma di shike::web::view::account::user . . . . .	141
139	Diagramma di Dashboard . . . . .	141
140	Diagramma di AddHelpNumber . . . . .	142
141	Diagramma di EditHelpNumber . . . . .	142
142	Diagramma di EditUser . . . . .	143
143	Diagramma di ListHelpNumbers . . . . .	143
144	Diagramma di shike::web::view::account::admin . . . . .	144
145	Diagramma di Dashboard . . . . .	145
146	Diagramma di ListCommonAccount . . . . .	145
147	Diagramma di shike::web::view::track . . . . .	146
148	Diagramma di Stats . . . . .	147
149	Diagramma di shike::web::view::track::poi . . . . .	148
150	Diagramma di DetailsPoi . . . . .	148
151	Diagramma di AddPoi . . . . .	149
152	Diagramma di ListPoi . . . . .	149
153	Diagramma di shike::web::view::track::virtual . . . . .	150
154	Diagramma di ListTrack . . . . .	150

155	Diagramma di EditTrack . . . . .	151
156	Diagramma di DetailsTrack . . . . .	151
157	Diagramma di shike::web::view::track::recorded . . . . .	152
158	Diagramma di DetailsTrack . . . . .	153
159	Diagramma di ShareTrack . . . . .	153
160	Diagramma di ListTrack . . . . .	154
161	Diagramma di shike::web::view::inc . . . . .	154
162	Diagramma di Footer . . . . .	155
163	Diagramma di NotFound . . . . .	155
164	Diagramma di Header . . . . .	155
165	Diagramma di Menu . . . . .	156
166	Diagramma di Message . . . . .	156
167	Diagramma di shike::web::helper . . . . .	157
168	Diagramma di MailManager . . . . .	157
169	Diagramma di shike::web::helper::sharing . . . . .	158
170	Diagramma di Share . . . . .	159
171	Diagramma di ShareFacebook . . . . .	159
172	Diagramma di ShareTwitter . . . . .	160
173	Diagramma di shike::web::helper::validator . . . . .	161
174	Diagramma di HelpNumberValidator . . . . .	161
175	Diagramma di CommonAccountValidator . . . . .	162
176	Diagramma di shike::web::model . . . . .	163
177	Diagramma di Message::Alert . . . . .	164
178	Diagramma di Message . . . . .	165
179	Diagramma di shike::web::model::weather . . . . .	167
180	Diagramma di TrackWeather . . . . .	168
181	Diagramma di Weather . . . . .	169
182	Diagramma di Weather::Wind . . . . .	171
183	Diagramma di Weather::ForecastType . . . . .	172
184	Diagramma di shike::web::model::user . . . . .	174
185	Diagramma di Account . . . . .	175
186	Diagramma di CommonAccount . . . . .	177
187	Diagramma di HelpNumber . . . . .	179
188	Diagramma di CommonAccount::Gender . . . . .	180
189	Diagramma di shike::web::model::session . . . . .	181
190	Diagramma di shike::web::model::session::track . . . . .	182
191	Diagramma di Track . . . . .	183
192	Diagramma di RecordedTrack . . . . .	186
193	Diagramma di VirtualTrack . . . . .	187
194	Diagramma di Poi . . . . .	189
195	Diagramma di Location . . . . .	191
196	Diagramma di Poi::PoiType . . . . .	193
197	Diagramma di VirtualTrack::Level . . . . .	194
198	Diagramma di shike::web::model::session::performance . . . . .	196
199	Diagramma di Performance . . . . .	197
200	Diagramma di Stats . . . . .	200
201	Diagramma di shike::web::model::dao . . . . .	202
202	Diagramma di shike::web::model::dao::account . . . . .	204
203	Diagramma di CommonAccountMapper . . . . .	205
204	Diagramma di AccountDao . . . . .	206
205	Diagramma di AccountMapper . . . . .	208
206	Diagramma di AccountDaoImpl . . . . .	209

207	Diagramma di shike::web::model::dao::helpnumber . . . . .	210
208	Diagramma di HelpNumberDao . . . . .	210
209	Diagramma di HelpNumberMapper . . . . .	211
210	Diagramma di HelpNumberDaoImpl . . . . .	212
211	Diagramma di shike::web::model::dao::poi . . . . .	213
212	Diagramma di PoiDao . . . . .	214
213	Diagramma di PoiMapper . . . . .	215
214	Diagramma di PoiDaoImpl . . . . .	215
215	Diagramma di shike::web::model::dao::recordtrack . . . . .	217
216	Diagramma di RecordedTrackDao . . . . .	218
217	Diagramma di PerformanceMapper . . . . .	219
218	Diagramma di RecordedTrackDaoImpl . . . . .	220
219	Diagramma di RecordedTrackMapper . . . . .	220
220	Diagramma di RecordedTrackLocationMapper . . . . .	221
221	Diagramma di StatsMapper . . . . .	221
222	Diagramma di shike::web::model::dao::weather . . . . .	222
223	Diagramma di WeatherDao . . . . .	223
224	Diagramma di WeatherMapper . . . . .	223
225	Diagramma di WeatherDaoImpl . . . . .	224
226	Diagramma di shike::web::model::dao::virtualtrack . . . . .	225
227	Diagramma di VirtualTrackDao . . . . .	226
228	Diagramma di VirtualTrackMapper . . . . .	227
229	Diagramma di VirtualTrackLocationMapper . . . . .	227
230	Diagramma di VirtualTrackDaoImpl . . . . .	228
231	Diagramma di shike::web::model::service . . . . .	229
232	Diagramma di VirtualTrackService . . . . .	230
233	Diagramma di WeatherService . . . . .	231
234	Diagramma di AccountService . . . . .	232
235	Diagramma di HelpNumberService . . . . .	234
236	Diagramma di PoiService . . . . .	236
237	Diagramma di RecordedTrackService . . . . .	237
238	Diagramma di shike::web::model::sync . . . . .	239
239	Diagramma di AccountLinkData . . . . .	240
240	Diagramma di SyncDataApp . . . . .	241
241	Diagramma di SyncDataWeb . . . . .	242
242	Diagramma di SyncDataWeb::Error . . . . .	243
243	Schema logico della base di dati - Parte applicazione . . . . .	244
244	Schema logico della base di dati - Parte web . . . . .	248
245	Gestione di una richiesta di registrazione . . . . .	254
246	Gestione dei tracciati comuni a tutti gli utenti . . . . .	255
247	Gestisce la selezione di un percorso . . . . .	256

## 1 Introduzione

### 1.1 Scopo del documento

Il presente documento ha lo scopo di definire in dettaglio la struttura e le relazioni tra le componenti del prodotto sHike, riprendendo e approfondendo quanto già descritto nel documento di Specifica Tecnica. Il documento serve da guida per i programmatore, fornendo indicazioni per l'implementazione del sistema e la codifica.

### 1.2 Scopo del prodotto

Il prodotto denominato “sHike” si propone di fornire agli escursionisti uno strumento per tracciare la propria attività, al fine di stimolarli a migliorarsi e condividere con gli altri utenti i propri risultati e percorsi. Inoltre, il prodotto è dotato di una controparte web, in cui l’utente può salvare e visualizzare i propri percorsi con i relativi risultati e successivamente può condividerli e confrontarli con quelli degli altri utenti.

### 1.3 Glossario

Per evitare ambiguità dovute all’uso di termini tecnici nei documenti, in allegato viene fornito il documento *glossario\_v3.0.pdf*. All’interno di tale documento è possibile trovare tutti i termini marcati da una sottolineatura. I termini sono definiti e descritti in modo chiaro così da evitare incomprensioni.

### 1.4 Riferimenti

#### 1.4.1 Normativi

- **Specifiche Tecniche:** [specifica\\_tecnica\\_v3.0.pdf](#);
- **Analisi dei Requisiti:** [analisi\\_dei\\_requisiti\\_v3.0.pdf](#);
- **Norme di Progetto:** [norme\\_di\\_progetto\\_v3.0.pdf](#).

#### 1.4.2 Informativi

- **HTML5 Candidate Recommendation:** <http://www.w3.org/TR/html5/>;
- **CSS, direttive W3C:** <http://www.w3.org/TR/CSS/>;
- **Standard ECMA-262 di JavaScript:** <http://www.ecma-international.org/publications/standards/Ecma-262.htm>;
- **Standard ECMA-404 di JSON:** <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>;
- **Connessione Spring e MySQL:** <http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-sql.html>;
- **Spring MVC:** <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>;
- **MySQL Documentation:** <http://dev.mysql.com/doc/>;
- **WearIT Developer Documentation:** <http://www.wearit.net/download/WearIT-Development-SDK.pdf>;

- **WearIT UI guidelines:** [http://www.wearit.net/download/WearIT-UI-guidelines-draft\\_002-web.pdf](http://www.wearit.net/download/WearIT-UI-guidelines-draft_002-web.pdf);
- **Android 4.4 APIs - API Level: 19:** <http://developer.android.com/reference/packages.html>;
- **SQLite Documentation:** <https://www.sqlite.org/docs.html>;
- **Codici delle condizioni meteo:** <http://openweathermap.org/weather-conditions/#Weather-Condition-Codes-2>;
- **ISO 5218:** [http://it.wikipedia.org/wiki/ISO\\_5218](http://it.wikipedia.org/wiki/ISO_5218);
- **Design Patterns: Elementi per il riuso di software a oggetti:** - E. Gamma, R. Helm, R. Johnson, J. Vlissides (Pearson, 2002);
- **Specifica IDEF1X:** <http://www.idef.com/pdf/Idef1x.pdf>.

## 2 Standard di progetto

### 2.1 Standard di progettazione architetturale

Gli standard di progettazione architetturale seguiti sono definiti nel documento “Specifica Tecnica v3.00”. Si faccia riferimento ad esso per approfondimenti.

### 2.2 Standard di documentazione del codice

Gli standard di documentazione del codice sono definiti e descritti nel documento “Norme di Progetto v3.00”. Si faccia riferimento ad esso per approfondimenti.

### 2.3 Standard di denominazione di entità e relazioni

Tutti gli elementi definiti nel seguente documento, siano essi *package*, classi, metodi o attributi, devono avere una denominazione chiara e concisa. Si faccia riferimento al documento “Norme di Progetto v3.00” per tutte le regole tipografiche adottate.

### 2.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nel documento “Norme di Progetto v3.00”. Si rimanda ad esso per le regole da seguire durante la codifica.

### 2.5 Strumenti di lavoro

Tutti gli strumenti di lavoro e le procedure da seguire per la corretta realizzazione del prodotto sono definiti nel documento “Norme di Progetto v3.00”. Si faccia riferimento ad esso per approfondimenti.

### 3 Specifica dei componenti

#### 3.1 Metodo e formalismo di specifica

L'esposizione dell'architettura dell'applicazione procede con un approccio di tipo *top-down*. Si descrive quindi l'architettura partendo dal generale, decomponendo le componenti fino a raggiungere il particolare. Si inizia quindi con la descrizione delle componenti. Per ogni componente si analizzeranno in dettaglio le singole classi, specificando per ognuna il tipo, la funzione, gli attributi e i metodi. Per i diagrammi delle componenti, di classe e di attività si utilizza il formalismo UML 2.X. Per permettere di distinguere facilmente le classi e le componenti presenti in librerie e framework esterni utilizzati dall'applicazione, questi verranno rappresentati con un colore verde.

#### 3.2 shike

##### 3.2.1 Informazioni sul package

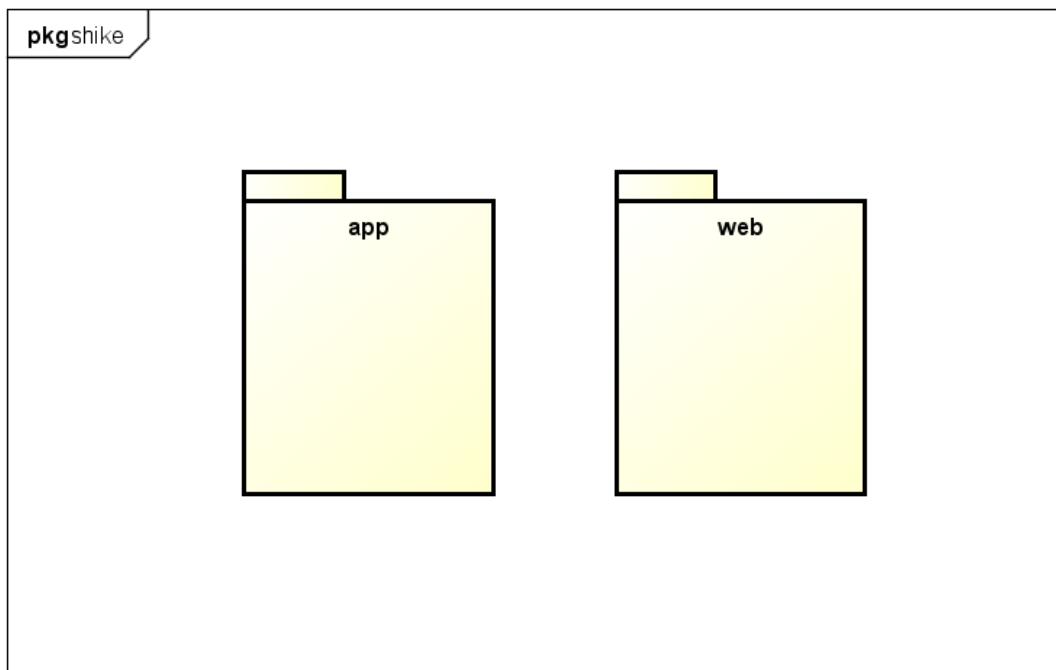


Figura 1: Diagramma di shike

- **Descrizione:** componente che contiene il prodotto sHike.
- **Componenti contenute**
  - shike::app
  - shike::web

### 3.3 shike::app

#### 3.3.1 Informazioni sul package

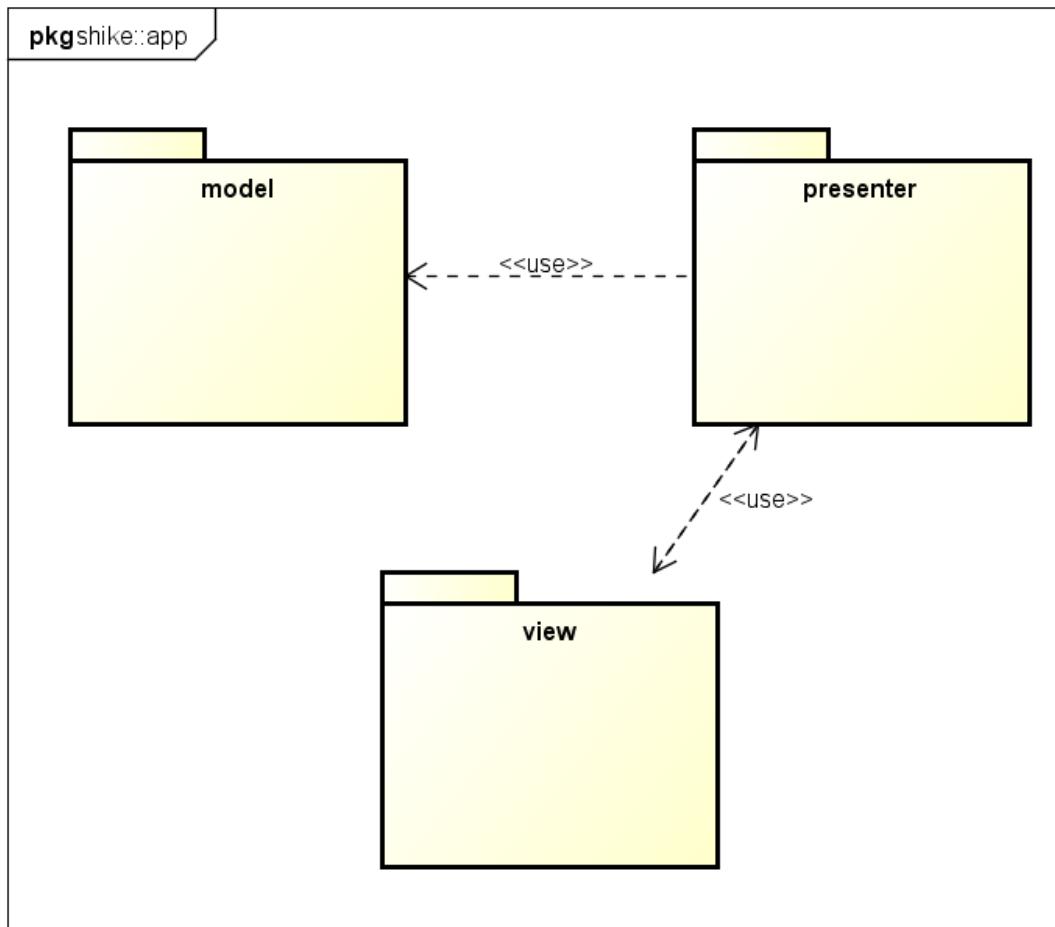


Figura 2: Diagramma di shike::app

- **Descrizione:** componente che contiene l'applicazione Android del prodotto sHike.
- **Componenti contenute**
  - shike::app::presenter
  - shike::app::model
  - shike::app::view
  - shike::app::helper
- **Componente padre:** shike

### 3.4 shike::app::presenter

#### 3.4.1 Informazioni sul package

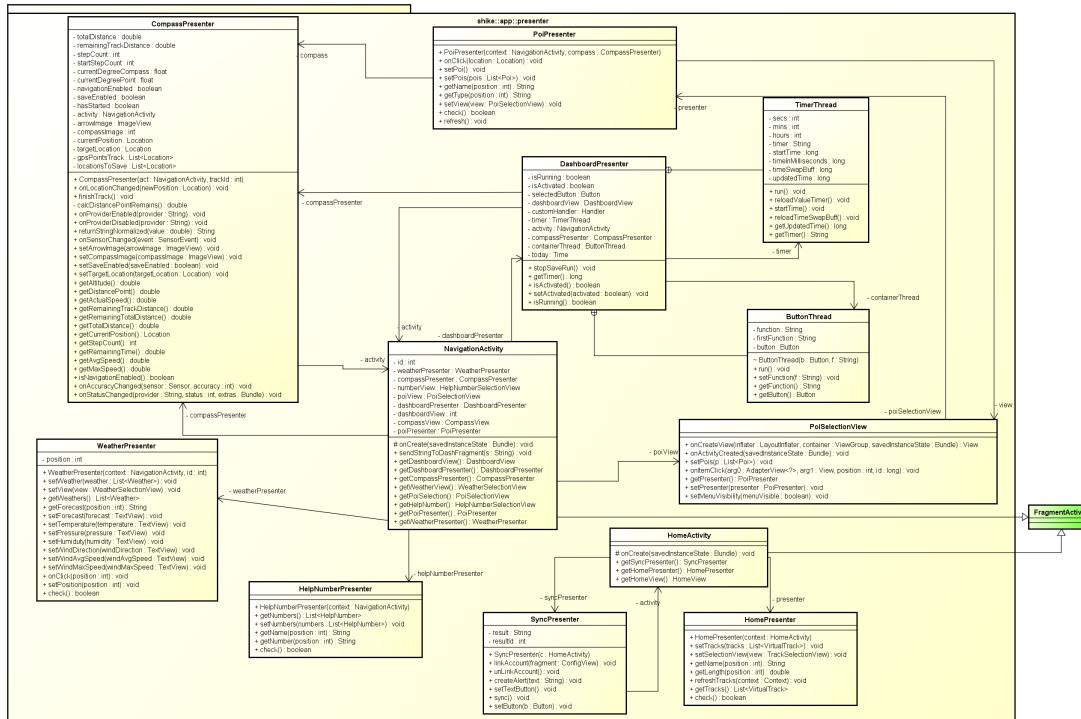


Figura 3: Diagramma di shike::app::presenter

- **Descrizione:** componente *Controller* dell'architettura MVP. Essa esegue le operazioni che l'utente ha richiesto tramite la *View* agendo se necessario sul *Model*.
- **Componente padre:** shike::app
- **Interazioni con altri componenti**
  - shike::app::model
  - shike::app::view

### 3.4.2 Classi

#### 3.4.2.1 shike::app::presenter::NavigationActivity

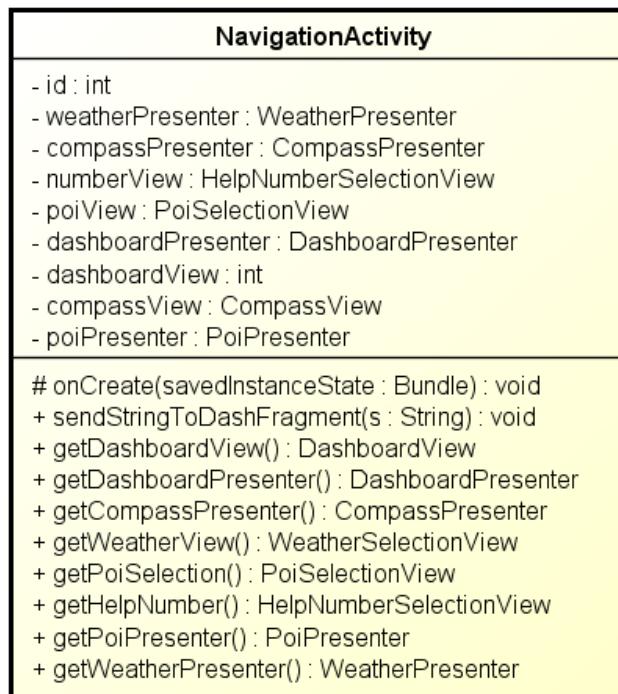


Figura 4: Diagramma di NavigationActivity

- **Tipo:** concreta
- **Descrizione:** *View* che gestisce le viste della navigazione.
- **Superclassi:**
  - android.support.v4.app.FragmentActivity
- **Attributi:**
  - **`_dashboardPresenter`** : `DashboardPresenter`  
Riferimento al *presenter* della *dashboard*.
  - **`_dashboardView`** : `DashboardView`  
Riferimento alla vista della *dashboard*.
  - **`_compassView`** : `CompassView`  
Riferimento alla vista della bussola.
  - **`_weatherView`** : `WeatherSelectionView`  
Riferimento alla vista del tempo.
  - **`_numberView`** : `HelpNumberSelectionView`  
Riferimento alla vista dei numeri di soccorso.
  - **`_poiView`** : `PoiSelectionView`  
Riferimento alla vista dei POI.

- `-compassPresenter : CompassPresenter`  
Riferimento al *presenter* della bussola.
- `-poiPresenter : poiPresenter`  
Riferimento al *presenter* dei punti di interesse.
- `-weatherPresenter : WeatherPresenter`  
Riferimento al *presenter* del meteo.

- **Metodi:**

- `+getDashboardPresenter() : DashboardPresenter`  
Restituisce la variabile `dashboardPresenter`.
- `+getDashboardView() : DashboardView`  
Restituisce la variabile `dashboardView`.
- `+getCompassView() : CompassView`  
Restituisce la variabile `compassView`.
- `+getWeatherView() : WeatherSelectionView`  
Restituisce la variabile `weatherView`.
- `+getNumberView() : HelpNumberSelectionView`  
Restituisce la variabile `numberView`.
- `+getPoiView() : PoiSelectionView`  
Restituisce la variabile `poiView`.
- `+getCompassPresenter() : CompassPresenter`  
Restituisce la variabile `compassPresenter`.
- `+getPoiPresenter() : poiPresenter`  
Restituisce la variabile `poiPresenter`.
- `+getWeatherPresenter() : WeatherPresenter`  
Restituisce la variabile `weatherPresenter`.
- `+onCreate( savedInstanceState : Bundle ) : void`  
Esegue l'inizializzazione di tutti i *Fragment*.

**Argomenti:**

- \* `savedInstanceState` : se l'attività viene re-inizializzata dopo un precedente stato di arresto allora vengono ricaricati dati precedentemente presenti nel `Bundle`.
- `+passStringAtFragment( s : String ) : void`  
Passa una stringa da un *Fragment* ad un'altro.

**Argomenti:**

- \* `s` : stringa da passare.

### 3.4.2.2 shike::app::presenter::WeatherPresenter

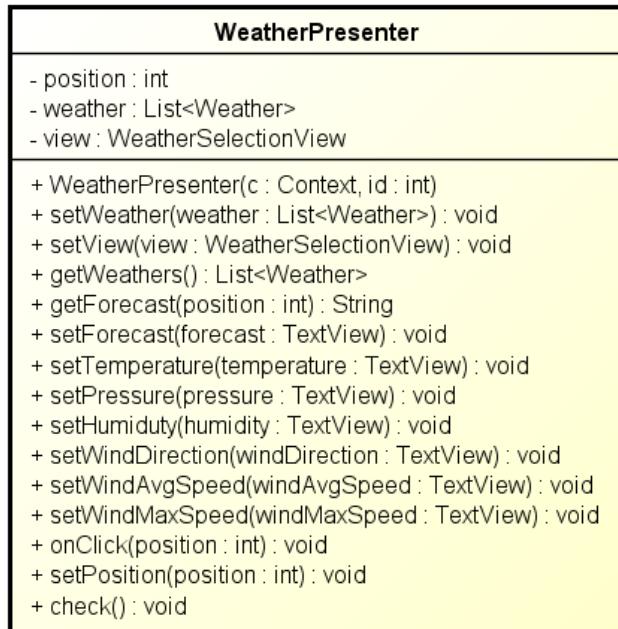


Figura 5: Diagramma di WeatherPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate per gestire le informazioni sul meteo.
- **Attributi:**
  - **`-weather`** : `List<Weather>`  
Lista delle previsioni meteo.
  - **`-view`** : `WeatherSelectionView`  
Riferimento alla view.
  - **`-position`** : `int`  
Posizione dell'elemento da gestire.
- **Metodi:**
  - **`+WeatherPresenter( context : Context, id : int )`** :  
Costruttore di WeatherPresenter.  
**Argomenti:**
    - \* `context` : riferimento all'activity.
    - \* `id` : id del percorso scelto.
  - **`+setView( view : WeatherSelectionView )`** :  
Imposta il riferimento alla view.  
**Argomenti:**
    - \* `view` : riferimento alla view.

- `+setWeathers() : List<Weather>`  
Ritorna la lista delle previsioni meteo.
- `+setForecast() : String`  
Imposta la condizione meteo prevista.
- `+setTemperature() : String`  
Imposta la temperatura prevista.
- `+setPressure() : String`  
Imposta il livello di pressione previsto.
- `+setHumidity() : String`  
Imposta il livello di umidità previsto.
- `+setWindDirection() : String`  
Imposta la direzione del vento prevista.
- `+setWindAvgSpeed() : String`  
Imposta la velocità media del vento prevista.
- `+setWindMaxSpeed() : String`  
Imposta la velocità massima del vento prevista.
- `+onClick( position : int ) : void`  
Visualizza il dialogFragment dell'elemento della lista selezionato.

**Argomenti:**

\* `position` : posizione dell'elemento sulla lista.

- `+setPosition( position : int ) : void`  
Imposta la posizione dell'elemento da utilizzare.

**Argomenti:**

\* `position` : posizione dell'elemento.

- `+getWeathers() : List<Weather>`  
Ritorna la lista dei meteo salvati.
- `+check() : void`  
Controlla se la lista del meteo è vuota, se lo è comunica che non ci sono informazioni meteo per quel percorso.
- `+getForecast( position : int ) : String`  
Ritorna la condizione meteo dell'elemento indicato.

**Argomenti:**

\* `position` : posizione dell'elemento.

### 3.4.2.3 shike::app::presenter::DashboardPresenter

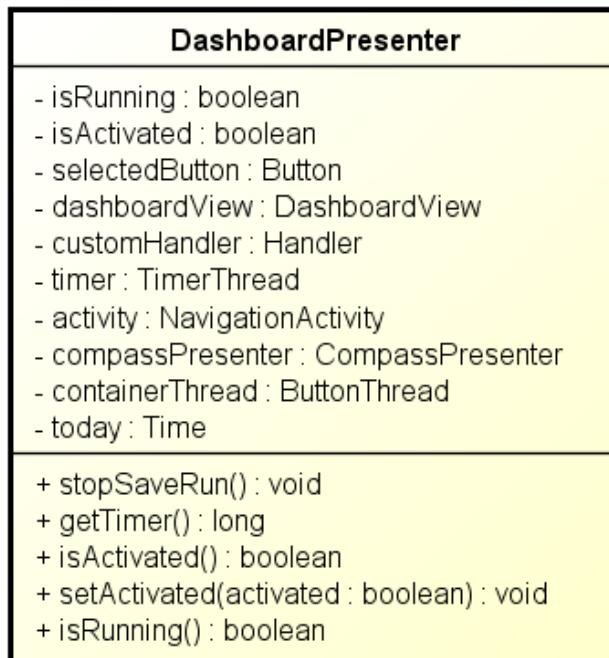


Figura 6: Diagramma di DashboardPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dalla *dashboard*.
- **Attributi:**
  - **`_isActivated`** : `boolean`  
Indica se è stata avviata la *dashboard*.
  - **`_compassPresenter`** : `CompassPresenter`  
Riferimento al *presenter* della bussola.
  - **`_activity`** : `NavigationActivity`  
Riferimento all' *activity* principale.
  - **`_timer`** : `timerThread`  
Oggetto che gestisce il *thread* del *timer*.
  - **`_containerThread`** : `Vector<customThread>`  
Contiene tutti i *thread*.
  - **`_customHandler`** : `Handler`  
Barra di gestione delle opzione del cronometro.
  - **`_selectedButton`** : `Button`  
Bottone selezionato.
  - **`_isRunning`** : `boolean`  
Se true indica che il cronometro sta andando altrimenti indica che è in stato di pausa.

- `-dashboardView : Dashboard`  
Riferimento alla dashboard.
- `-today : Time`  
Contiene la data e l'ora corrente.

- **Metodi:**

- `+getTimer() : timerThread`  
Restituisce la variabile `timer`.
- `+setTimer( timer : timerThread ) : void`  
**Argomenti:**

\* `timer` : valore da impostare nella variabile `timer`.

- `+onLongClick( b : Button ) : void`  
Indica le operazioni da effettuare se è avvenuto un *long click*.

**Argomenti:**

\* `b` : indica il bottone premuto.

- `+onStop( button : Button, dialog : DialogConfirmSaveTrack ) : void`  
Ferma il cronometro e fa vedere un *dialog* di conferma salvataggio.

**Argomenti:**

\* `button` : riceve il bottone da cambiare.

\* `dialog` : riceve il *dialog* da impostare.

- `+startPauseRun( ) : String`  
Gestisce le fasi di *start* e pausa del cronometro.
- `+setWidget( b : Button, name : String ) : void`  
Imposta il *widget* desiderato sulla cella indicata.

**Argomenti:**

\* `b` : bottone da impostare.

\* `name` : nome del widget da impostare nel bottone.

- `+changeWidget( s : String ) : void`  
Imposta il *widget* selezionato nel pulsante indicato.

**Argomenti:**

\* `s` : nome del widget da impostare nel bottone.

- `+stopSaveRun( ) : void`  
Se nel *dialog* si conferma la chiusura della sessione invoca il metodo che salva la sessione.

- `+DashboardPresenter( act : NavigationActivity ) : void`  
Costruttore della classe.

**Argomenti:**

\* `act` : riferimento all' *activity*.

### 3.4.2.4 shike::app::presenter::CompassPresenter

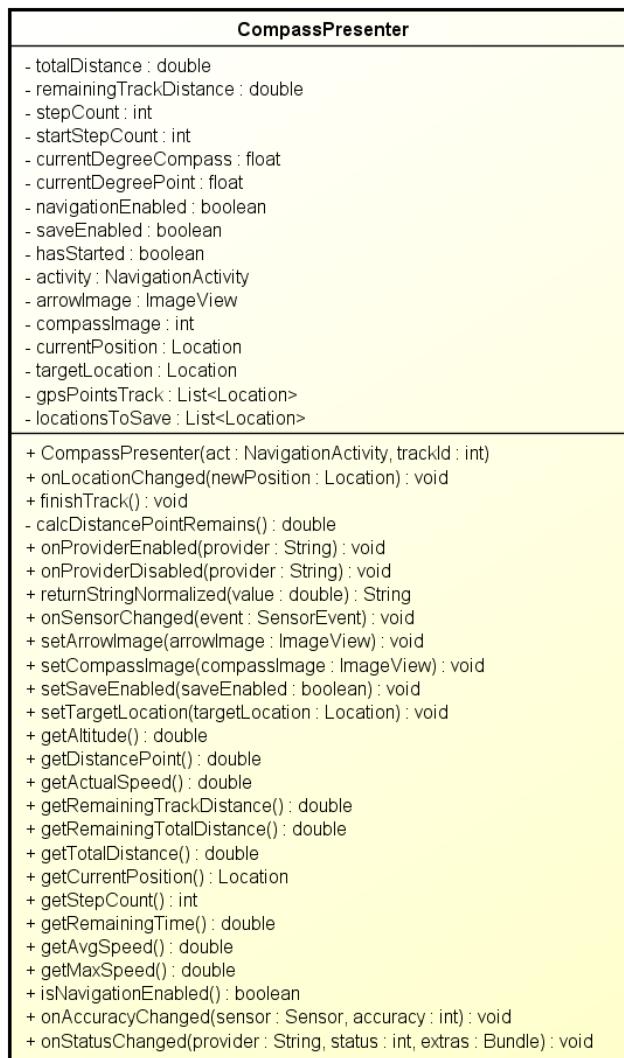


Figura 7: Diagramma di CompassPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dalla bussola.
- **Attributi:**
  - **`-activity`** : `NavigationActivity`  
Contesto dell'applicazione.
  - **`-currentDegreeCompass`** : `float`  
Angolazione dell'immagine della bussola attuale.
  - **`-currentDegreePoint`** : `float`  
Angolazione dell'immagine della freccia attuale.
  - **`-imageCompass`** : `ImageView`  
Immagine della bussola.

- `imageArrow : ImageView`  
Immagine dell' freccia che indica il prossimo punto da raggiungere.
- `navigationEnabled : boolean`  
Booleano che indica l'attivazione o meno della navigazione verso un punto.
- `GpsPointsTrack : List<Location>`  
Lista dei punti da seguire.
- `currentPosition : Location`  
Posizione corrente.
- `targetLocation : Location`  
Posizione del prossimo punto da raggiungere.
- `locationsToSave : private List<Location>`  
Lista dei punti da salvare.
- `hasStarted : boolean`  
Indica se l'applicazione è appena stata attivata.
- `saveEnabled : boolean`  
Indica se il salvataggio è abilitato.
- `startStepCount : int`  
Intero che indica i passi che l'applicazione ha registrato fino ad ora. Android per ragioni di efficienza resetta il contatore solo al riavvio.
- `remainingTrackDistance : double`  
Distanza rimanente del percorso.
- `stepCount : int`  
Intero che indica i passi effettuati in questa sessione.
- `totalDistance : double`  
Distanza totale percorsa.

- **Metodi:**

- `+getActivity() : NavigationActivity`  
Restituisce la variabile `activity`.
- `+setActivity( activity : NavigationActivity ) : void`  
**Argomenti:**
  - \* `activity` : valore da impostare nella variabile `activity`.
- `+getCurrentDegreeCompass() : float`  
Restituisce la variabile `currentDegreeCompass`.
- `+setCurrentDegreeCompass( currentDegreeCompass : float ) : void`  
**Argomenti:**
  - \* `currentDegreeCompass` : valore da impostare nella variabile `currentDegreeCompass`.
- `+getCurrentDegreePoint() : float`  
Restituisce la variabile `currentDegreePoint`.
- `+setCurrentDegreePoint( currentDegreePoint : float ) : void`  
**Argomenti:**
  - \* `currentDegreePoint` : valore da impostare nella variabile `currentDegreePoint`.
- `+getImageCompass() : ImageView`  
Restituisce la variabile `imageCompass`.

- `+setImageCompass( imageCompass : ImageView ) : void`  
**Argomenti:**
  - \* `imageCompass` : valore da impostare nella variabile `imageCompass`.
- `+getImageArrow() : ImageView`  
Restituisce la variabile `imageArrow`.
- `+onProviderEnabled( provider : String ) : void`  
Indica che il GPS è attivo.  
**Argomenti:**
  - \* `provider` : stringa da passare.
- `+onProviderDisabled( provider : String ) : void`  
Indica che il GPS è disabilitato.  
**Argomenti:**
  - \* `provider` : stringa da passare.
- `+onAccuracyChanged( accuracy : int, sensor : Sensor ) : void`  
Cambio raggio di accettazione del punto.  
**Argomenti:**
  - \* `accuracy` : indica la precisione.
  - \* `sensor` : indica i sensori.
- `+onSensorChanged( event : SensorEvent ) : void`  
Operazioni da compiere quando c'è un aggiornamento dei dati dei sensori.  
**Argomenti:**
  - \* `event` : evento avvento.
- `+finishTrack( ) :`  
Operazioni da effettuare quando si decide di finire il percorso.
- `+onLocationChanged( newLocation : Location ) : void`  
Quando cambia la posizione attuale aggiorna i campi dato che devono essere aggiornati.
- `+CompassPresenter( ) :`  
Costruttore della classe.
- `+onStatusChanged( ) : void`  
Operazioni da effettuare quando lo stato viene cambiato.
- `+calcDistancePointRemains( ) : double`  
Calcola la distanza di un percorso.

### 3.4.2.5 shike::app::presenter::SyncPresenter

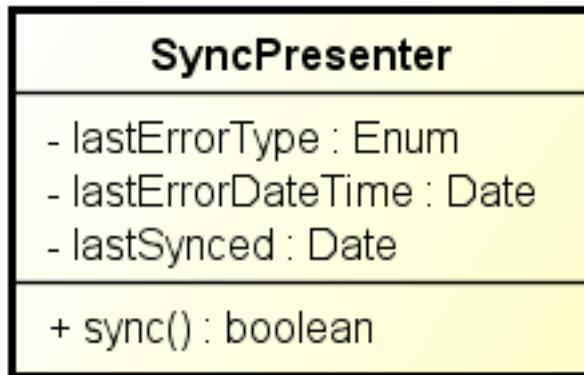


Figura 8: Diagramma di SyncPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate nella sincronizzazione.
- **Attributi:**
  - `lastErrorCode` : enum  
Tipo di errore.
  - `lastErrorDateTime` : Date  
Data dell'ultimo errore verificato.
  - `lastSynced` : Date  
Data ultima sincronizzazione.
- **Metodi:**
  - `+sync()` : boolean  
Sincronizza il *device* con la piattaforma web.

### 3.4.2.6 shike::app::presenter::PoiPresenter

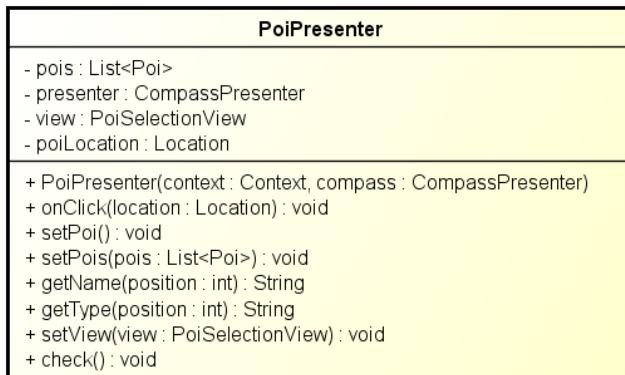


Figura 9: Diagramma di PoiPresenter

- **Tipo:** concreta

- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dei POI.

- **Attributi:**

- `–pois : List<Poi>`  
Contiene la lista dei POI.
- `–compass : CompassPresenter`  
Riferimento alla bussola.
- `–view : PoiSelectionView`  
Riferimento alla view.
- `–poiLocation : Location`  
Posizione del punto di interesse da raggiungere.

- **Metodi:**

- `+onClick( location : Location ) : void`  
Visualizza il dialogFragment associato all'elemento cliccato.

**Argomenti:**

- \* `location` : posizione del punto di interesse.

- `+setPoi() : void`  
Imposta il poi come destinazione della navigazione.

- `+PoiPresenter( context : Context, view : PoiSelectionView, compass : CompassPresenter ) :`  
Costruttore di PoiPresenter.

**Argomenti:**

- \* `context` : riferimento all'activity.
- \* `view` : riferimento alla view.
- \* `compass` : riferimento alla bussola.

- `+setView( view : PoiSelectionView ) : void`  
Imposta il riferimento alla view.

**Argomenti:**

- \* `view` : riferimento alla view.

- `+getName( position : int ) : String`  
Ritorna il nome dell'elemento nella posizione indicata.

**Argomenti:**

- \* `position` : posizione dell'elemento.

- `+getType( position : int ) : String`  
Ritorna il tipo dell'elemento nella posizione indicata.

**Argomenti:**

- \* `position` : posizione dell'elemento.

- `+setPois( pois : List<Poi> ) : void`  
Imposta i POI nelle vicinanze.

**Argomenti:**

- \* `pois` : lista dei POI da salvare.

- `+check() : void`  
Controlla se la lista dei poi è vuota, se lo è comunica che non ci sono poi nelle vicinanze.

### 3.4.2.7 shike::app::presenter::SettingPresenter

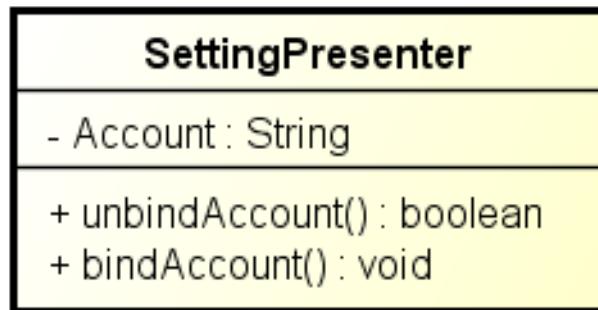


Figura 10: Diagramma di SettingPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate nella modifica delle impostazioni.
- **Attributi:**
  - **-Account : String**  
Chiave che collega l'orologio all' *account*.
- **Metodi:**
  - **+unbindAccount() : boolean**  
Scollega il *device* cancellando il *token*.
  - **+bindAccount( onetimepass : String ) : boolean**  
Collega il *device* alla piattaforma web.  
**Argomenti:**
    - \* **onetimepass** : token di conferma.

### 3.4.2.8 shike::app::presenter::DashboardPresenter::TimerThread

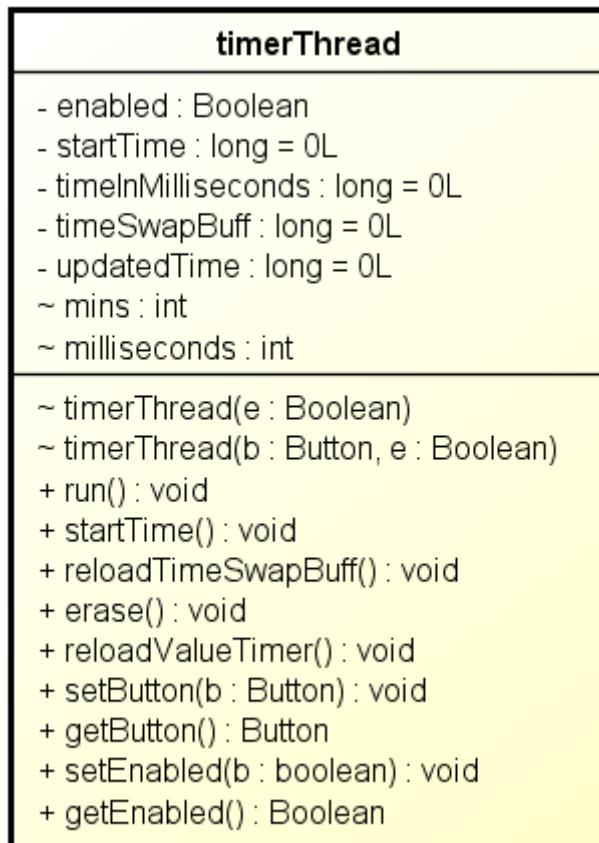


Figura 11: Diagramma di DashboardPresenter::TimerThread

- **Tipo:** concreta
- **Descrizione:** *Thread* che si occupa di gestire il *timer*.
- **Implementa:**
  - java.lang.Runnable
- **Attributi:**
  - **`_startTime`** : **long**  
Valore di inizializzazione del *timer*.
  - **`_timeInMilliseconds`** : **long**  
Tempo del *timer* in millisecondi.
  - **`_timeSwapBuff`** : **long**  
Variabile per memorizzare il tempo quando il *timer* è fermo.
  - **`_updatedTime`** : **long**  
Tempo del *timer* aggiornato.
  - **`_secs`** : **int**  
Secondi passati.

- `—mins : int`  
Minuti passati.
- `—hours : int`  
Ore passati.
- `—timer : String`  
Stringa del tempo trascorso.

- **Metodi:**

- `+run() : void`  
Funzione di avvio del *thread* che imposta tutti i valori necessari.
- `+startTime() : void`  
Imposta la variabile con l'ora attuale.
- `+reloadTimeSwapBuff() : void`  
Ricarica la variabile *timeSwapBuff*.
- `+reloadValueTimer() : void`  
Aggiorna la stringa da stampare.

### 3.4.2.9 shike::app::presenter::HomeActivity

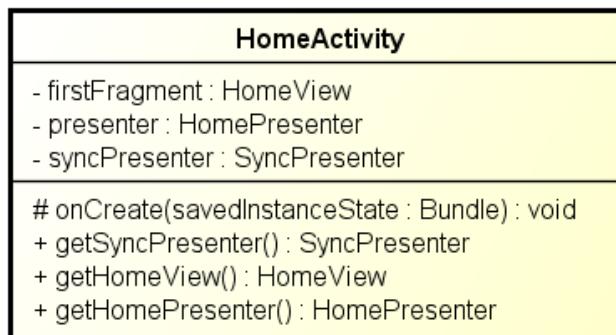


Figura 12: Diagramma di HomeActivity

- **Tipo:** concreta
- **Descrizione:** classe che gestisce le viste da istanziare in *homepage*.
- **Superclassi:**
  - `android.support.v4.app.FragmentActivity`
- **Attributi:**
  - `—firstFragment : HomeView`  
*Fragment* da istanziare all'avvio dell'applicazione.
  - `—presenter : HomePresenter`  
Riferimento al presenter.
  - `—syncPresenter : SyncPresenter`  
Riferimento al presenter della sincronizzazione.
- **Metodi:**

– `+onCreate( savedInstanceState : Bundle ) : void`

Esegue l'inizializzazione di tutti i *Fragment*.

**Argomenti:**

\* `savedInstanceState` : se l'attività viene re-inizializzata dopo un precedente stato di arresto allora vengono ricaricati dati precedentemente presenti nel Bundle.

– `+getHomeView( ) : HomeView`

Ritorna un riferimento alla HomeView.

– `+SyncPresenter( ) : SyncPresenter`

Ritorna il riferimento al presenter della sincronizzazione.

– `+getHomePresenter( ) : HomePresenter`

Ritorna il riferimento al presenter.

### 3.4.2.10 shike::app::presenter::HomePresenter

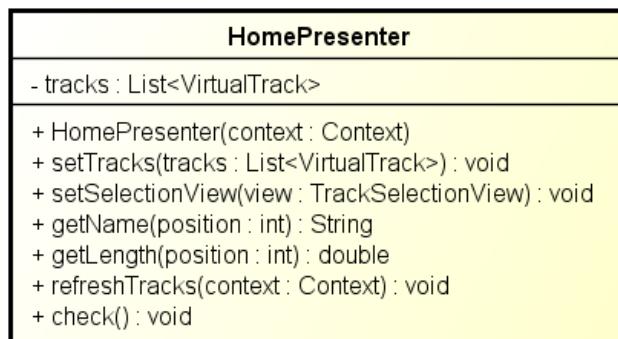


Figura 13: Diagramma di HomePresenter

- **Tipo:** concreta

- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dalla *homepage*.

- **Attributi:**

– `_tracks : List<VirtualTrack>`

Lista dei tracciati salvati nel dispositivo.

- **Metodi:**

– `+HomePresenter( context : Context ) :`

Costruttore di HomePresenter.

**Argomenti:**

\* `context` : riferimento all'activity.

– `+setSelectionView( view : TrackSelectionView ) : void`

Imposta il riferimento alla view.

**Argomenti:**

\* `view` : riferimento alla view.

– `+getName( position : int ) : String`

Ritorna il nome del tracciato nella posizione indicata.

**Argomenti:**

\* `position` : riferimento alla posizione dell'oggetto da utilizzare.

- `+getLength( position : int ) : double`  
Ritorna la lunghezza del tracciato nella posizione indicata.

**Argomenti:**

- \* `position` : riferimento alla posizione dell'oggetto da utilizzare.

- `+refreshTracks( context : Context ) : void`  
Aggiorna la lista dei tracciati.

**Argomenti:**

- \* `context` : riferimento al contenuto.

- `+check( ) : void`  
Controlla se la lista dei tracciati è vuota, se lo è comunica che non ci sono tracciati salvati.

- `+setTracks( tracks : List<VirtualTrack> ) : void`  
Modifica la lista dei percorsi salvati.

**Argomenti:**

- \* `tracks` : lista dei percorsi.

### 3.4.2.11 shike::app::presenter::NavigationActivity::MyPagerAdapter

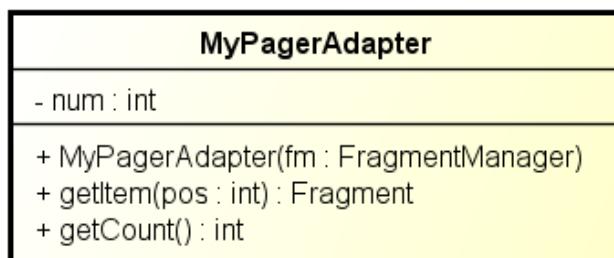


Figura 14: Diagramma di NavigationActivity::MyPagerAdapter

- **Tipo:** concreta
- **Descrizione:** classe che modella i *Fragment* all'interno della classe *ActivityNavigation*.
- **Superclassi:**
  - android.support.v4.app.FragmentPagerAdapter
- **Metodi:**
  - `+getItem( pos : int ) : Fragment`  
Ritorna il *Fragment* presente nella posizione passata.

**Argomenti:**

  - \* `pos` : posizione del Fragment.
  - `+getCount( ) : int`  
Ritorna il numero di *Fragment* usati.

### 3.4.2.12 shike::app::presenter::SyncPresenter

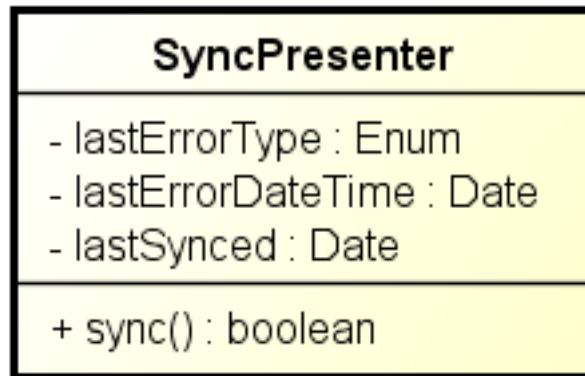


Figura 15: Diagramma di SyncPresenter

- **Tipo:** concreta
- **Descrizione:** classe che gestisce la logica della sincronizzazione con il *server web*.
- **Attributi:**
  - `–activity` : `HomeActivity`  
Riferimento all'*activity*.
  - `–result` : `String`  
Risultato dell'operazione.
  - `–button` : `Button`  
Riferimento al pulsante da modificare.
  - `–resultId` : `int`  
Id dell'errore.
- **Metodi:**
  - `+setActivity( activity : HomeActivity ) : void`  
**Argomenti:**
    - \* `activity` : valore da impostare nella variabile `activity`.
  - `+linkAccount( fragment : ConfigView ) : void`  
Crea una richiesta al server che restituisce il numero da inserire nel portale web per collegare il dispositivo al portale. Nel caso fallisca mostra l'errore.  
**Argomenti:**
    - \* `fragment` : riferimento al *fragment*.
  - `+SyncPresenter( ) :`  
Costruttore della classe.
  - `+unLinkAccount( ) : void`  
Scollega l'account dal dispositivo.
  - `+createAlert( text : String ) : void`  
Crea un *dialog* di errore.  
**Argomenti:**
    - \* `text` : testo del dialogo.

- \* `text` : stringa di errore.
- `+setTextButton( ) : void`  
Sostituisce il testo giusto al pulsante.
- `+sync( ) : void`  
Effettua le operazioni di sincronizzazione.

### 3.4.2.13 shike::app::presenter::DashboardPresenter::ButtonThread

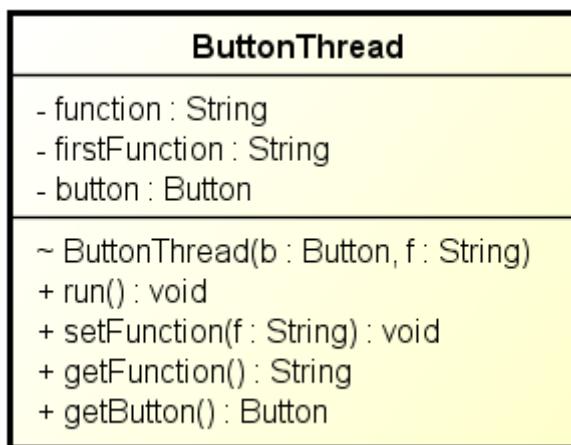


Figura 16: Diagramma di DashboardPresenter::ButtonThread

- **Tipo:** concreta
- **Descrizione:** *Thread* che gestisce un pulsante della *dashboard* permettendo di aggiornare il valore o cambiare il valore visualizzato
- **Implementa:**
  - java.lang.Runnable
- **Attributi:**
  - `_function : String`  
Contiene la stringa che indica la funzionalità voluta.
  - `_button : Button`  
Pulsante a cui si riferisce il *thread*.
  - `_firstFunction : String`  
Contiene la funzione precedente.
- **Metodi:**
  - `+getFunction() : String`  
Restituisce la variabile `function`.
  - `+getButton() : Button`  
Restituisce la variabile `button`.
  - `+setButton( button : Button ) : void`  
**Argomenti:**

- \* `button` : valore da impostare nella variabile `button`.
- `+shike.app.presenter.DashboardPresenter.ButtonThread( )` :  
Costruttore della classe.
- `+run( ) : void`  
Corpo del *thread* che aggiorna o cambia la funzionalità richiesta.

### 3.4.2.14 shike::app::presenter::HelpNumberPresenter

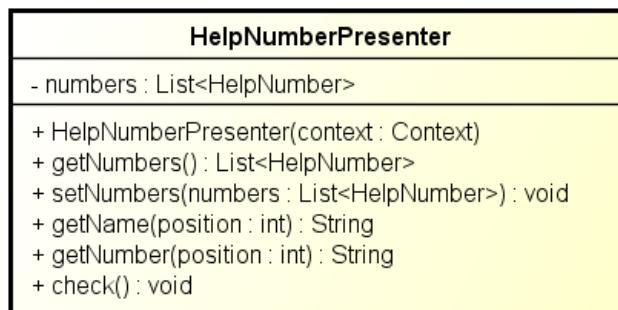


Figura 17: Diagramma di HelpNumberPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dai numeri di soccorso.
- **Attributi:**
  - `-numbers : List<HelpNumber>`  
Lista dei numeri di soccorso salvati nel dispositivo.
- **Metodi:**
  - `+getName( position : int ) : String`  
Ritorna il nome del numero di soccorso nella posizione indicata.  
**Argomenti:**
    - \* `position` : posizione dell'elemento.
  - `+HelpNumberPresenter( context : Context )` :  
Costruttore di HelpNumberPresenter.  
**Argomenti:**
    - \* `context` : riferimento all'activity.
  - `+getNumbers( ) : List<HelpNumber>`  
Ritorna la lista dei numeri di soccorso.
  - `+setNumbers( numbers : List<HelpNumber> ) : void`  
Imposta la lista dei numeri di soccorso.  
**Argomenti:**
    - \* `numbers` : lista dei numeri di soccorso.
  - `+getNumber( position : int ) : String`  
Ritorna il numero dell'elemento nella posizione indicata.  
**Argomenti:**
    - \* `position` : posizione dell'elemento.

- \* `position` : posizione dell'elemento.
- `+check()` : `void`  
Controlla se la lista dei numeri di soccorso è vuota, se lo è comunica che non ci sono numeri di soccorso salvati.

### 3.5 shike::app::model

#### 3.5.1 Informazioni sul package

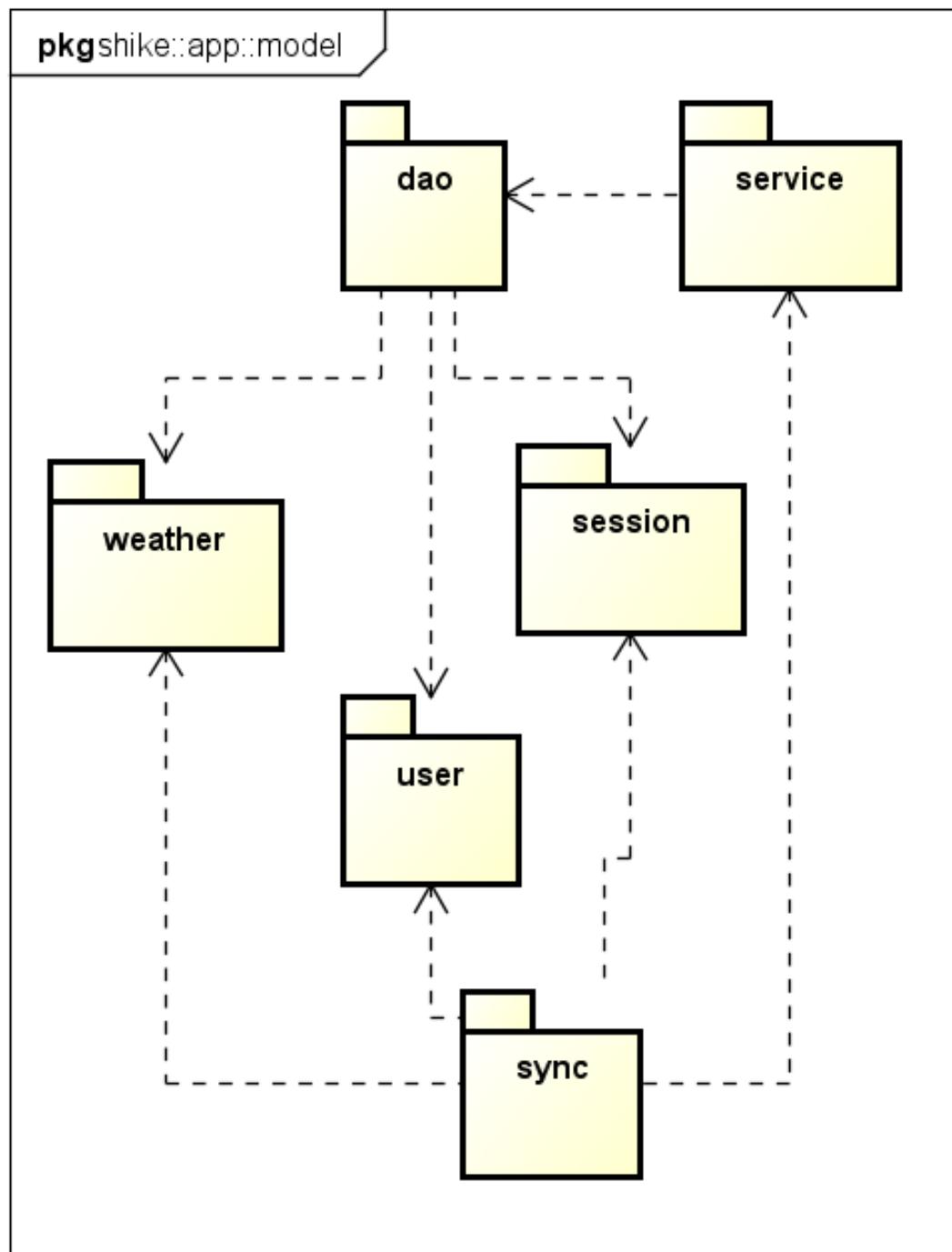


Figura 18: Diagramma di shike::app::model

- **Descrizione:** componente *Model* dell'architettura MVP. Essa fornisce un modello dei dati che la *View* deve rappresentare.
- **Componenti contenute**
  - shike::app::model::weather
  - shike::app::model::session
  - shike::app::model::user
  - shike::app::model::dao
  - shike::app::model::service
  - shike::app::model::sync
- **Componente padre:** shike::app
- **Interazioni con altri componenti**
  - shike::app::view

### 3.6 shike::app::model::weather

#### 3.6.1 Informazioni sul package

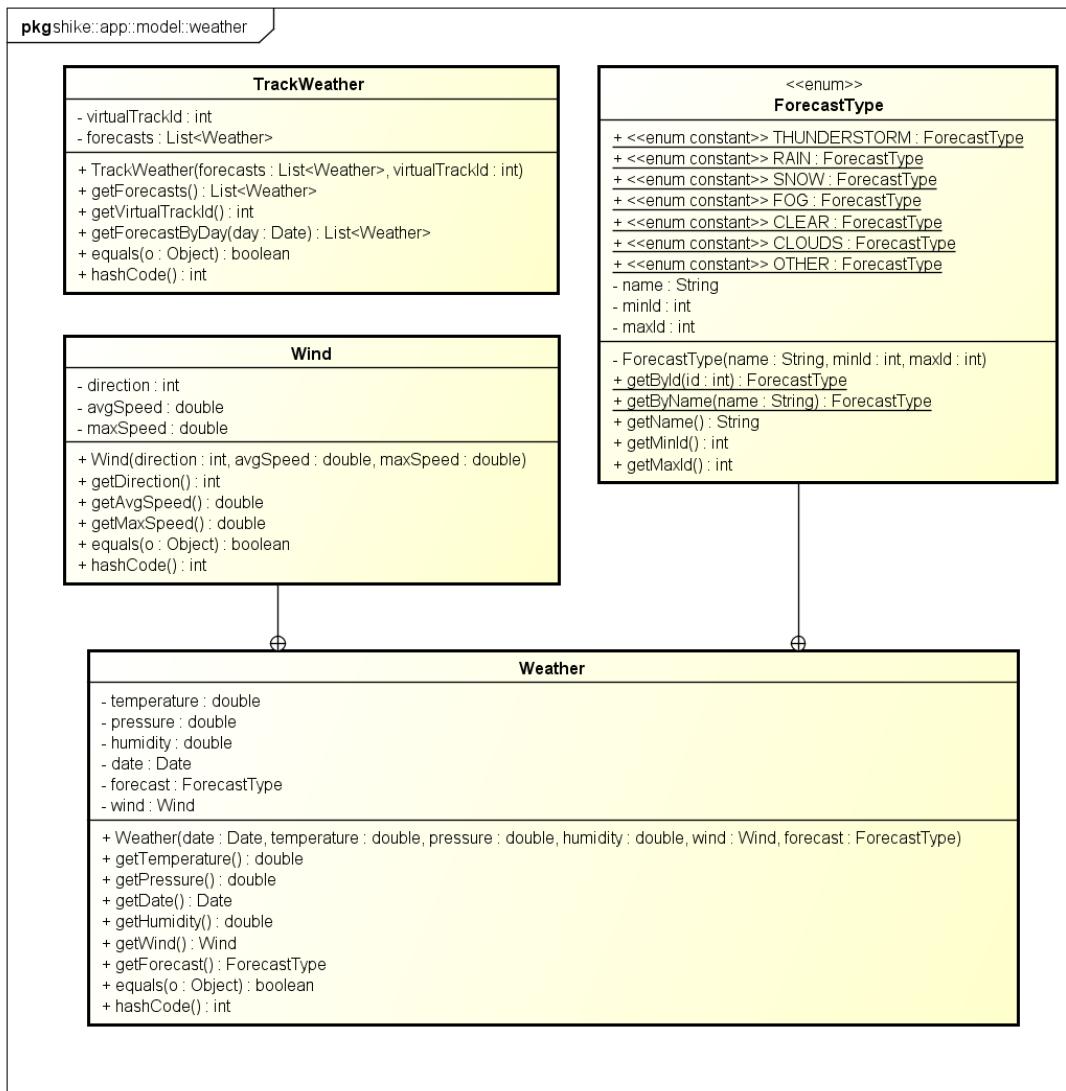


Figura 19: Diagramma di shike::app::model::weather

- **Descrizione:** componente del *Model* utilizzata per modellare la struttura dei dati delle condizioni climatiche riguardanti un certo percorso.
- **Componente padre:** shike::app::model
- **Interazioni con altri componenti**
  - shike::app::model::session::track

### 3.6.2 Classi

#### 3.6.2.1 shike::app::model::weather::Weather

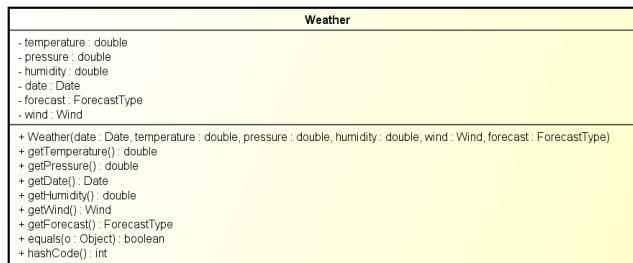


Figura 20: Diagramma di Weather

- **Tipo:** concreta
- **Descrizione:** classe che modella una singola previsione meteo
- **Attributi:**
  - `+temperature : double`  
Temperatura prevista (in K).
  - `+pressure : double`  
Pressione prevista (in hPa).
  - `+forecast : Weather.ForecastType`  
Tipologia di condizione meteo prevista.
  - `+humidity : double`  
Umidità relativa prevista (in percentuale).
  - `+wind : Weather.Wind`  
Informazioni sul vento previste.
- **Metodi:**
  - `+getTemperature() : double`  
Restituisce la variabile `temperature`.
  - `+getPressure() : double`  
Restituisce la variabile `pressure`.
  - `+getForecast() : Weather.ForecastType`  
Restituisce la variabile `forecast`.
  - `+getHumidity() : double`  
Restituisce la variabile `humidity`.
  - `+getWind() : Weather.Wind`  
Restituisce la variabile `wind`.
  - `+Weather( temperature : double, pressure : double, humidity : double, wind : Weather.Wind, forecast : Weather.ForecastType ) :`  
Costruttore completo della classe.
- **Argomenti:**
  - \* `temperature` : temperatura in K della previsione.
  - \* `pressure` : pressione in hPa della previsione.

- \* `humidity` : percentuale di umidità relativa.
- \* `wind` : informazioni sul vento.
- \* `forecast` : tipo di condizione meteo prevista.
- `+equals( o : Object ) : boolean`  
*OVERRIDING* che controlla l'uguaglianza per ogni singolo attributo della classe.
- Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+hashCode() : int`  
*OVERRIDE* che genera un codice *hash* considerando tutti gli attributi della classe.

### 3.6.2.2 shike::app::model::weather::TrackWeather

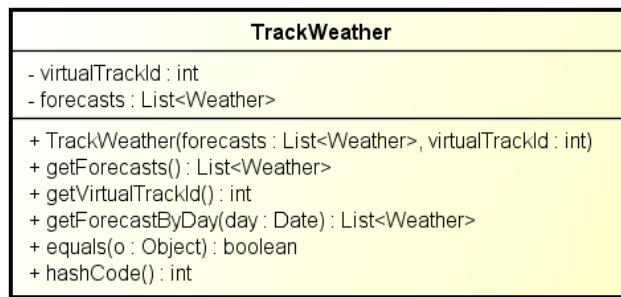


Figura 21: Diagramma di TrackWeather

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni relative alle condizioni meteo previste in un percorso.
- **Attributi:**
  - `-forecasts : List<Weather>`  
 Lista delle previsioni meteo del percorso, ordinate per data.
  - `-virtualTrackId : int`  
 Id del *VirtualTrack* associato alle previsioni.
- **Metodi:**
  - `+getForecasts() : List<Weather>`  
 Restituisce la variabile `forecasts`.
  - `+getVirtualTrackId() : int`  
 Restituisce la variabile `virtualTrackId`.
  - `+TrackWeather( forecasts : List<Weather>, virtualTrackId : int ) :`  
 Costruttore della classe. Ordina la lista delle previsioni passate per parametro per data.
- Argomenti:**
  - \* `forecasts` : previsioni associate al percorso.
  - \* `virtualTrackId` : id del percorso associato alle previsioni.
- `+getForecastByDay( day : Date ) : List<Weather>`  
 Ritorna le previsioni meteo per il giorno indicato.
- Argomenti:**

- \* `day` : data contenente il giorno di cui si desiderano le previsioni.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.
- Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+hashCode( ) : int`  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe.

### 3.6.2.3 shike::app::model::weather::Weather::ForecastType

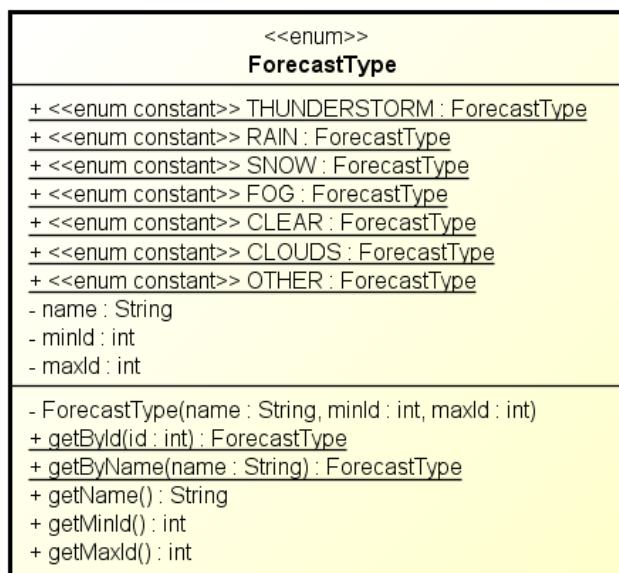


Figura 22: Diagramma di Weather::ForecastType

- **Tipo:** enum
- **Descrizione:** enum che indica il tipo di condizioni meteo previste. I codici indicati nelle condizioni si riferiscono agli identificatori di <http://openweathermap.org/weather-conditions>
- **Attributi:**
  - `-name : String`  
 Nome della condizione meteo.
  - `-minId : int`  
 Limite inferiore (incluso) dei numeri che identificano la condizione meteo.
  - `-maxId : int`  
 Limite superiore (incluso) dei numeri che identificano la condizione meteo.
- **Metodi:**
  - `+getName() : String`  
 Restituisce la variabile `name`.
  - `+getMinId() : int`  
 Restituisce la variabile `minId`.

- `+getMaxId() : int`  
Restituisce la variabile `maxId`.
- `+ForecastType( name : String, minId : int, maxId : int ) :`  
Costruttore completo della enum.
- Argomenti:**
  - \* `name` : nome della condizione meteo.
  - \* `minId` : id minimo della condizione.
  - \* `maxId` : id massimo della condizione.
- `+ getById( id : int ) : ForecastType`  
Ritorna la prima condizione meteo che contiene nel suo intervallo degli id accettabili quello indicato.
- Argomenti:**
  - \* `id` : id della condizione meteo ricercata.
- `+getByName( name : String ) : ForecastType`  
Ritorna la prima condizione meteo che ha il nome uguale a quello passato per parametro.
- Argomenti:**
  - \* `name` : nome della condizione da cercare.

### 3.6.2.4 shike::app::model::weather::Weather::Wind

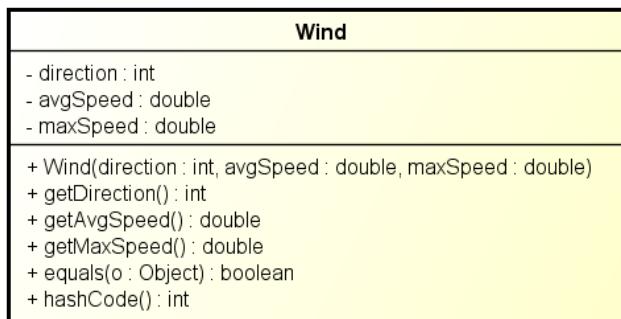


Figura 23: Diagramma di Weather::Wind

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni sul vento di una previsione meteo
- **Attributi:**
  - `–direction : int`  
Direzione del vento in gradi.
  - `–avgSpeed : double`  
Velocità media del vento in m/s.
  - `–maxSpeed : double`  
Velocità di picco del vento in m/s.
- **Metodi:**
  - `+getDirection() : int`  
Restituisce la variabile `direction`.

- `+getAvgSpeed() : double`  
Restituisce la variabile `avgSpeed`.
- `+getMaxSpeed() : double`  
Restituisce la variabile `maxSpeed`.
- `+Wind( direction : int, avgSpeed : double, maxSpeed : double ) :`  
Costruttore completo della classe.

**Argomenti:**

- \* `direction` : direzione del vento.
- \* `avgSpeed` : velocità media del vento in m/s.
- \* `maxSpeed` : velocità massima del vento in m/s.

### 3.7 shike::app::model::session

#### 3.7.1 Informazioni sul package

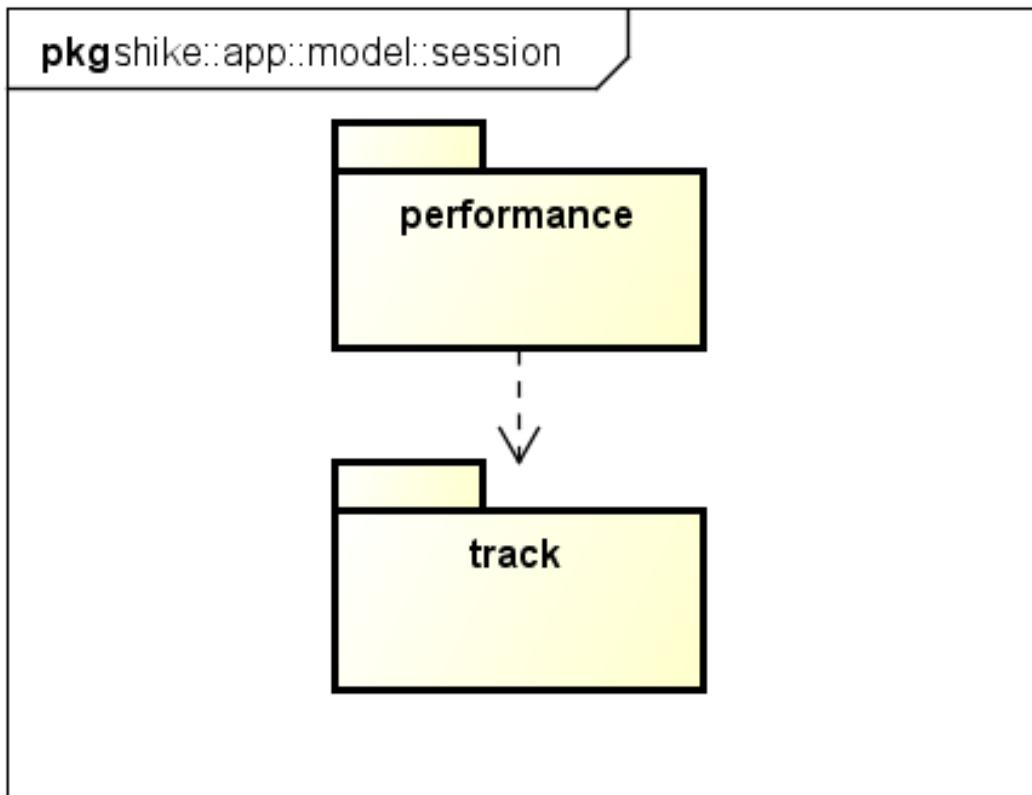


Figura 24: Diagramma di shike::app::model::session

- **Descrizione:** componente del *Model* che raccoglie i dati relativi ad una sessione di allenamento.
- **Componenti contenute**
  - `shike::app::model::session::track`
  - `shike::app::model::session::performance`

- **Componente padre:** shike::app::model
- **Interazioni con altri componenti**
  - shike::app::model::weather

### 3.8 shike::app::model::session::track

#### 3.8.1 Informazioni sul package

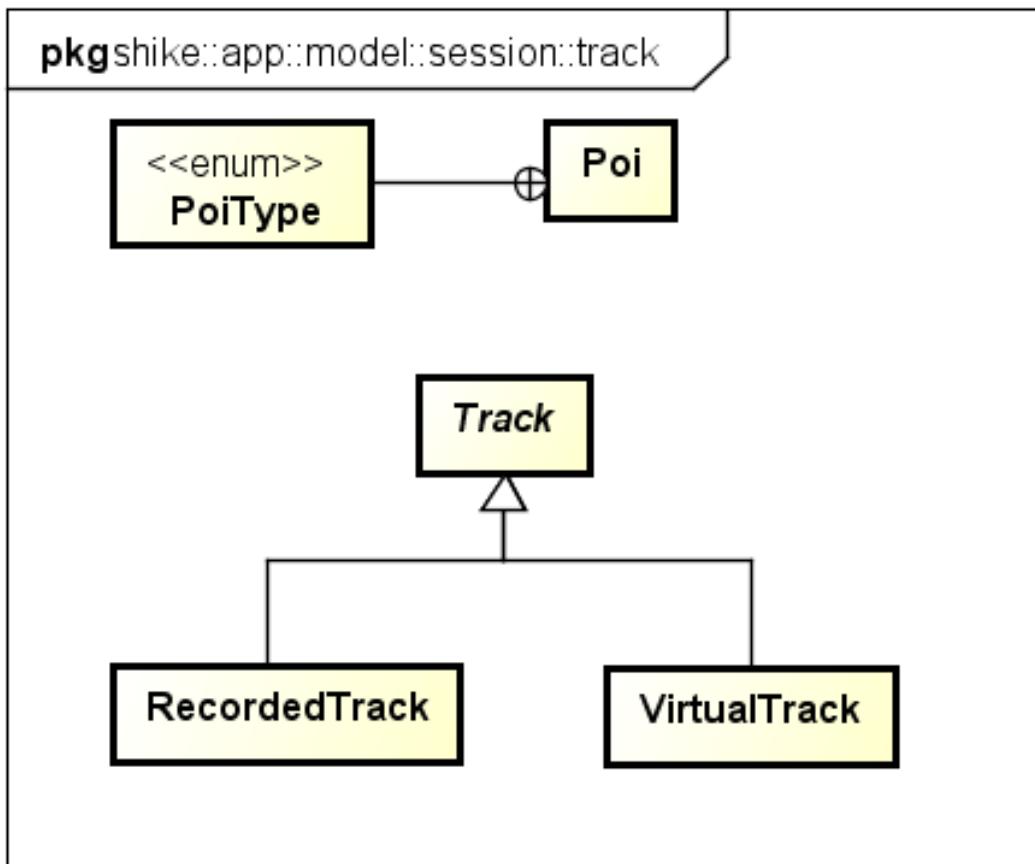


Figura 25: Diagramma di shike::app::model::session::track

- **Descrizione:** componente per la gestione dei dati relativi ad un percorso.
- **Componente padre:** shike::app::model::session

### 3.8.2 Classi

#### 3.8.2.1 shike::app::model::session::track::Poi

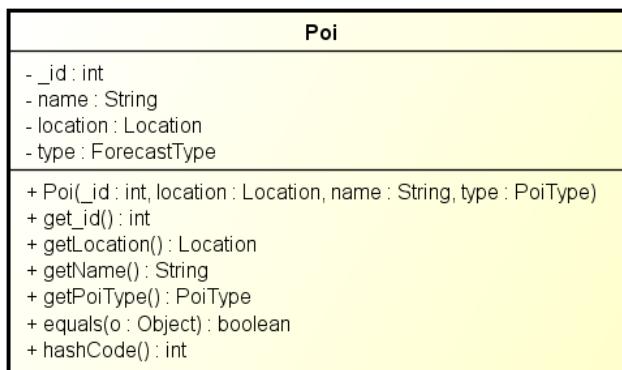


Figura 26: Diagramma di Poi

- **Tipo:** concreta
- **Descrizione:** classe che modella un POI.
- **Attributi:**
  - `_location` : Location  
Posizione del punto di interesse.
  - `_name` : String  
Nome del punto di interesse.
  - `_id` : int  
Identificatore univoco del POI.
  - `_type` : Poi.PoiType  
Tipologia di punto di interesse.
- **Metodi:**
  - `+getLocation() : Location`  
Restituisce la variabile `location`.
  - `+getName() : String`  
Restituisce la variabile `name`.
  - `+get_id() : int`  
Restituisce la variabile `_id`.
  - `+getType() : Poi.PoiType`  
Restituisce la variabile `type`.
  - `+hashcode( ) : int`  
*Override* che genera un codice `hash` considerando tutti gli attributi della classe.
  - `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.
- **Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.

### 3.8.2.2 shike::app::model::session::track::Track

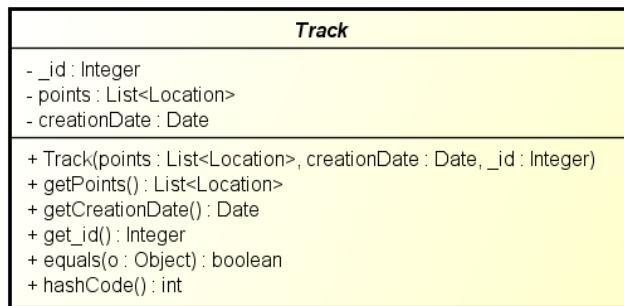


Figura 27: Diagramma di Track

- **Tipo:** astratta
- **Descrizione:** classe astratta che modella un percorso generico
- **Sottoclassi:**
  - shike::app::model::session::track::VirtualTrack;
  - shike::app::model::session::track::RecordedTrack;
- **Attributi:**
  - `_points : List<Location>`  
I punti di cui è composto il percorso.
  - `_creationDate : Date`  
Data di creazione del percorso.
  - `_id : Integer`  
Identificatore univoco del percorso (se è un percorso in fase di creazione vale ancora *null*).
- **Metodi:**
  - `+getPoints() : List<Location>`  
Restituisce la variabile `points`.
  - `+getCreationDate() : Date`  
Restituisce la variabile `creationDate`.
  - `+getId() : Integer`  
Restituisce la variabile `_id`.
  - `+Track( points : List<Location>, creationDate : Date, _id : Integer ) :`  
Costruttore della classe.  
**Argomenti:**
    - \* `points` : lista ordinata dei punti del percorso.
    - \* `creationDate` : data di creazione del percorso.
    - \* `_id` : id del percorso.
  - `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**

- \* `o` : oggetto da confrontare con `this`.
- `+hashcode()` : `int`  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe.

### 3.8.2.3 shike::app::model::session::track::VirtualTrack

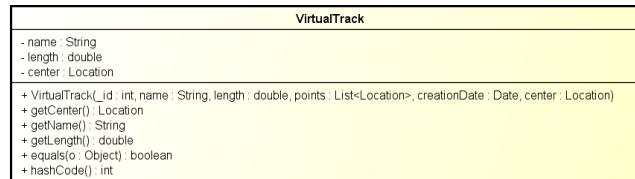


Figura 28: Diagramma di VirtualTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella un percorso scaricato dalla piattaforma web
- **Superclassi:**
  - shike::app::model::session::track::Track
- **Attributi:**
  - `_name` : `String`  
 Nome del percorso.
  - `_length` : `double`  
 Lunghezza del percorso (in metri).
  - `_center` : `Location`  
 Baricentro del percorso (utilizzato per le previsioni meteo).
- **Metodi:**
  - `+getName()` : `String`  
 Restituisce la variabile `name`.
  - `+getLength()` : `double`  
 Restituisce la variabile `length`.
  - `+getCenter()` : `Location`  
 Restituisce la variabile `center`.
  - `+VirtualTrack(_id : int, name : String, length : double, points : List<Location>, creationDate : Date, center : Location)` :  
 Costruttore completo della classe. Invoca il costruttore della superclasse.
- **Argomenti:**
  - \* `_id` : id del percorso.
  - \* `name` : nome del percorso.
  - \* `length` : lunghezza in metri del percorso.
  - \* `points` : lista ordinata dei punti del percorso.
  - \* `creationDate` : data di creazione del percorso.
  - \* `center` : baricentro del percorso.

- `+equals( o : Object ) : boolean`  
*OVERRIDING* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+hashCode() : int`  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe.

### 3.8.2.4 shike::app::model::session::track::RecordedTrack

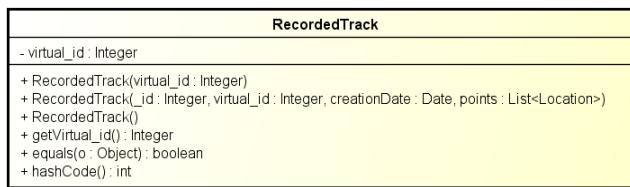


Figura 29: Diagramma di RecordedTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella un percorso creato durante una sessione
- **Superclassi:**
  - shike::app::model::session::track::Track
- **Attributi:**
  - `-virtual_id : Integer`  
 Id del *VirtualTrack* associato (*null* in caso di sessione libera).
- **Metodi:**
  - `+RecordedTrack( virtual_id : Integer ) :`  
 Costruttore a 1 parametro, utile per la creazione in diretta di un nuovo percorso.  
**Argomenti:**
    - \* `virtual_id` : id del percorso associato (può essere *null* nel caso di sessione libera).
  - `+equals( o : Object ) : boolean`  
*OVERRIDING* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
    - \* `o` : oggetto da confrontare con `this`.
  - `+RecordedTrack( _id : Integer, virtual_id : Integer, creationDate : Date, points : List<Location> ) :`  
 Costruttore completo della classe. Utile per ricreare percorsi già salvati nel *database*.  
**Argomenti:**
    - \* `_id` : id del percorso (*null* se è un nuovo percorso, non ancora salvato nel database).
    - \* `virtual_id` : id del *VirtualTrack* associato (*null* se non esiste).
    - \* `creationDate` : data di creazione del percorso.
    - \* `points` : lista ordinata dei punti del percorso.
  - `+hashCode() : int`  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe.

### 3.8.2.5 shike::app::model::session::track::Poi::PoiType

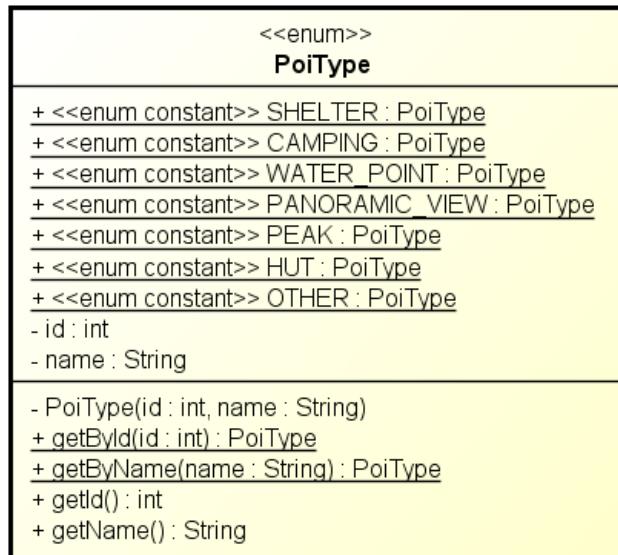


Figura 30: Diagramma di Poi::PoiType

- **Tipo:** enum
- **Descrizione:** enumeratore interno alla classe shike::app::model::session::track::Poi contenente le varie tipologie di POI
- **Attributi:**
  - –id : int  
Identificatore univoco della tipologia di POI.
  - –name : String  
Nome della tipologia di POI.
- **Metodi:**
  - +getId() : int  
Restituisce la variabile id.
  - +getName() : String  
Restituisce la variabile name.
  - +getById( id : int ) : PoiType  
Ritorna il tipo di POI con l'id inserito, o null se tale tipologia non esiste.

**Argomenti:**

  - \* id : id del tipo di POI desiderato.
  - +getByName( name : String ) : PoiType  
Ritorna il tipo di POI con il nome inserito, o null se tale tipologia non esiste. Se ci sono più tipologie con lo stesso nome, viene ritornata la prima dichiarata.

**Argomenti:**

  - \* name : nome della tipologia di POI desiderata.

### 3.9 shike::app::model::session::performance

#### 3.9.1 Informazioni sul package

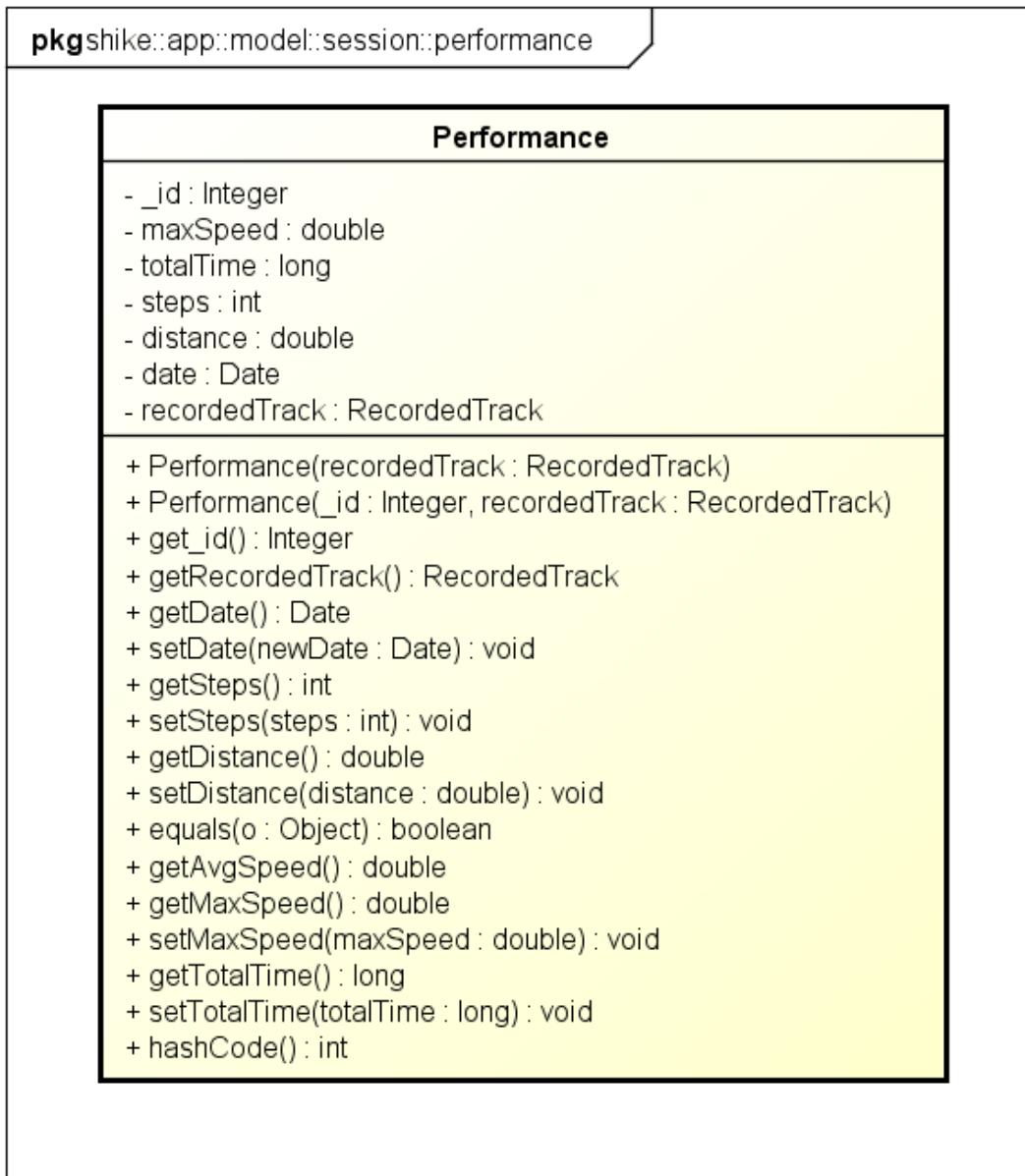


Figura 31: Diagramma di shike::app::model::session::performance

- **Descrizione:** componente che si occupa della gestione delle statistiche globali dell'utente e delle *performance* registrate in una sessione.
- **Componente padre:** shike::app::model::session
- **Interazioni con altri componenti**
  - shike::app::model::session::track

### 3.9.2 Classi

#### 3.9.2.1 shike::app::model::session::performance::Performance

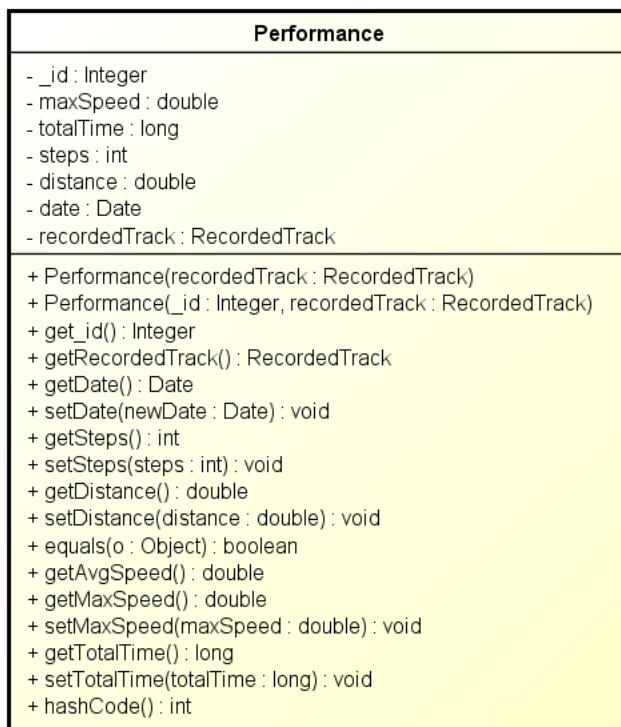


Figura 32: Diagramma di Performance

- **Tipo:** concreta
- **Descrizione:** classe che modella una performance dell'utente
- **Attributi:**
  - `_totalTime : long`  
Memorizza la durata totale della performance (in ms).
  - `_recordedTrack : RecordedTrack`  
Riferimento al percorso associato alla performance.
  - `_id : Integer`  
Id della performance. Vale `null` se la performance non è stata ancora salvata nel database.
  - `_date : Date`  
Data in cui è stata effettuata la performance.
  - `_maxSpeed : double`  
Memorizza il valore massimo di velocità rilevato (in m/s).
  - `_steps : int`  
Memorizza i passi effettuati.
  - `_distance : double`  
Memorizza la distanza percorsa (in metri).

- Metodi:

- `+getTotalTime() : long`

Restituisce la variabile `totalTime`.

- `+setTotalTime( totalTime : long ) : void`

**Argomenti:**

- \* `totalTime` : valore da impostare nella variabile `totalTime`.

- `+getRecordedTrack() : RecordedTrack`

Restituisce la variabile `recordedTrack`.

- `+setRecordedTrack( recordedTrack : RecordedTrack ) : void`

**Argomenti:**

- \* `recordedTrack` : valore da impostare nella variabile `recordedTrack`.

- `+getId() : Integer`

Restituisce la variabile `_id`.

- `+setId( id : Integer ) : void`

**Argomenti:**

- \* `_id` : valore da impostare nella variabile `_id`.

- `+getDate() : Date`

Restituisce la variabile `date`.

- `+ setDate( date : Date ) : void`

**Argomenti:**

- \* `date` : valore da impostare nella variabile `date`.

- `+getMaxSpeed() : double`

Restituisce la variabile `maxSpeed`.

- `+setMaxSpeed( maxSpeed : double ) : void`

**Argomenti:**

- \* `maxSpeed` : valore da impostare nella variabile `maxSpeed`.

- `+getSteps() : int`

Restituisce la variabile `steps`.

- `+setSteps( steps : int ) : void`

**Argomenti:**

- \* `steps` : valore da impostare nella variabile `steps`.

- `+getDistance() : double`

Restituisce la variabile `distance`.

- `+setDistance( distance : double ) : void`

**Argomenti:**

- \* `distance` : valore da impostare nella variabile `distance`.

- `+Performance( id : Integer, recordedTrack : RecordedTrack ) :`

Costruttore basilare della classe. Inizializza solo gli attributi `_id` e `recordedTrack`, per gli altri è necessario usare i *setter*.

**Argomenti:**

- \* `_id` : id della performance (può essere *null*).

- \* `recordedTrack` : percorso associato alla performance.

- `+getAvgSpeed() : double`

Ritorna la velocità media della *performance*.

- `+hashcode() : int`  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe (tranne per `recordedTrack`).
  - `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe (tranne per `recordedTrack`).
- Argomenti:**
- \* `o` : oggetto da confrontare con `this`.

### 3.10 shike::app::model::user

#### 3.10.1 Informazioni sul package

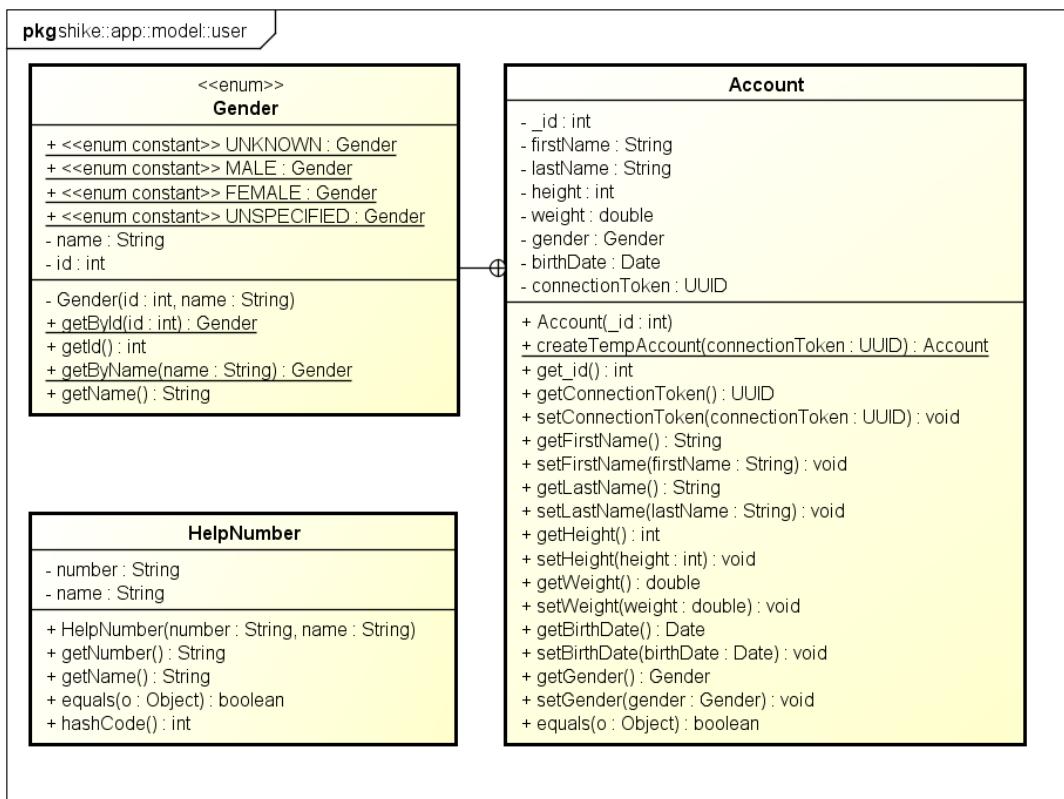


Figura 33: Diagramma di `shike::app::model::user`

- **Descrizione:** componente del *Model* per la raccolta delle informazioni personali dell'utente.
- **Componente padre:** `shike::app::model`
- **Interazioni con altri componenti**
  - `shike::app::model::session`
  - `shike::app::model::session::track`

### 3.10.2 Classi

#### 3.10.2.1 shike::app::model::user::Account

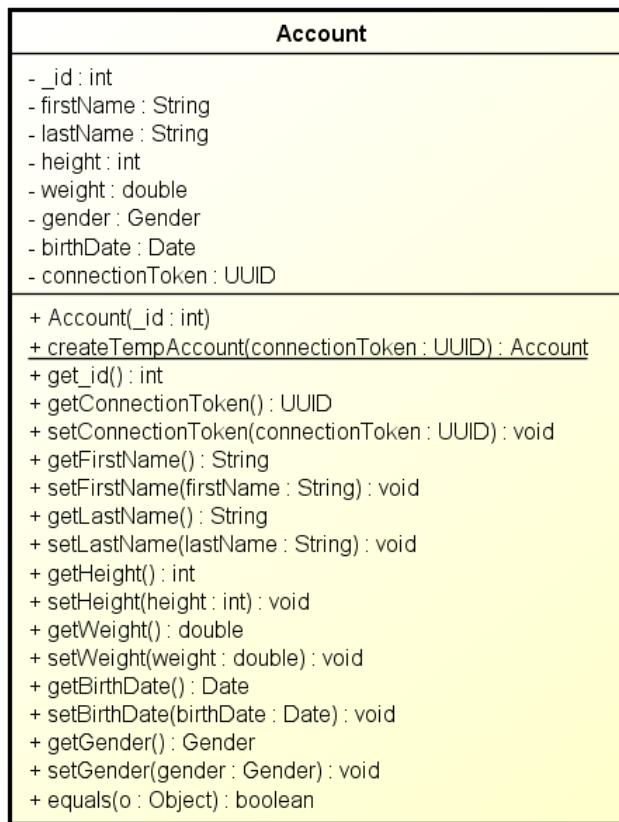


Figura 34: Diagramma di Account

- **Tipo:** concreta
- **Descrizione:** classe che modella l' *account* utente collegato al dispositivo.
- **Attributi:**
  - **`_firstName`** : `String`  
Il nome dell'utente.
  - **`_lastName`** : `String`  
Il cognome dell'utente.
  - **`_height`** : `int`  
Altezza dell'utente in cm.
  - **`_weight`** : `double`  
Peso dell'utente in kg.
  - **`_gender`** : `Account.Gender`  
Il sesso dell'utente.
  - **`_birthDate`** : `Date`  
Data di nascita dell'utente.

- `+connectionToken : UUID`  
*Token* di connessione alla piattaforma web.

- **Metodi:**

- `+getFirstName() : String`  
 Restituisce la variabile `firstName`.
  - Argomenti:**
    - \* `firstName` : valore da impostare nella variabile `firstName`.
- `+setFirstName( firstName : String ) : void`  
**Argomenti:**
  - \* `firstName` : valore da impostare nella variabile `firstName`.
- `+getLastName() : String`  
 Restituisce la variabile `lastName`.
  - Argomenti:**
    - \* `lastName` : valore da impostare nella variabile `lastName`.
- `+setLastName( lastName : String ) : void`  
**Argomenti:**
  - \* `lastName` : valore da impostare nella variabile `lastName`.
- `+getHeight() : int`  
 Restituisce la variabile `height`.
  - Argomenti:**
    - \* `height` : valore da impostare nella variabile `height`.
- `+setHeight( height : int ) : void`  
**Argomenti:**
  - \* `height` : valore da impostare nella variabile `height`.
- `+getWeight() : double`  
 Restituisce la variabile `weight`.
  - Argomenti:**
    - \* `weight` : valore da impostare nella variabile `weight`.
- `+setWeight( weight : double ) : void`  
**Argomenti:**
  - \* `weight` : valore da impostare nella variabile `weight`.
- `+getGender() : Account.Gender`  
 Restituisce la variabile `gender`.
  - Argomenti:**
    - \* `gender` : valore da impostare nella variabile `gender`.
- `+setGender( gender : Account.Gender ) : void`  
**Argomenti:**
  - \* `gender` : valore da impostare nella variabile `gender`.
- `+getBirthDate() : Date`  
 Restituisce la variabile `birthDate`.
  - Argomenti:**
    - \* `birthDate` : valore da impostare nella variabile `birthDate`.
- `+setBirthDate( birthDate : Date ) : void`  
**Argomenti:**
  - \* `birthDate` : valore da impostare nella variabile `birthDate`.
- `+getConnectionToken() : UUID`  
 Restituisce la variabile `connectionToken`.
  - Argomenti:**
    - \* `connectionToken` : valore da impostare nella variabile `connectionToken`.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.
  - Argomenti:**
    - \* `o` : oggetto da confrontare con `this`.
- `+hashcode( ) : int`  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe.

– `+createTempAccount( connectionToken : UUID ) : Account`  
 Crea un oggetto Account temporaneo, contenente solo il *token* di connessione (anch'esso temporaneo) alla piattaforma web. Viene utilizzato dopo il collegamento dell'account con la piattaforma web per mantenere le informazioni di collegamento fino alla prima sincronizzazione.

**Argomenti:**

\* `connectionToken` : token di connessione da associare all'account.

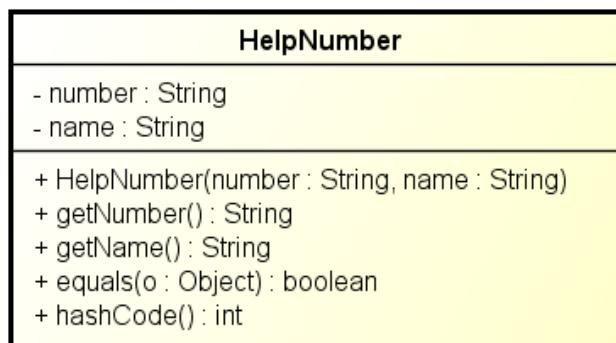
**3.10.2.2 shike::app::model::user::HelpNumber**

Figura 35: Diagramma di HelpNumber

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.
- **Attributi:**

- `–number : String`  
 Numero telefonico.
- `–name : String`  
 Nome associato al numero.

- **Metodi:**

- `+getNumber() : String`  
 Restituisce la variabile `number`.
- `+getName() : String`  
 Restituisce la variabile `name`.
- `+hashcode( ) : int`  
`Override` che genera un codice `hash` considerando tutti gli attributi della classe.
- `+HelpNumber( number : String, name : String ) :`  
 Costruttore della classe.

**Argomenti:**

\* `number` : numero di telefono.  
 \* `name` : nome associato al numero.

- `+equals( o : Object ) : boolean`  
`Overriding` che controlla l'uguaglianza per ogni singolo attributo della classe.

**Argomenti:**

\* `o` : oggetto da confrontare con `this`.

### 3.10.2.3 shike::app::model::user::Account::Gender

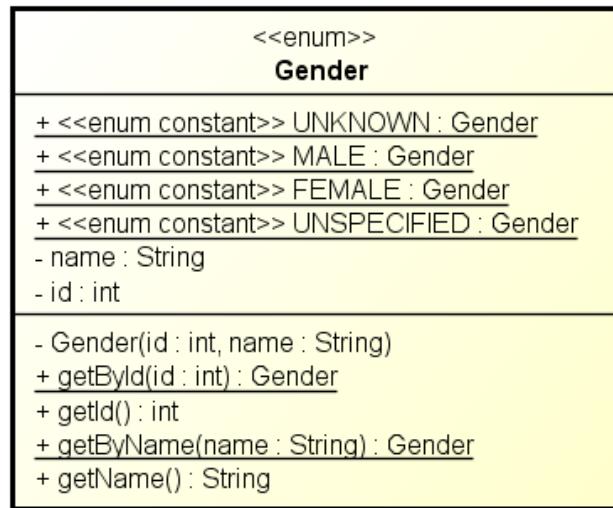


Figura 36: Diagramma di Account::Gender

- **Tipo:** enum
- **Descrizione:** enum che indica i possibili sessi indicati dallo standard ISO 5218
- **Attributi:**
  - **-name : String**  
Nome del sesso.
  - **-id : int**  
Identificatore del sesso (in accordo con ISO 5218).
- **Metodi:**
  - **+getName() : String**  
Restituisce la variabile name.
  - **+getId() : int**  
Restituisce la variabile id.
  - **+Gender( id : int, name : String ) :**  
Costruttore dell'enum.  
**Argomenti:**
    - \* **id** : id del sesso.
    - \* **name** : nome del sesso.
  - **+getById( id : int ) : Gender**  
Ritorna il sesso con l'id cercato.  
**Argomenti:**
    - \* **id** : id del sesso.
  - **+getByName( name : String ) : Gender**  
Ritorna il sesso con il nome cercato.  
**Argomenti:**
    - \* **name** : nome del sesso.

### 3.11 shike::app::model::dao

#### 3.11.1 Informazioni sul package

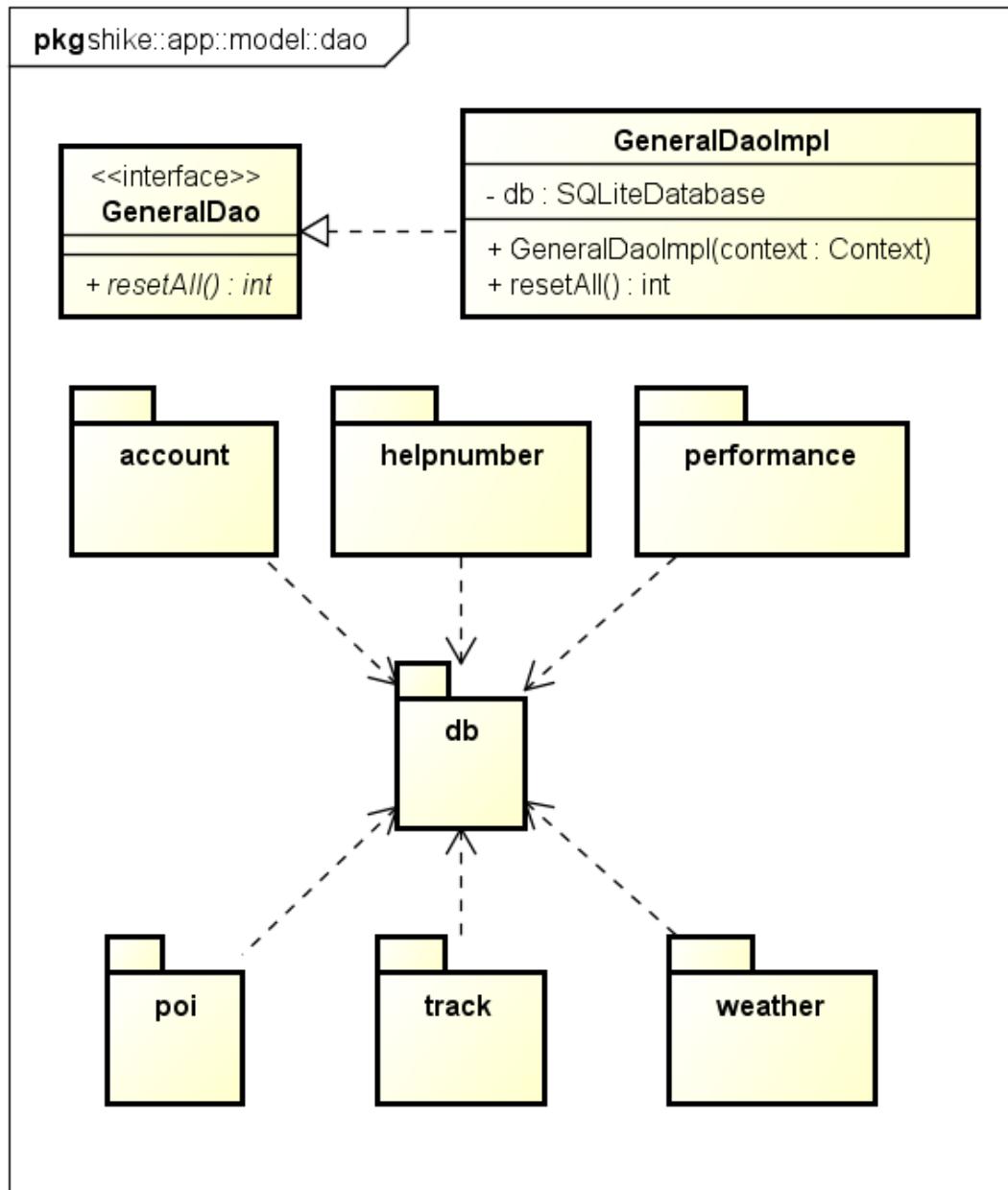


Figura 37: Diagramma di shike::app::model::dao

- **Descrizione:** componente del *Model* adatta all'interfacciamento con la base di dati dell'applicazione.
- **Componenti contenute**
  - `shike::app::model::dao::account`

- shike::app::model::dao::helpnumber
- shike::app::model::dao::performance
- shike::app::model::dao::poi
- shike::app::model::dao::track
- shike::app::model::dao::weather
- shike::app::model::dao::db

- **Componente padre:** shike::app::model

- **Interazioni con altri componenti**

- shike::app::model::weather
- shike::app::model::session
- shike::app::model::user

### 3.11.2 Classi

#### 3.11.2.1 shike::app::model::dao::GeneralDao

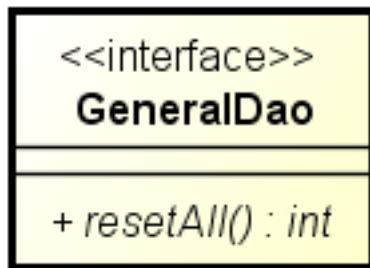


Figura 38: Diagramma di GeneralDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce un metodo per la cancellazione di tutti i dati utente dal database (Non cancella i dati delle tabelle costanti, come **genders** e **forecasttypes**).
- **Implementata da:**
  - shike::app::model::dao::GeneralDaoImpl
- **Metodi:**
  - **+resetAll() : int**  
Elimina tutti i dati delle tabelle contenenti dati creati dall'utente o scaricati durante una sincronizzazione. Ritorna il numero di righe totali cancellate.

### 3.11.2.2 shike::app::model::dao::GeneralDaoImpl

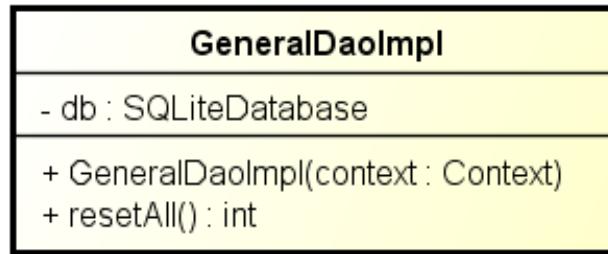


Figura 39: Diagramma di GeneralDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di shike::app::model::dao::GeneralDao.
- **Implementa:**
  - shike::app::model::dao::GeneralDao:
- **Metodi:**
  - `+GeneralDaoImpl( context : Context )`:  
Costruttore della classe.  
**Argomenti:**  
\* `context` : contesto dell'applicazione.

## 3.12 shike::app::model::dao::account

### 3.12.1 Informazioni sul package

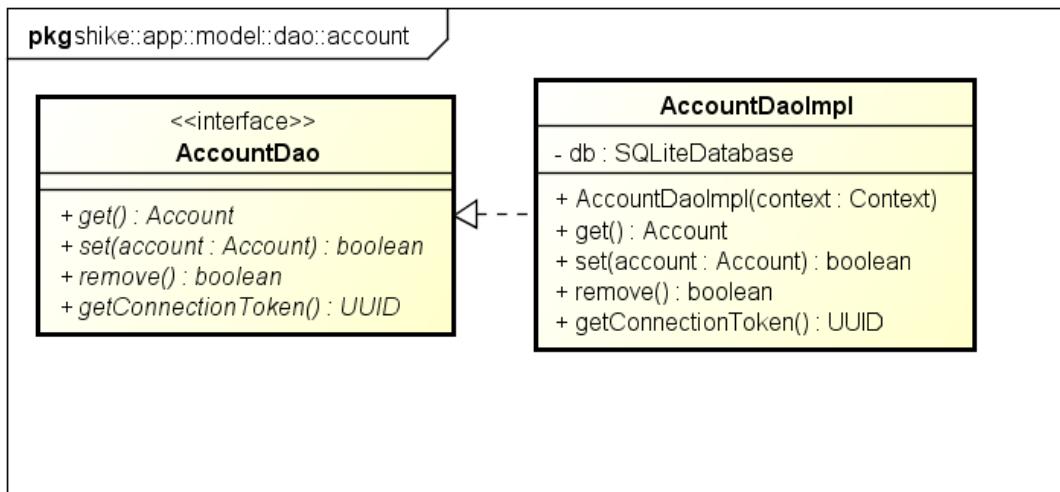


Figura 40: Diagramma di shike::app::model::dao::account

- **Descrizione:** componente che modella il *design pattern* DAO relativo all' *account* associato all'applicazione.
- **Componente padre:** shike::app::model::dao

### 3.12.2 Classi

#### 3.12.2.1 shike::app::model::dao::account::AccountDao

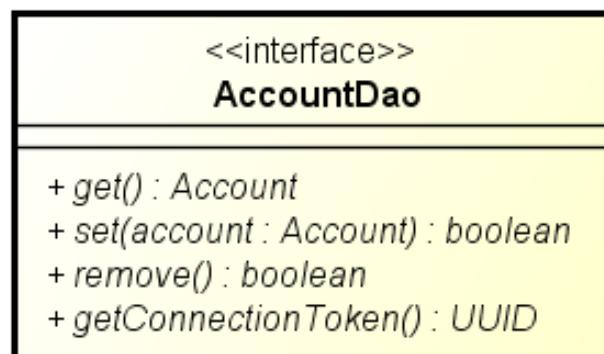


Figura 41: Diagramma di AccountDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi di interfacciamento con il *database* relativamente all' *account* associato all'applicazione.
- **Implementata da:**
  - shike::app::model::dao::account::AccountDaoImpl
- **Metodi:**
  - **+get( ) : Account**  
Ritorna l'unico *account* salvato nel *database*.
  - **+set( account : Account ) : boolean**  
Aggiorna l'*account* salvato nel *database*. Ritorna *true* se l'operazione ha avuto successo, *false* altrimenti.
  - Argomenti:**
    - \* **account** : il nuovo *account* da inserire al posto di quello presente nel database.
  - **+remove( ) : boolean**  
Rimuove l'*account* salvato nel *database*. Ritorna *true* se l'operazione ha avuto successo, *false* se è fallita o se la tabella è già vuota.
  - **+getConnectionToken( ) : UUID**  
Ritorna il *token* di connessione associato all'*account*, o null se non è stato collegato alcun *account* all'applicazione.

### 3.12.2.2 shike::app::model::dao::account::AccountDaoImpl

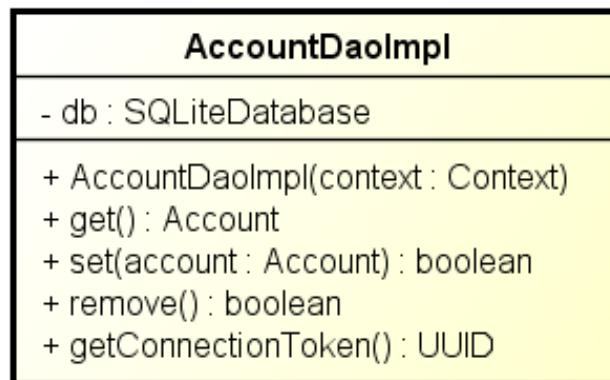


Figura 42: Diagramma di AccountDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di shike::app::model::dao::account::**AccountDao**.
- **Implementa:**
  - shike::app::model::dao::account::**AccountDao**:
- **Attributi:**
  - **–db** : **SQLiteDatabase**  
Riferimento al *database* dell'applicazione.
- **Metodi:**
  - **+AccountDaoImpl( context : Context )**:  
Costruttore della classe.  
**Argomenti:**
    - \* **context** : contesto dell'applicazione, necessario per inizializzare l'attributo **db**.

### 3.13 shike::app::model::dao::helpnumber

#### 3.13.1 Informazioni sul package

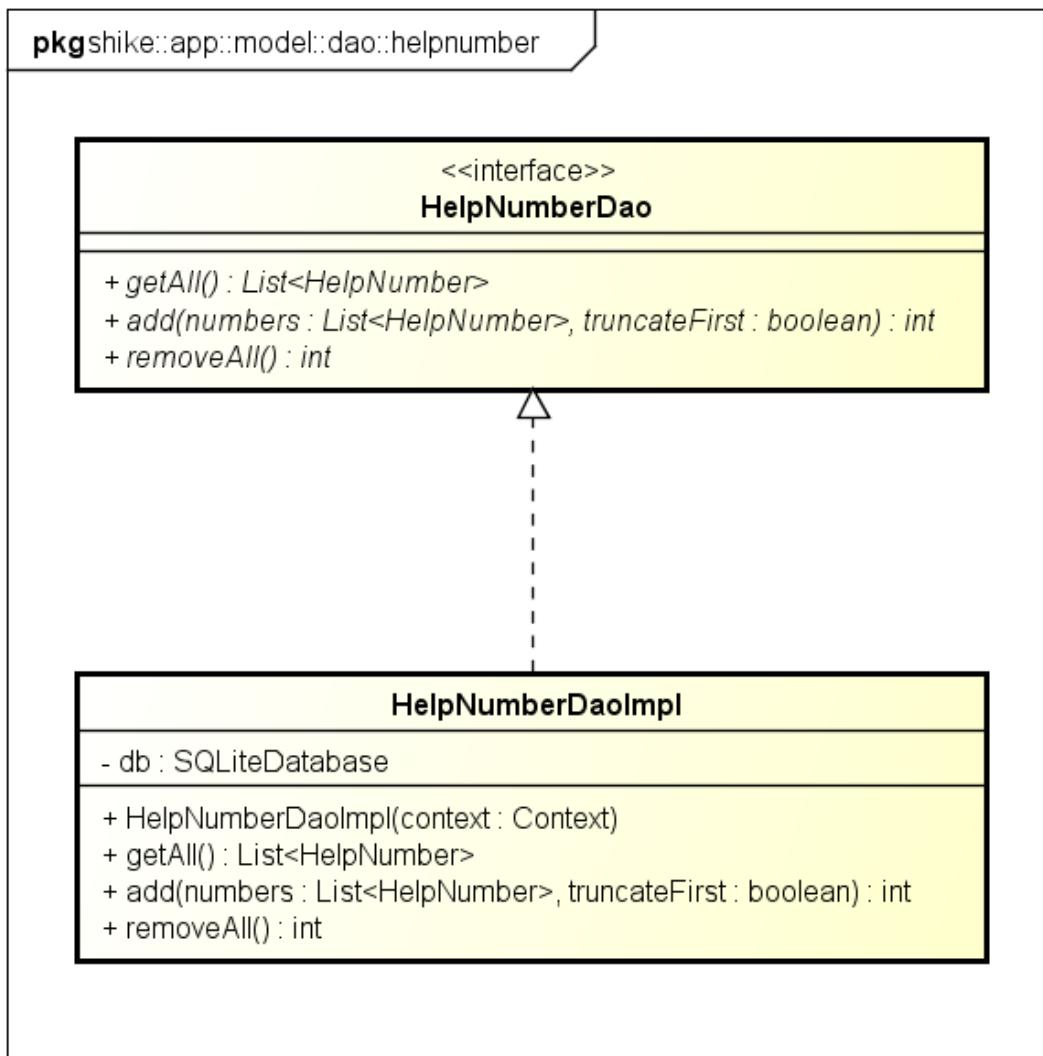


Figura 43: Diagramma di shike::app::model::dao::helpnumber

- **Descrizione:** componente che modella il *design pattern* DAO relativo ai numeri di soccorso dell'utente dell'applicazione
- **Componente padre:** shike::app::model::dao

### 3.13.2 Classi

#### 3.13.2.1 shike::app::model::dao::helpnumber::HelpNumberDao

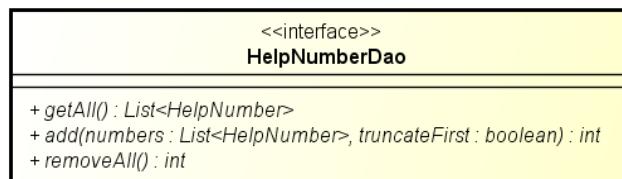


Figura 44: Diagramma di HelpNumberDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per gestire la tabella 'helpnumbers' del *database*.
- **Implementata da:**
  - shike::app::model::dao::helpnumber::HelpNumberDaoImpl
- **Metodi:**
  - `+remove( number : String ) : boolean`  
Rimuove dal *database* un numero di soccorso.  
**Argomenti:**  
\* `number` : numero di telefono da cercare ed eliminare.
  - `+removeAll() : int`  
Svuota la tabella 'helpnumbers'.
  - `+add( numbers : List<HelpNumber>, truncateFirst : boolean ) : int`  
Aggiunge una lista di numeri al *database*, sovrascrivendo eventuali numeri uguali.  
**Argomenti:**  
\* `numbers` : lista dei numeri di soccorso da aggiungere.  
\* `truncateFirst` : indica se svuotare la tabella prima dell'inserimento.
  - `+getAll() : List<HelpNumber>`  
Ritorna una lista di tutti i numeri di soccorso salvati nel *database*.

#### 3.13.2.2 shike::app::model::dao::helpnumber::HelpNumberDaoImpl

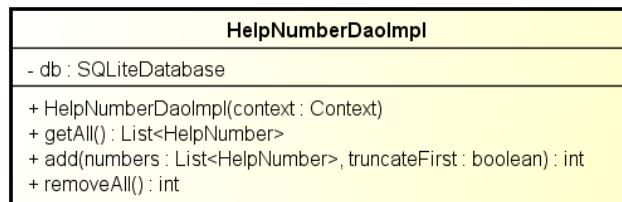


Figura 45: Diagramma di HelpNumberDaoImpl

- **Tipo:** concreta

- **Descrizione:** implementazione di shike::app::model::dao::helpnumber::HelpNumberDao
- **Implementa:**
  - shike::app::model::dao::helpnumber::HelpNumberDao
- **Attributi:**
  - **-db : SQLiteDatabase**  
Riferimento al *database* dell'applicazione.
- **Metodi:**
  - **+HelpNumberDaoImpl( ) :**  
Costruttore della classe.

### 3.14 shike::app::model::dao::performance

#### 3.14.1 Informazioni sul package

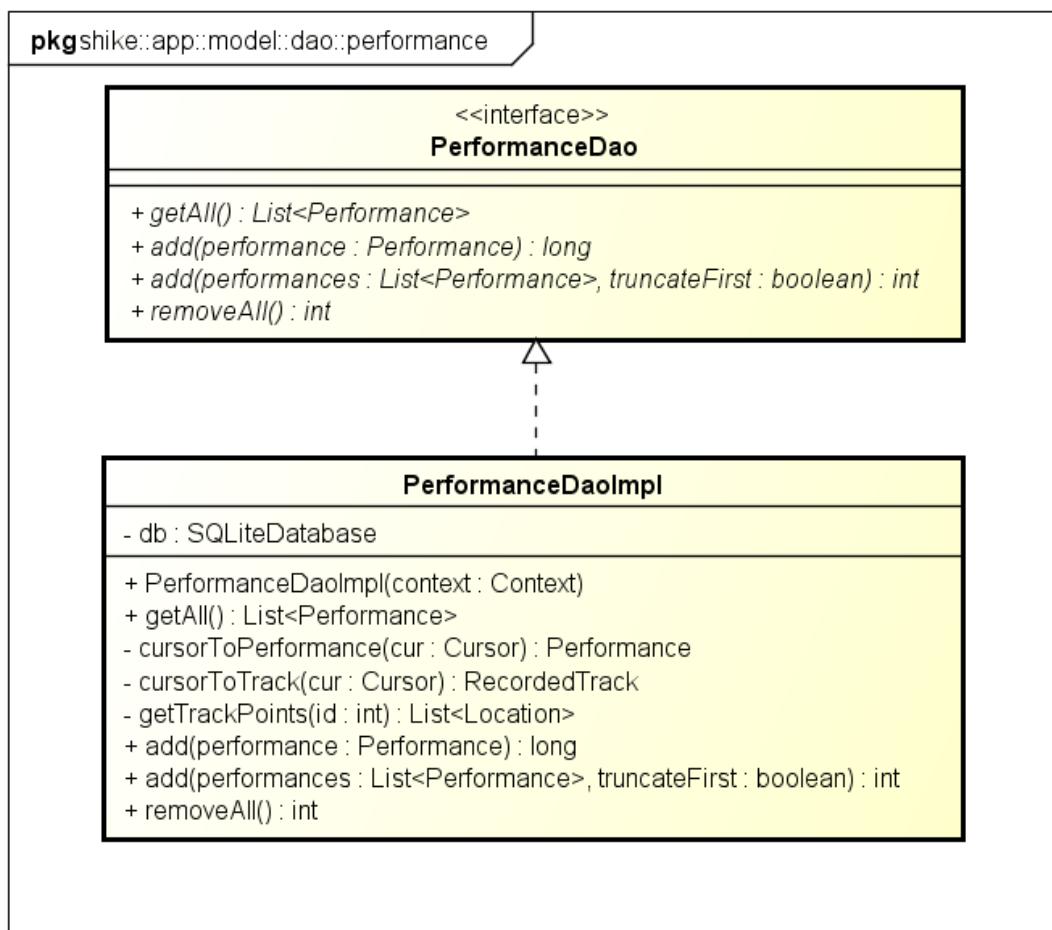


Figura 46: Diagramma di shike::app::model::dao::performance

- **Descrizione:** componente che modella il *design pattern* DAO relativo alle *performance* dell'utente.

- **Componente padre:** shike::app::model::dao

### 3.14.2 Classi

#### 3.14.2.1 shike::app::model::dao::performance::PerformanceDaoImpl

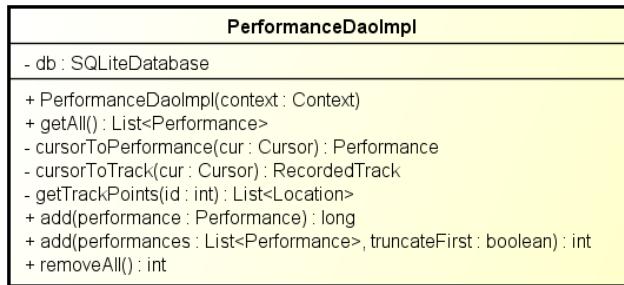


Figura 47: Diagramma di PerformanceDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di shike::app::model::dao::performance::**PerformanceDao**.
- **Implementa:**
  - shike::app::model::dao::performance::**PerformanceDao**:
- **Attributi:**
  - **–db** : **SQLiteDatabase**  
Riferimento al *database* dell'applicazione.
- **Metodi:**
  - **+RecordedTrackDaoImpl( context : Context )**:  
Costruttore necessario per ottenere il riferimento al *database*.  
**Argomenti:**
    - \* **context** : contesto dell'applicazione.
  - **–getTrackPoints( id : int ) : List<Location>**  
Ritorna la lista ordinata dei punti del percorso.  
**Argomenti:**
    - \* **id** : id del percorso.
  - **–cursorToTrack( cur : Cursor ) : RecordedTrack**  
Crea e ritorna un *RecordedTrack* a partire da una riga della tabella corrispondente.  
**Argomenti:**
    - \* **cur** : cursore che rappresenta una riga di 'recordedtracks'.
  - **+cursorToPerformance( cur : Cursor ) : Performance**  
Crea e ritorna una performance a partire da una riga della tabella corrispondente (incluso il percorso associato).  
**Argomenti:**
    - \* **cur** : cursore che rappresenta una riga di 'recordedtracks'.

### 3.14.2.2 shike::app::model::dao::performance::PerformanceDao

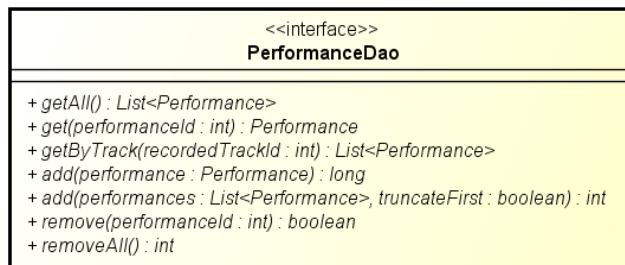


Figura 48: Diagramma di PerformanceDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per gestire la tabella 'performance'
- **Implementata da:**
  - shike::app::model::dao::performance::**PerformanceDaoImpl**:
- **Metodi:**
  - **+get( performanceId : int ) : Performance**  
Ritorna la *performance* con l'id cercato, *null* se tale *performance* non esiste.  
**Argomenti:**  
\* *performanceId* : id della *performance* desiderata.
  - **+getAll( ) : List<Performance>**  
Ritorna tutte le *performance* salvate nel *database*.
  - **+getByTrack( recordedTrackId : int ) : List<Performance>**  
Ritorna tutte le *performance* associate ad un certo percorso.  
**Argomenti:**  
\* *recordedTrackId* : id del percorso associato alle *performance* desiderate.
  - **+add( performance : Performance ) : long**  
Aggiunge nel *database* una *performance*. Nel caso di conflitti, la *performance* nuova sovrascrive quella precedente. Ritorna l'id della *performance* aggiunta.  
**Argomenti:**  
\* *performance* : *performance* da inserire nel database.
  - **+add( performances : List<Performances>, truncateFirst : boolean ) : int**  
Aggiunge al *database* più *performance* (sovrascrivendo in caso di conflitti). Ritorna il numero di *performance* aggiunte con successo.  
**Argomenti:**  
\* *performances* : lista delle *performance* da aggiungere.  
\* *truncateFirst* : true se si desidera svuotare la tabella prima dell'inserimento, false altrimenti.
  - **+remove( performanceId : int ) : boolean**  
Rimuove una *performance* dal *database*. Ritorna *true* se l'eliminazione è avvenuta con successo, *false* altrimenti.  
**Argomenti:**  
\* *performanceId* : id della *performance* da rimuovere.

- `+removeAll() : int`  
Rimuove tutte le *performance* dal *database*. Ritorna il numero di *performance* eliminate.

### 3.14.2.3 shike::app::model::dao::performance::PerformanceDaoImpl

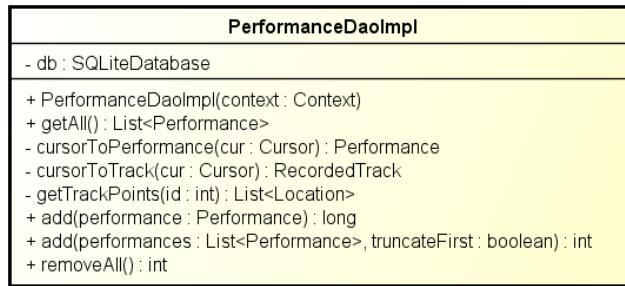


Figura 49: Diagramma di PerformanceDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di shike::app::model::dao::performance::**PerformanceDao**
- **Implementa:**
  - shike::app::model::dao::performance::**PerformanceDao**:
- **Attributi:**
  - `_db : SQLiteDatabase`  
Riferimento al *database*.
- **Metodi:**
  - `+PerformanceDaoImpl( ) :`  
Costruttore della classe.
  - `-cursorToPerformance( ) : Performance`  
Costruisce un oggetto *Performance* a partire da una riga della rispettiva tabella.

### 3.14.2.4 shike::app::model::dao::performance::PerformanceDao



Figura 50: Diagramma di PerformanceDao

- **Tipo:** interfaccia

- **Descrizione:** interfaccia che fornisce i metodi per la gestione delle performance dell'utente salvate nel database.

- **Implementata da:**

- shike::app::model::dao::performance::**PerformanceDaoImpl**:

- **Metodi:**

- **+removeAll( ) : int**

Rimuove tutte le performance dal database. Ritorna il numero di performance cancellate.

- **+add( performances : List<Performance>, truncateFirst : boolean ) : int**

Aggiunge al database più performance (sovrascrivendo in caso di conflitti). Ritorna il numero di performance aggiunte con successo.

**Argomenti:**

- \* **performances** : performance da aggiungere al database.

- \* **truncateFirst** : **true** se si desidera svuotare completamente la tabella prima di effettuare l'inserimento, **false** altrimenti.

- **+add( performance : Performance ) : long**

Aggiunge nel database una performance. Nel caso di conflitti, la performance nuova sovrascrive quella precedente. Ritorna l'id della performance aggiunta.

**Argomenti:**

- \* **performance** : performance da aggiungere al database.

- **+getAll( ) : List<Performance>**

Ritorna tutte le performance salvate nel database.

### 3.15 shike::app::model::dao::poi

#### 3.15.1 Informazioni sul package

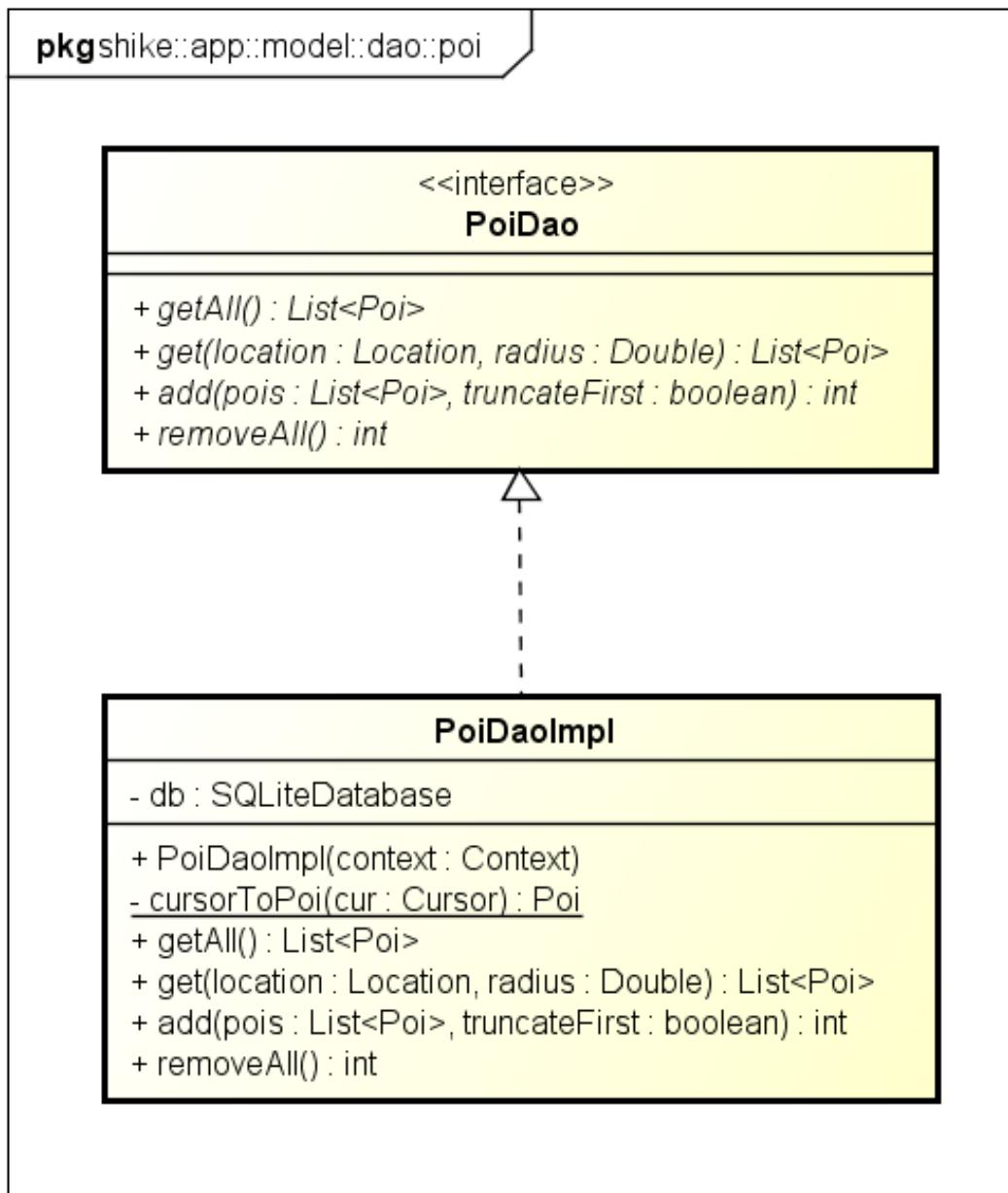


Figura 51: Diagramma di shike::app::model::dao::poi

- **Descrizione:** componente che modella il *design pattern* DAO relativo ai punti di interesse.
- **Componente padre:** shike::app::model::dao

### 3.15.2 Classi

#### 3.15.2.1 shike::app::model::dao::poi::PoiDaoImpl

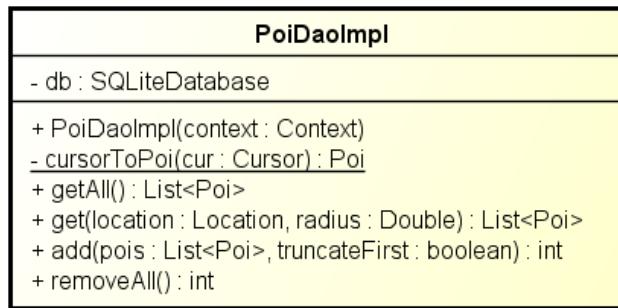


Figura 52: Diagramma di PoiDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di shike::app::model::dao::poi::**PoiDao**
- **Implementa:**
  - shike::app::model::dao::poi::**PoiDao**:
- **Attributi:**
  - **–db : SQLiteDatabase**  
Riferimento al *database* interno.
- **Metodi:**
  - **+PoiDaoImpl( context : Context ) :**  
Costruttore della classe, serve a ottenere il riferimento al *database*.  
**Argomenti:**
    - \* **context** : contesto dell'applicazione.
  - **–cursorToPoi( cur : Cursor ) : Poi**  
Costruisce e ritorna un POI a partire da una riga della tabella 'pois'.  
**Argomenti:**
    - \* **cur** : cursore che rappresenta una riga della tabella 'pois'.

#### 3.15.2.2 shike::app::model::dao::poi::PoiDao

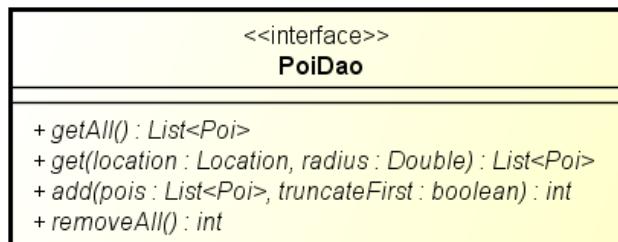


Figura 53: Diagramma di PoiDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per gestire la tabella 'pois'
- **Implementata da:**
  - shike::app::model::dao::poi::**PoiDaoImpl**:
- **Metodi:**
  - **+getAll()** : **List<Poi>**  
Ritorna la lista di tutti i POI salvati nel *database*.
  - **+get( location : Location, radius : Double )** : **List<Poi>**  
Ritorna tutti i POI a un certo raggio di distanza da una posizione geografica, ordinati per distanza crescente.  
**Argomenti:**
    - \* **location** : posizione di riferimento.
    - \* **radius** : raggio di distanza massimo di ricerca (in metri). Se è 0 o *null*, vengono ritornati tutti i POI.
  - **+add( pois : List<Poi>, truncateFirst : boolean )** : **int**  
Aggiunge più POI al *database*. Ritorna il numero di POI aggiunti con successo.  
**Argomenti:**
    - \* **pois** : lista dei POI da aggiungere.
    - \* **truncateFirst** : *true* se si desidera prima svuotare la tabella 'pois', *false* altrimenti.
  - **+removeAll()** : **int**  
Rimuove tutti i POI salvati nel *database*. Ritorna il numero di POI eliminati.

### 3.16 shike::app::model::dao::track

#### 3.16.1 Informazioni sul package

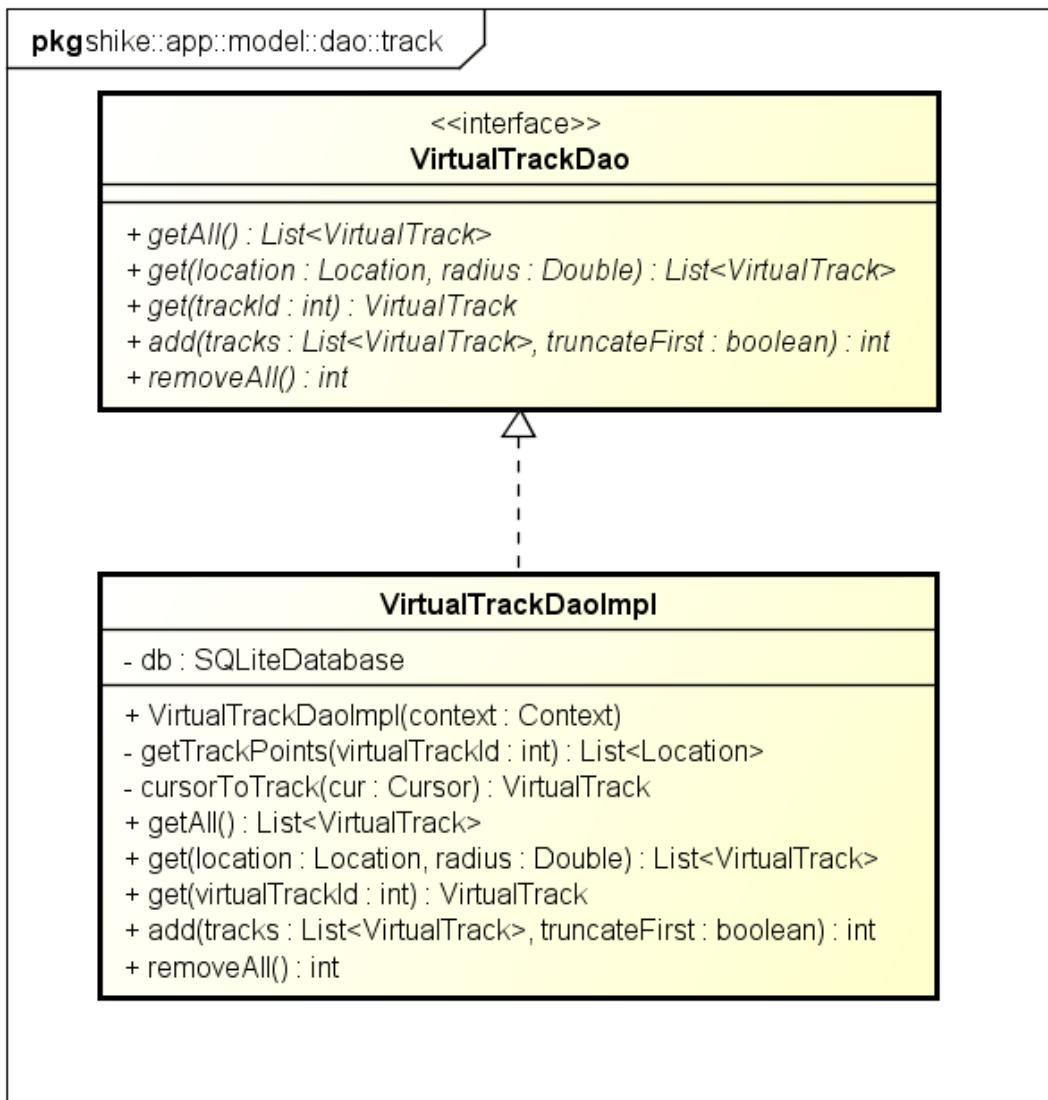


Figura 54: Diagramma di shike::app::model::dao::track

- **Descrizione:** componente che modella il *design pattern* DAO relativo ai percorsi, sia quelli scaricati dalla parte web, sia quelli creati *ex novo* dall'applicazione.
- **Componente padre:** shike::app::model::dao

### 3.16.2 Classi

#### 3.16.2.1 shike::app::model::dao::track::VirtualTrackDaoImpl

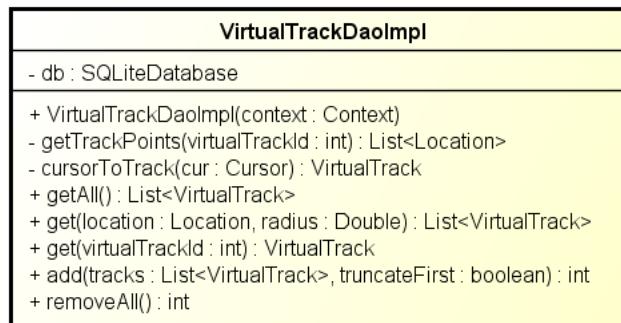


Figura 55: Diagramma di VirtualTrackDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di shike::app::model::dao::track::VirtualTrackDao
- **Implementa:**
  - shike::app::model::dao::track::VirtualTrackDao
- **Attributi:**
  - **–db : SQLiteDatabase**  
Riferimento al *database* dell'applicazione.
- **Metodi:**
  - **+VirtualTrackDaoImpl( context : Context ) :**  
Asdasdsad.  
**Argomenti:**
    - \* **context** : il contesto dell'applicazione.
  - **–getTrackPoints( id : int ) : List<Location>**  
Ritorna tutti i punti che compongono un percorso, in sequenza ordinata.  
**Argomenti:**
    - \* **id** : id del percorso.
  - **–cursorToTrack( cur : Cursor ) : VirtualTrack**  
Crea e ritorna un *VirtualTrack* a partire da una riga della tabella 'virtualtracks'.  
**Argomenti:**
    - \* **cur** : cursore che rappresenta una riga di 'virtualtracks'.

### 3.16.2.2 shike::app::model::dao::track::VirtualTrackDao

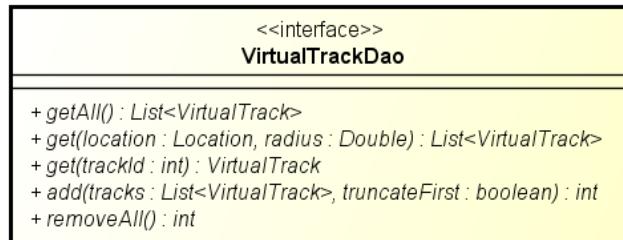


Figura 56: Diagramma di VirtualTrackDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per la gestione dei percorsi scaricati dal web e salvati nel *database*.
- **Implementata da:**
  - shike::app::model::dao::track::VirtualTrackDaoImpl
- **Metodi:**
  - **+getAll()** : `List<VirtualTrack>`  
Ritorna tutti i percorsi di tipo T salvati nel *database*.
  - **+get( location : Location, radius : Double )** : `List<VirtualTrack>`  
Ritorna tutti i percorsi entro una certa distanza da un punto geografico.
  - Argomenti:**
    - \* `location` : punto geografico di ricerca.
    - \* `radius` : raggio di distanza massimo (in metri) da location entro cui cercare. Se è 0 o null, il limite non viene considerato.
  - **+get( trackId : int )** : `VirtualTrack`  
Ritorna il percorso con l'id inserito, o `null` se tale percorso non esiste.
  - Argomenti:**
    - \* `trackId` : id del percorso desiderato.
  - **+add( tracks : List<VirtualTrack>, truncateFirst : boolean )** : `long`  
Aggiunge più percorsi al *database*. Ritorna il numero di percorsi aggiunti con successo.
  - Argomenti:**
    - \* `tracks` : percorsi da aggiungere.
    - \* `truncateFirst` : `true` se si desidera prima troncare la tabella dei percorsi, `false` altrimenti.
  - **+removeAll()** : `int`  
Rimuove tutti i percorsi dal *database*. Ritorna il numero di percorsi eliminati.

### 3.17 shike::app::model::dao::weather

#### 3.17.1 Informazioni sul package

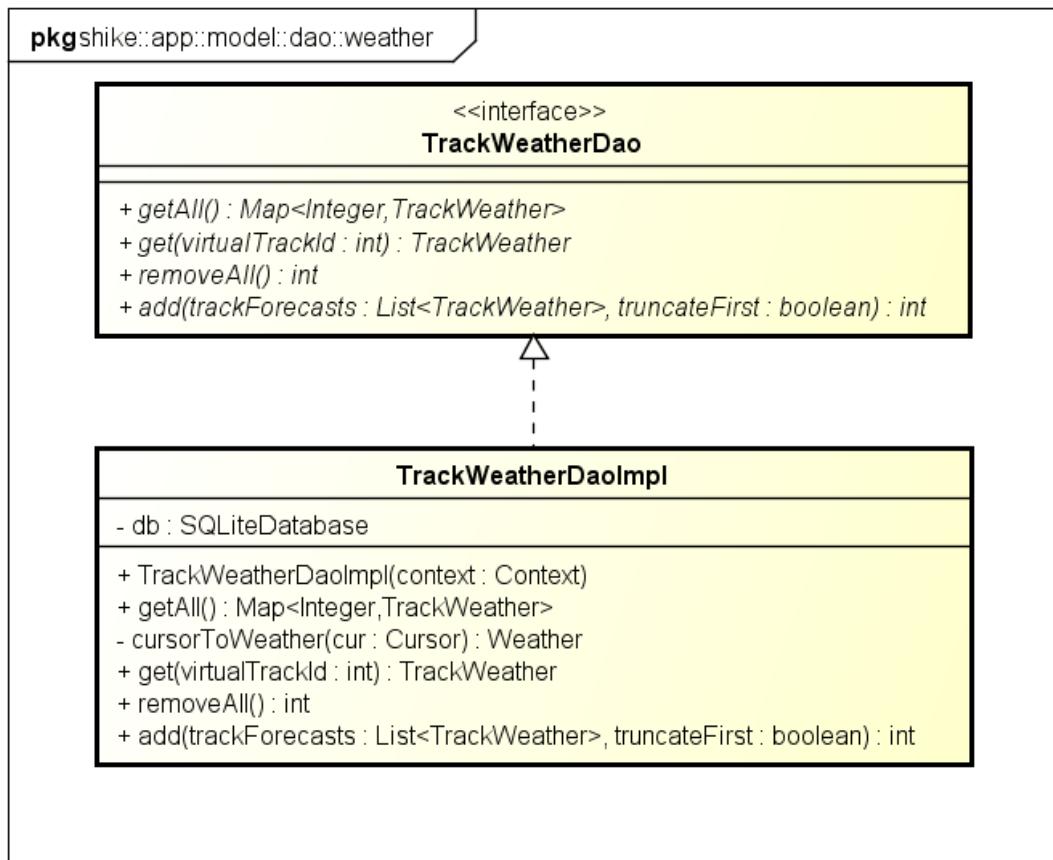


Figura 57: Diagramma di shike::app::model::dao::weather

- **Descrizione:** componente che modella il *design pattern* DAO relativo alle previsioni meteo dei percorsi scaricati dalla parte web
- **Componente padre:** shike::app::model::dao

#### 3.17.2 Classi

##### 3.17.2.1 shike::app::model::dao::weather::TrackWeatherDao

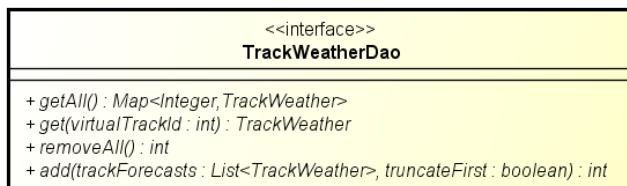


Figura 58: Diagramma di TrackWeatherDao

- **Tipo:** interfaccia
  - **Descrizione:** interfaccia che fornisce i metodi per la gestione delle previsioni meteo dei percorsi
  - **Implementata da:**
    - shike::app::model::dao::weather::**TrackWeatherDaoImpl**:
  - **Metodi:**
    - **+getAll()** : Map<Integer, TrackWeather>  
Ritorna tutte le previsioni salvate sul dispositivo, in una mappa con l'id del percorso associato come chiave.
    - **+get( virtualTrackId : int )** : TrackWeather  
Ritorna le previsioni meteo per un determinato percorso, o null se il percorso inserito non esiste.

**Argomenti:**

    - \* **virtualTrackId** : id del percorso.
  - **+removeAll()** : int  
Rimuove le previsioni meteo associate a un certo percorso, ritorna il numero di previsioni singole rimosse.
  - **+add( forecasts : TrackWeather, truncateFirst : boolean )** : int  
Aggiunge (rimpiazzando in caso di conflitti) al *database* delle previsioni meteo associate ad un percorso.
- Argomenti:**
- \* **forecasts** : le previsioni da aggiungere, associate ad un determinato percorso.
  - \* **truncateFirst** : se impostato a true elimina prima tutte le previsioni associate al percorso, altrimenti si limita ad effettuare un rimpiazzo.

### 3.17.2.2 shike::app::model::dao::weather::TrackWeatherDaoImpl

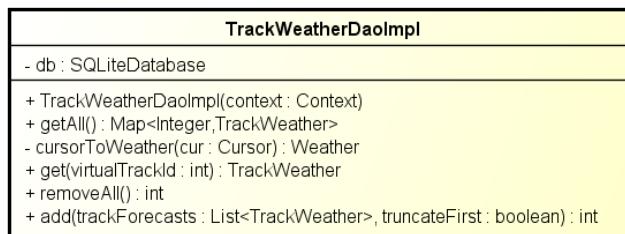


Figura 59: Diagramma di TrackWeatherDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di shike::app::model::dao::weather::**TrackWeatherDao**
- **Implementa:**
  - shike::app::model::dao::weather::**TrackWeatherDao**:
- **Attributi:**

- `-db : SQLiteDatabase`  
Riferimento al *database*.

- **Metodi:**

- `+TrackWeatherDaoImpl( context : Context ) :`  
Costruttore della classe, necessario per ottenere il riferimento al *database*.

**Argomenti:**

- \* `context` : contesto dell'applicazione.

- `-cursorToWeather( cur : Cursor ) : Weather`

Metodo di utilità interno che crea e ritorna un oggetto di tipo *Weather* a partire da una riga della tabella 'weatherforecasts'.

**Argomenti:**

- \* `cur` : cursore che indica una riga della tabella 'weatherforecasts'.

### 3.18 shike::app::model::dao::db

#### 3.18.1 Informazioni sul package

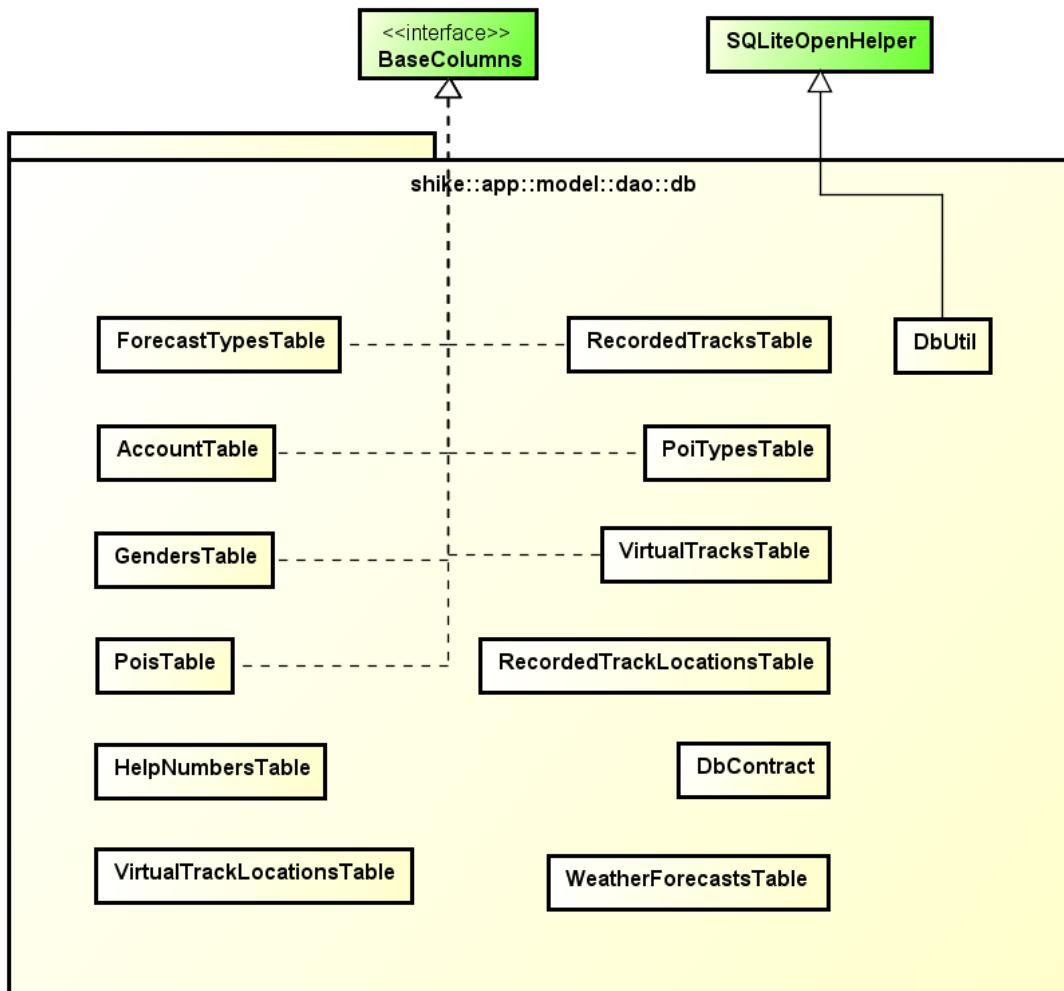


Figura 60: Diagramma di `shike::app::model::dao::db`

- **Descrizione:** componente contenente tutte le classi che permettono ai DAO l'interfacciamento con il database.
- **Componente padre:** shike::app::model::dao

### 3.18.2 Classi

#### 3.18.2.1 shike::app::model::dao::db::AccountTable

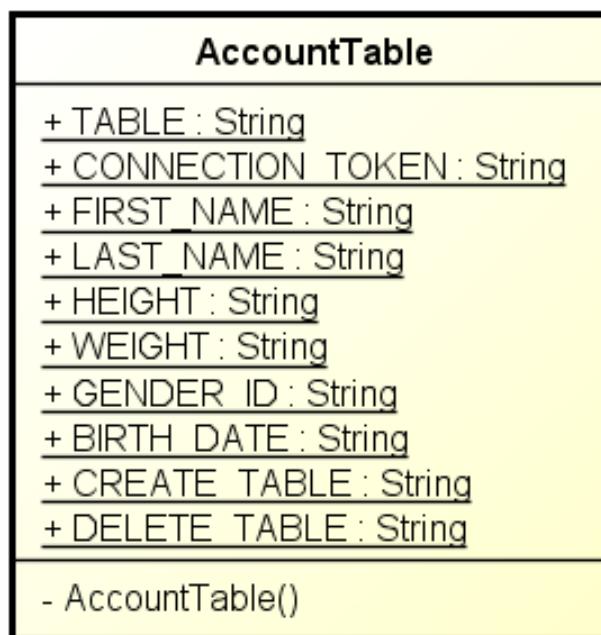


Figura 61: Diagramma di AccountTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *account* del *database*.
- **Implementa:**
  - android.provider.BaseColumns
- **Attributi:**
  - **+\_ID : String**  
Nome della colonna che costituisce una chiave primaria intera per la tabella (Valore: *\_id*).
  - **+\_COUNT : String**  
Nome della colonna che contiene il conto delle righe della tabella stessa.
- **Metodi:**
  - **-AccountTable( ) :**  
Costruttore privato della classe.

## 3.18.2.2 shike::app::model::dao::db::GendersTable

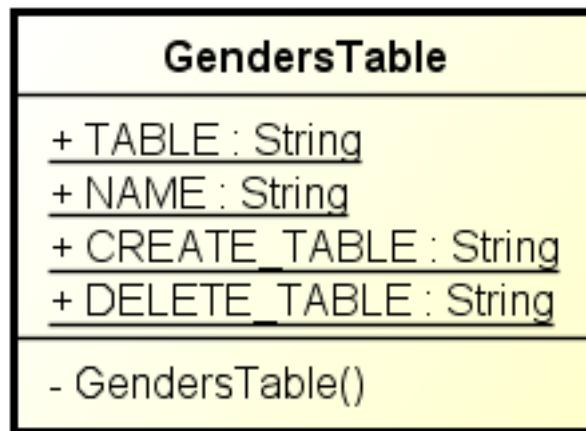


Figura 62: Diagramma di GendersTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *genders* del *database*.
- **Implementa:**
  - android.provider.BaseColumns
- **Attributi:**
  - `+_ID : String`  
Nome della colonna che costituisce una chiave primaria intera per la tabella (Valore: `_id`).
  - `+_COUNT : String`  
Nome della colonna che contiene il conto delle righe della tabella stessa.
- **Metodi:**
  - `–GendersTable( ) :`  
Costruttore privato della classe.

### 3.18.2.3 shike::app::model::dao::db::DbContract

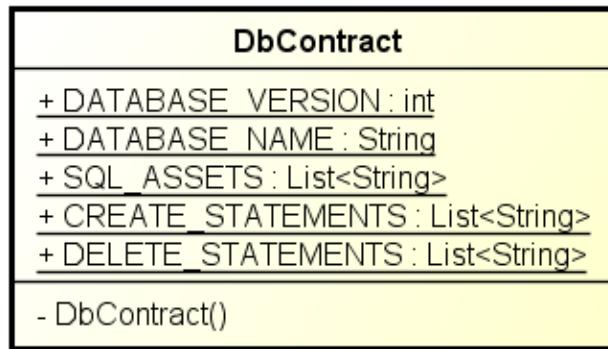


Figura 63: Diagramma di DbContract

- **Tipo:** concreta
- **Descrizione:** classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione. Ogni sua classe interna modella una tabella del database. Ognuna di esse contiene le seguenti stringhe costanti:
  - **TABLE**, contenente il nome della tabella;
  - **CREATE\_QUERY**, contenente la query di creazione della tabella;
  - **DELETE\_QUERY**, contenente la *query* di cancellazione della tabella;
  - più costanti, ognuna contenente il nome di una colonna della tabella.
- **Attributi:**
  - `+DATABASE_VERSION : int` Costante intera che indica la versione corrente del *database* (Valore:1).
  - `+DATABASE_NAME : String` Costante che indica il nome del *database* (Valore:shike.db).
  - `+SQL_ASSETS : List<String>` Lista dei nomi dei file (presenti tra gli *asset* dell'applicazione) contenenti le frasi SQL per il popolamento del *database* al primo avvio.
  - `+CREATE_STATEMENTS : List<String>` Lista non modificabile contenente tutti gli *statement* di creazione delle tabelle del *database*, ordinati in modo tale da non provocare errori durante la creazione delle chiavi esterne.
  - `+DELETE_STATEMENTS : List<String>` Lista non modificabile contenente tutti gli *statement* di cancellazione delle tabelle del *database*.
- **Metodi:**
  - `-DbContract( ) :`  
Costruttore privato, in modo da impedire l'istanziazione della classe.

### 3.18.2.4 shike::app::model::dao::db::DbUtil

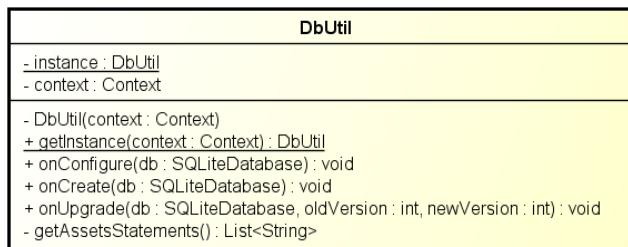


Figura 64: Diagramma di DbUtil

- **Tipo:** concreta
- **Descrizione:** classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione implementata tramite il *design pattern singleton*
- **Superclassi:**
  - android.database.sqlite.SQLiteOpenHelper
- **Attributi:**
  - `–instance : DbUtil` Unica istanza di DbUtil. Vale ancora *null*, se non è mai stato invocato il metodo `getInstance`.
  - `–context : Context` Contesto dell'applicazione, necessario per poter effettuare operazioni sul database.
- **Metodi:**
  - `–DbUtil( context : Context )`:  
Costruttore della classe, privato per impedire istanziazioni multiple della classe.  
**Argomenti:**
    - \* `context` : contesto dell'applicazione.
  - `+getInstance( context : Context ) : DbUtil`  
Ritorna l'unica istanza della classe (se non esiste ancora, viene creata).  
**Argomenti:**
    - \* `context` : contesto dell'applicazione.
  - `+onConfigure( db : SQLiteDatabase ) : void`  
Metodo eseguito automaticamente durante la configurazione della connessione con il database. Si limita ad attivare i vincoli di chiave esterna, in quanto di default non vengono applicati.  
**Argomenti:**
    - \* `db` : il database da configurare.
  - `+onCreate( db : SQLiteDatabase ) : void`  
Metodo eseguito automaticamente alla creazione di un nuovo database. Si occupa di inizializzare le tabelle e di inserire i valori di default letti dagli *asset*.  
**Argomenti:**
    - \* `db` : .

- `+onUpgrade( db : SQLiteDatabase, oldVersion : int, newVersion : int ) : void`  
Metodo eseguito automaticamente all'aggiornamento (cambio di numero di versione) di un database. Elimina tutte le tabelle del database e le ricrea invocando `onCreate`.  
**Argomenti:**
  - \* `db` : il database aggiornato.
  - \* `oldVersion` : il numero di versione vecchio.
  - \* `newVersion` : il numero di versione nuovo.
- `-getAssetsStatements() : List<String>`  
Metodo che prende i file .sql indicati in `DbContract.SQL_ASSETS` e ne estrae gli *statement SQL*, ritornandoli come lista di stringhe.

### 3.18.2.5 shike::app::model::dao::db::HelpNumbersTable

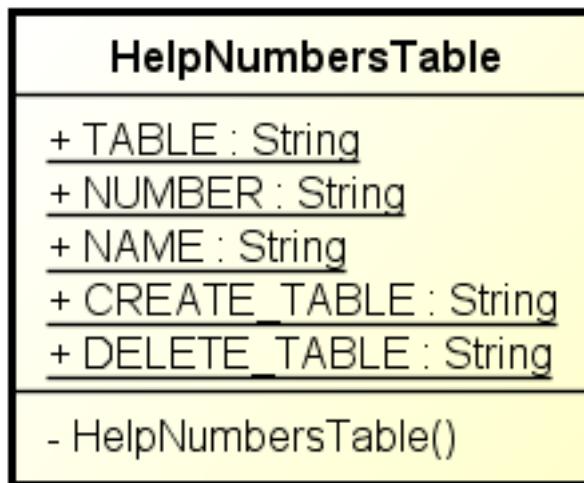


Figura 65: Diagramma di HelpNumbersTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *helpnumbers* del *database*.
- **Metodi:**
  - `-HelpNumbersTable( ) :`  
Costruttore privato della classe.

### 3.18.2.6 shike::app::model::dao::db::RecordedTracksTable

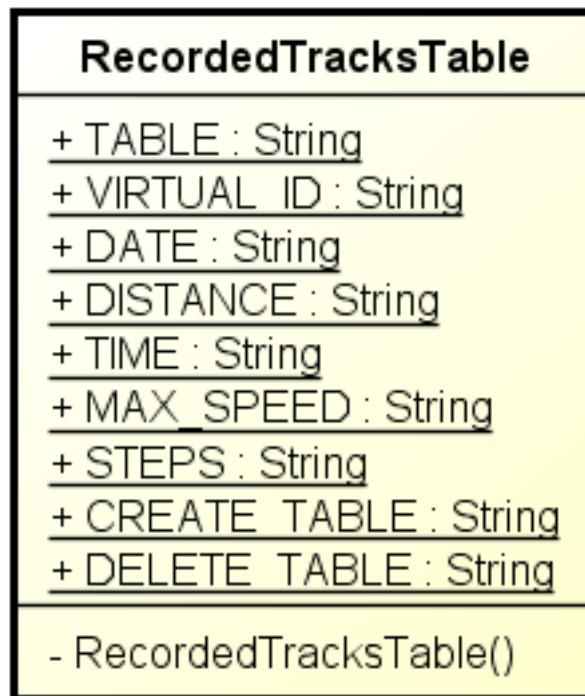


Figura 66: Diagramma di RecordedTracksTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracks* del *database*.
- **Implementa:**
  - android.provider.BaseColumns
- **Attributi:**
  - **+\_ID : String**  
Nome della colonna che costituisce una chiave primaria intera per la tabella (Valore: *\_id*).
  - **+\_COUNT : String**  
Nome della colonna che contiene il conto delle righe della tabella stessa.
- **Metodi:**
  - **-RecordedTracksTable( ) :**  
Costruttore privato della classe.

### 3.18.2.7 shike::app::model::dao::db::RecordedTrackLocationsTable



Figura 67: Diagramma di RecordedTrackLocationsTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracklocations* del *database*.
- **Metodi:**
  - `-RecordedTrackLocationsTable()` :  
Costruttore privato della classe.

### 3.18.2.8 shike::app::model::dao::db::VirtualTracksTable

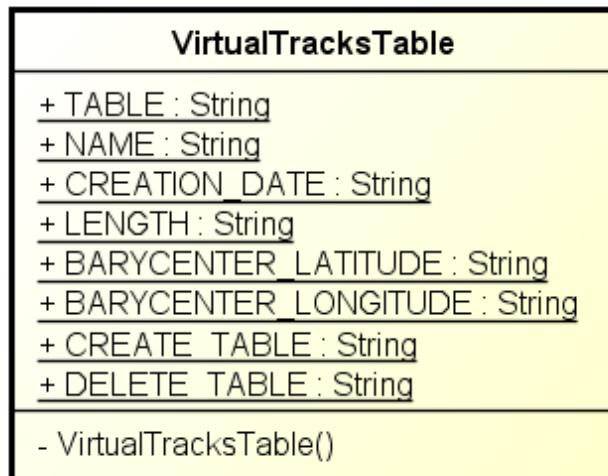


Figura 68: Diagramma di VirtualTracksTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracks* del database.
- **Implementa:**
  - android.provider.BaseColumns
- **Attributi:**
  - `+_ID : String`  
Nome della colonna che costituisce una chiave primaria intera per la tabella (Valore: `_id`).
  - `+_COUNT : String`  
Nome della colonna che contiene il conto delle righe della tabella stessa.
- **Metodi:**
  - `-VirtualTracksTable( ) :`  
Costruttore privato della classe.

### 3.18.2.9 shike::app::model::dao::db::VirtualTrackLocationsTable

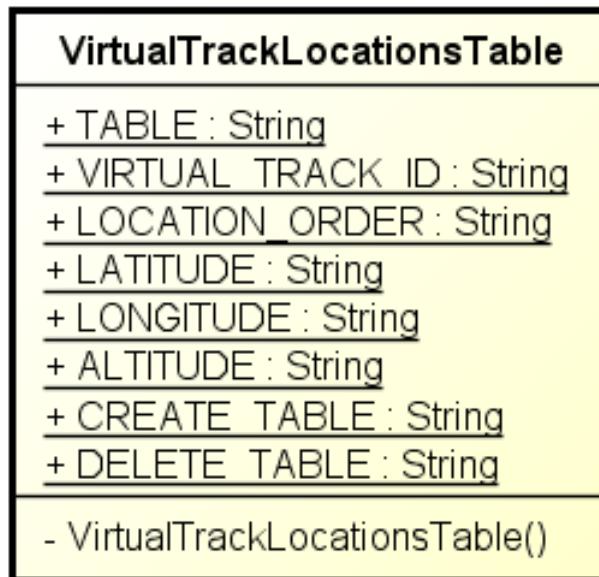


Figura 69: Diagramma di `VirtualTrackLocationsTable`

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracklocations* del database.
- **Metodi:**
  - `-VirtualTrackLocationsTable( ) :`  
Costruttore privato della classe.

## 3.18.2.10 shike::app::model::dao::db::WeatherForecastsTable

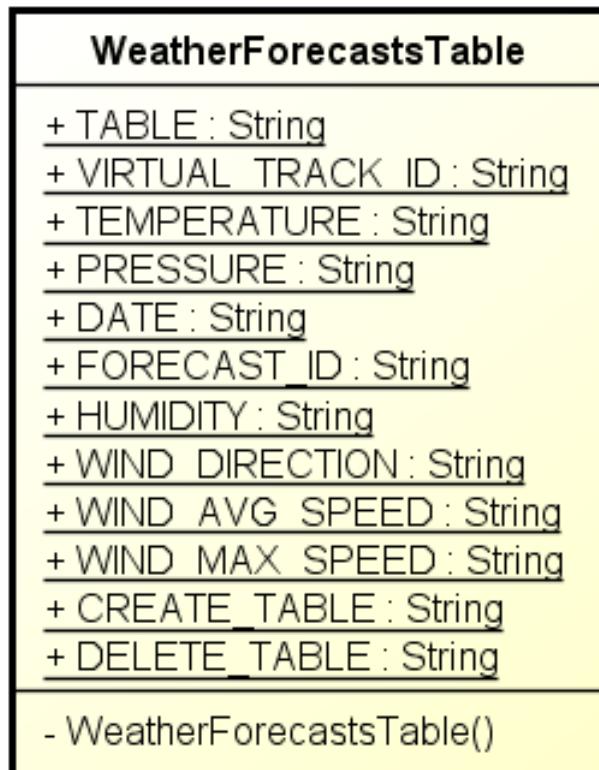


Figura 70: Diagramma di WeatherForecastsTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *weatherforecasts* del *database*.
- **Metodi:**
  - `WeatherForecastsTable()` :  
Costruttore privato della classe.

### 3.18.2.11 shike::app::model::dao::db::ForecastTypesTable

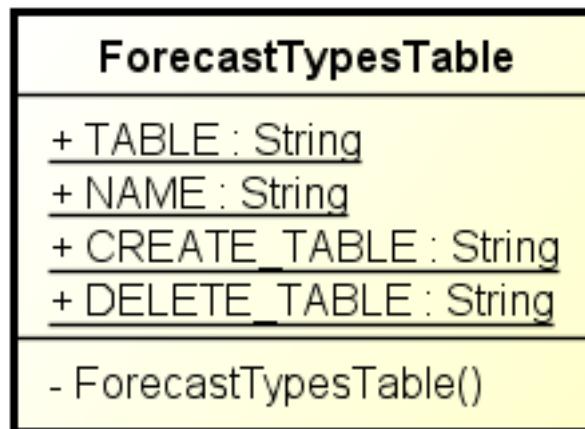


Figura 71: Diagramma di ForecastTypesTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *forecasttypes* del *database*.
- **Implementa:**
  - android.provider.BaseColumns
- **Attributi:**
  - **+\_ID : String**  
Nome della colonna che costituisce una chiave primaria intera per la tabella (Valore: *\_id*).
  - **+\_COUNT : String**  
Nome della colonna che contiene il conto delle righe della tabella stessa.
- **Metodi:**
  - **-ForecastTypesTable( ) :**  
Costruttore privato della classe.

### 3.18.2.12 shike::app::model::dao::db::PoisTable

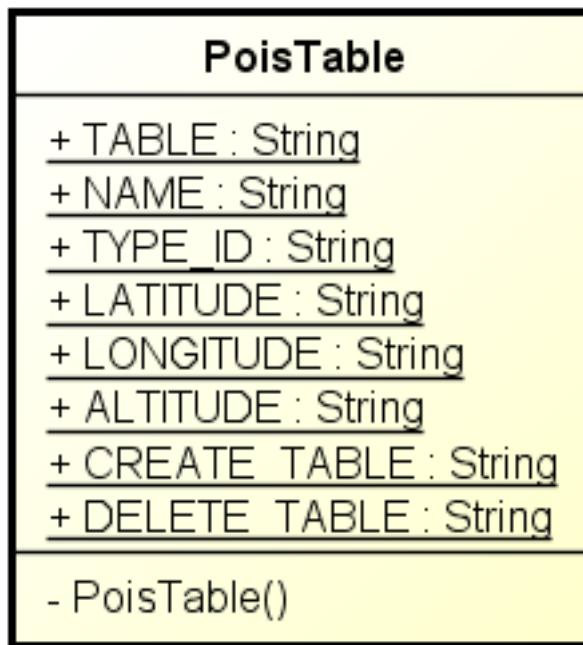


Figura 72: Diagramma di PoisTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *pois* del *database*.
- **Implementa:**
  - android.provider.BaseColumns
- **Attributi:**
  - `+_ID` : `String`  
Nome della colonna che costituisce una chiave primaria intera per la tabella (Valore: `_id`).
  - `+_COUNT` : `String`  
Nome della colonna che contiene il conto delle righe della tabella stessa.
- **Metodi:**
  - `-PoisTable()` :  
Costruttore privato della classe.

### 3.18.2.13 shike::app::model::dao::db::PoiTypesTable

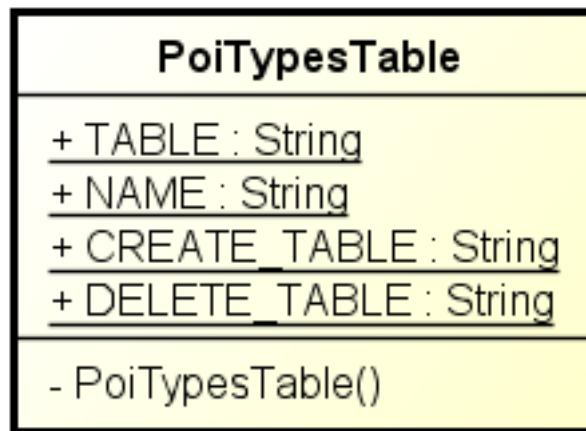


Figura 73: Diagramma di PoiTypesTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *poitypes* del *database*.
- **Implementa:**
  - android.provider.BaseColumns
- **Attributi:**
  - `+_ID : String`  
Nome della colonna che costituisce una chiave primaria intera per la tabella (Valore: `_id`).
  - `+_COUNT : String`  
Nome della colonna che contiene il conto delle righe della tabella stessa.
- **Metodi:**
  - `-PoiTypesTable( )`:  
Costruttore privato della classe.

### 3.19 shike::app::model::service

#### 3.19.1 Informazioni sul package

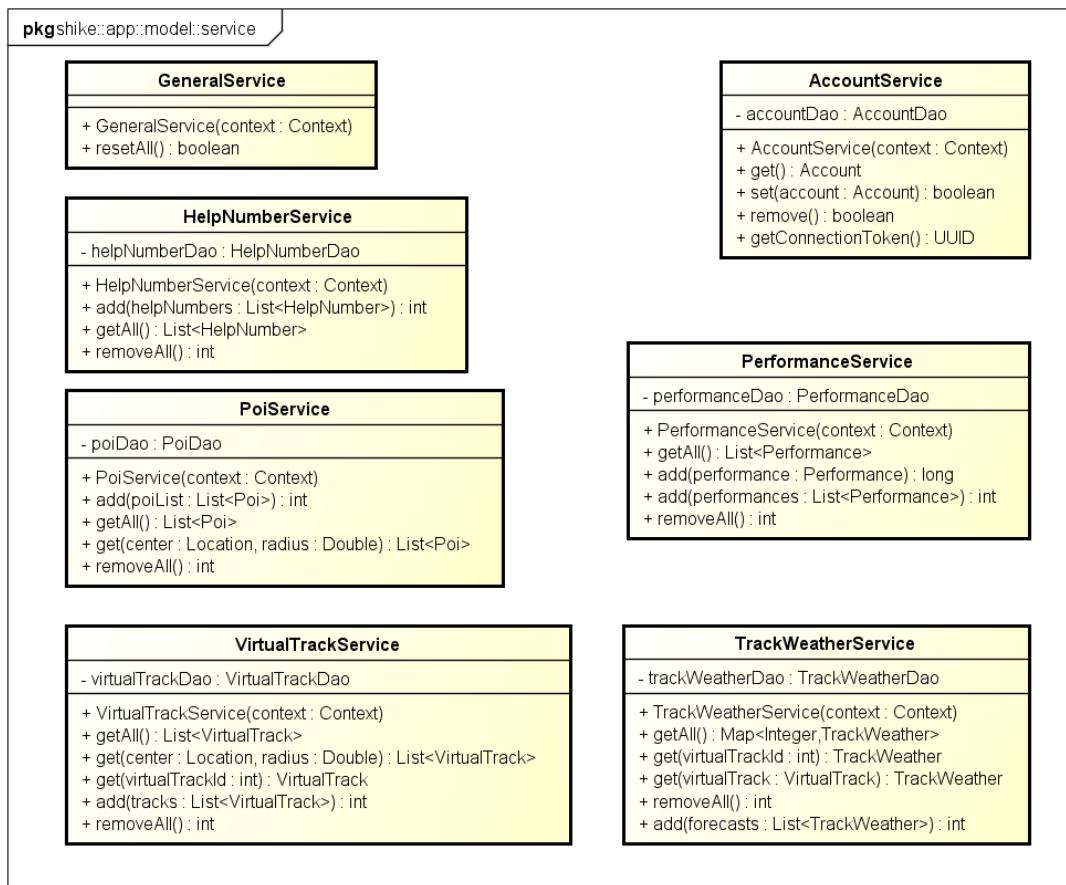


Figura 74: Diagramma di shike::app::model::service

- **Descrizione:** componente contenente tutte le classi *service* che permettono l'interfacciamento con il dao.
- **Componente padre:** shike::app::model
- **Interazioni con altri componenti**
  - shike::app::model::dao

### 3.19.2 Classi

#### 3.19.2.1 shike::app::model::service::AccountService

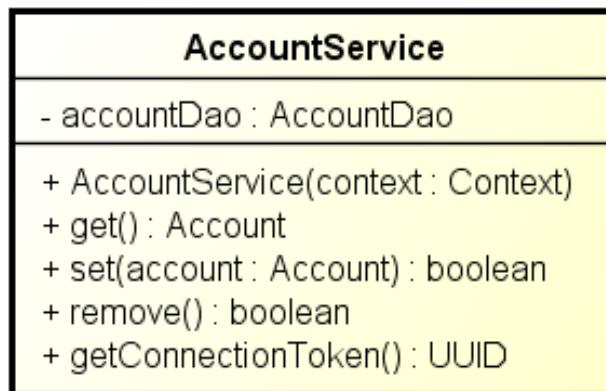


Figura 75: Diagramma di AccountService

- **Tipo:** concreta
- **Descrizione:** classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dell'*account*.
- **Attributi:**
  - **accountDao** : AccountDao  
Riferimento al DAO dell'account.
- **Metodi:**
  - **+get()** : Account  
Ritorna l'account salvato nella memoria del dispositivo.
  - **+AccountDaoService( context : Context )** :  
Costruttore della classe.

**Argomenti:**

  - \* **context** : contesto dell'applicazione, necessario per istanziare il DAO.
- **+set( account : Account ) : boolean**  
Viene salvato l'account in memoria. Se è presente già un account, viene sovrascritto (ci dev'essere sempre al più un account salvato). Ritorna true o false in base all'esito dell'operazione.

**Argomenti:**

  - \* **account** : account da salvare nella memoria.

- **+remove() : boolean**  
Rimuove l'account dalla memoria. Ritorna true se è stato rimosso con successo, false se non era presente nessun account.
- **+getConnectionToken() : UUID**  
Ritorna il token di connessione associato all'account, se presente, altrimenti ritorna null.

### 3.19.2.2 shike::app::model::service::GeneralService

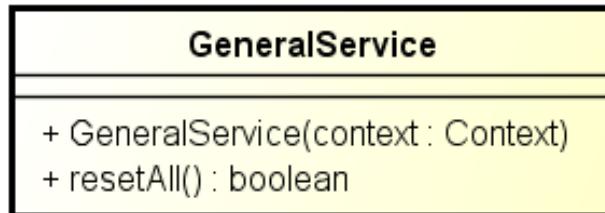


Figura 76: Diagramma di GeneralService

- **Tipo:** concreta
- **Descrizione:** servizio di gestione generale della memoria dell'applicazione.
- **Attributi:**
  - `-generalDao : GeneralDao`  
DAO di riferimento del service.
- **Metodi:**
  - `+GeneralService( context : Context )`:  
Costruttore della classe.  
**Argomenti:**  
\* `context` : contesto dell'applicazione.
  - `+resetAll() : int`  
Cancella tutti i dati creati dall'utente o da una sincronizzazione. Ritorna true se ha cancellato dati, false se non ha cancellato niente (la memoria potrebbe essere già vuota).

### 3.19.2.3 shike::app::model::service::HelpNumberService

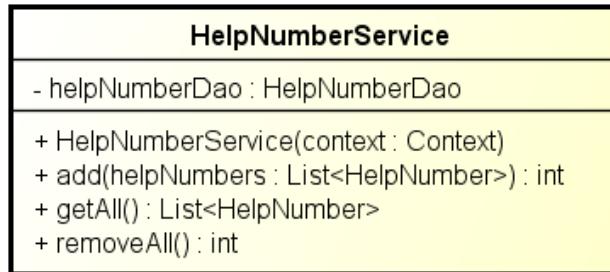


Figura 77: Diagramma di HelpNumberService

- **Tipo:** concreta
- **Descrizione:** classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dei numeri di soccorso.
- **Attributi:**

- `–helpNumberDao : HelpNumberDao`  
DAO di gestione dei numeri di soccorso.

- **Metodi:**

- `+HelpNumberService( context : Context ) :`  
Costruttore della classe.

**Argomenti:**

\* `context` : contesto dell'applicazione.

- `+add( helpNumbers : List<HelpNumber> ) : int`  
Aggiunge una lista di numeri in memoria, cancellando prima tutti quelli già presenti.  
Ritorna il numero di numeri aggiunti.

**Argomenti:**

\* `helpNumbers` : lista di numeri da aggiungere.

- `+getAll() : List<HelpNumber>`  
Ritorna la lista di tutti i numeri salvati in memoria.

- `+removeAll() : int`  
Rimuove tutti i numeri salvati in memoria. Ritorna il numero di numeri rimossi.

### 3.19.2.4 shike::app::model::service::PerformanceService

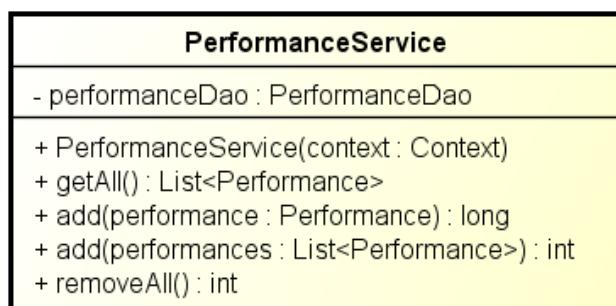


Figura 78: Diagramma di PerformanceService

- **Tipo:** concreta

- **Descrizione:** classe *service* che fa da ponte tra i presenter e il DAO delle performance dell'utente.

- **Attributi:**

- `–recordedTrackDao : RecordedTrackDao`  
DAO delle performance dell'utente (e dei relativi percorsi creati).

- **Metodi:**

- `+PerformanceService( context : Context ) :`  
Costruttore della classe.

**Argomenti:**

\* `context` : contesto dell'applicazione.

- `+getAll( ) : List<Performance>`  
Ritorna tutte le performance salvate nella memoria dell'applicazione.

- `+add( performance : Performance ) : long`  
Salva una performance in memoria e ne ritorna l'id.  
**Argomenti:**  
\* `performance` : performance da salvare nella memoria.
- `+add( performances : List<Performance> ) : int`  
Aggiunge una lista di performance in memoria, prima cancellando tutte quelle già presenti. Ritorna il numero di performance aggiunte correttamente.  
**Argomenti:**  
\* `performances` : lista delle performance da aggiungere.
- `+removeAll() : int`  
Rimuove tutte le performance dalla memoria dell'applicazione e ritorna il numero di performance eliminate.

### 3.19.2.5 shike::app::model::service::PoiService

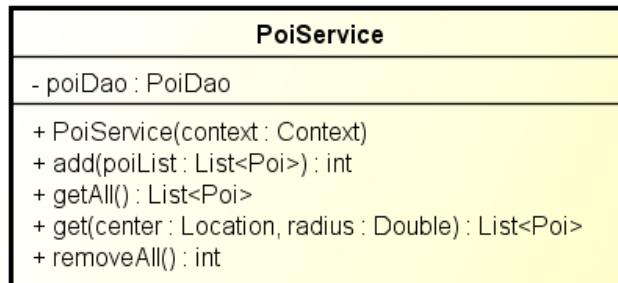


Figura 79: Diagramma di PoiService

- **Tipo:** concreta
- **Descrizione:** classe *service* che fa da ponte tra i presenter e il DAO dei punti di interesse.
- **Attributi:**
  - `-poiDao : PoiDao`  
Riferimento al DAO dei punti di interesse.
  - `-poiDao : PoiDao`  
Riferimento al DAO dei punti di interesse.
- **Metodi:**
  - `+PoiService( context : Context ) :`  
Costruttore della classe.  
**Argomenti:**  
\* `context` : contesto dell'applicazione.
  - `+add( poiList : List<Poi> ) : int`  
Salva in memoria la lista di POI passata per parametro, cancellando prima tutta la memoria dei POI. Ritorna il numero di POI salvati con successo.  
**Argomenti:**  
\* `poiList` : lista dei POI da aggiungere.

- `+getAll() : List<Poi>`  
Ritorna tutti i POI salvati in memoria.
  - `+get( center : Location, radius : Double ) : List<Poi>`  
Ritorna tutti i POI entro un certo raggio radius di distanza dalla posizione centrale center, ordinati per distanza crescente.
- Argomenti:**
- \* `center` : posizione centrale.
  - \* `radius` : raggio di distanza massimo da center in cui cercare. Se è 0 o `null`, vengono ritornati tutti i POI in ordine di distanza da center.
- `+removeAll() : int`  
Rimuove tutti i POI salvati in memoria. Ritorna il numero di POI rimossi.

### 3.19.2.6 shike::app::model::service::TrackWeatherService

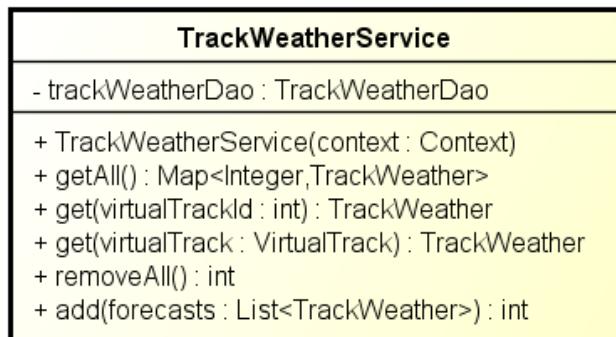


Figura 80: Diagramma di TrackWeatherService

- **Tipo:** concreta
- **Descrizione:** classe *service* che fa da ponte tra i *presenter* e il DAO delle previsioni meteo dei percorsi.
- **Attributi:**
  - `-trackWeatherDao : TrackWeatherDao`  
Riferimento al DAO per le previsioni meteo.
  - `-trackWeatherDao : TrackWeatherDao`  
Riferimento al DAO per le previsioni meteo.
- **Metodi:**
  - `+getAll() : Map<Integer, TrackWeather>`  
Ritorna una mappa contenente tutte le previsioni meteo associate ai percorsi scaricati dalla piattaforma web. La chiave è l'id del percorso associato alle previsioni.
  - `+get( virtualTrackId : int ) : TrackWeather`  
Ritorna le previsioni meteo del percorso desiderato.

**Argomenti:**

  - \* `virtualTrackId` : id del percorso di cui si desiderano le previsioni.
  - `+get( virtualTrack : VirtualTrack ) : TrackWeather`  
Ritorna le previsioni meteo per il percorso indicato.

**Argomenti:**

- \* `virtualTrack` : percorso di cui si desiderano le previsioni.
- `+removeAll() : int`  
Elimina tutte le previsioni meteo salvate nel dispositivo. Ritorna il numero di previsioni rimosse.
- `+add( forecasts : List<TrackWeather> ) : int`  
Aggiunge previsioni meteo per più percorsi nella memoria del dispositivo, cancellando prima tutte le previsioni già presenti in essa.
- Argomenti:**
  - \* `forecasts` : lista delle previsioni da aggiungere.

### 3.19.2.7 shike::app::model::service::VirtualTrackService

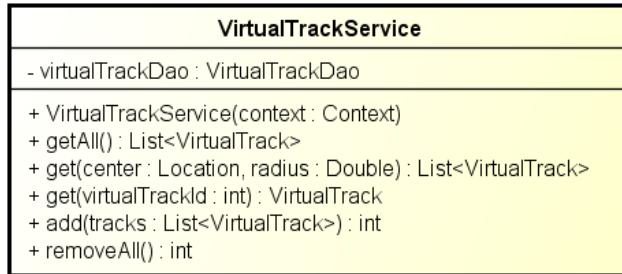


Figura 81: Diagramma di VirtualTrackService

- **Tipo:** concreta
- **Descrizione:** classe *service* che fa da ponte tra i *presenter* e il DAO dei percorsi scaricati dalla parte web.
- **Attributi:**
  - `-virtualTrackDao : VirtualTrackDao`  
DAO dei percorsi scaricati dalla piattaforma web.
- **Metodi:**
  - `+VirtualTrackService( context : Context ) :`  
Costruttore della classe.
  - Argomenti:**
    - \* `context` : contesto dell'applicazione.
  - `+getAll() : List<VirtualTrack>`  
Ritorna tutti i percorsi salvati in memoria.
  - `+get( center : Location, radius : Double ) : List<VirtualTrack>`  
Ritorna tutti i percorsi distanti al più `radius` dalla posizione `center`, ordinati per distanza crescente.
  - Argomenti:**
    - \* `center` : posizione in cui ricercare i percorsi.
    - \* `radius` : raggio di distanza massima da `center` entro cui cercare. Se è 0 o null, vengono ritornati tutti i percorsi ordinati per distanza crescente da `center`.

- **+get( virtualTrackId : int ) : VirtualTrack**  
Ritorna il percorso con l'id cercato (null se tale percorso non esiste).
 **Argomenti:**
  - \* **virtualTrackId** : id del percorso desiderato.
- **+add( tracks : List<VirtualTrack> ) : int**  
Aggiunge più percorsi in memoria, cancellando quelli già presenti. Ritorna il numero di percorsi aggiunti con successo.
 **Argomenti:**
  - \* **tracks** : lista di percorsi da aggiungere.
- **+removeAll() : int**  
Elimina tutti i percorsi dalla memoria del dispositivo.

### 3.20 shike::app::model::sync

#### 3.20.1 Informazioni sul package

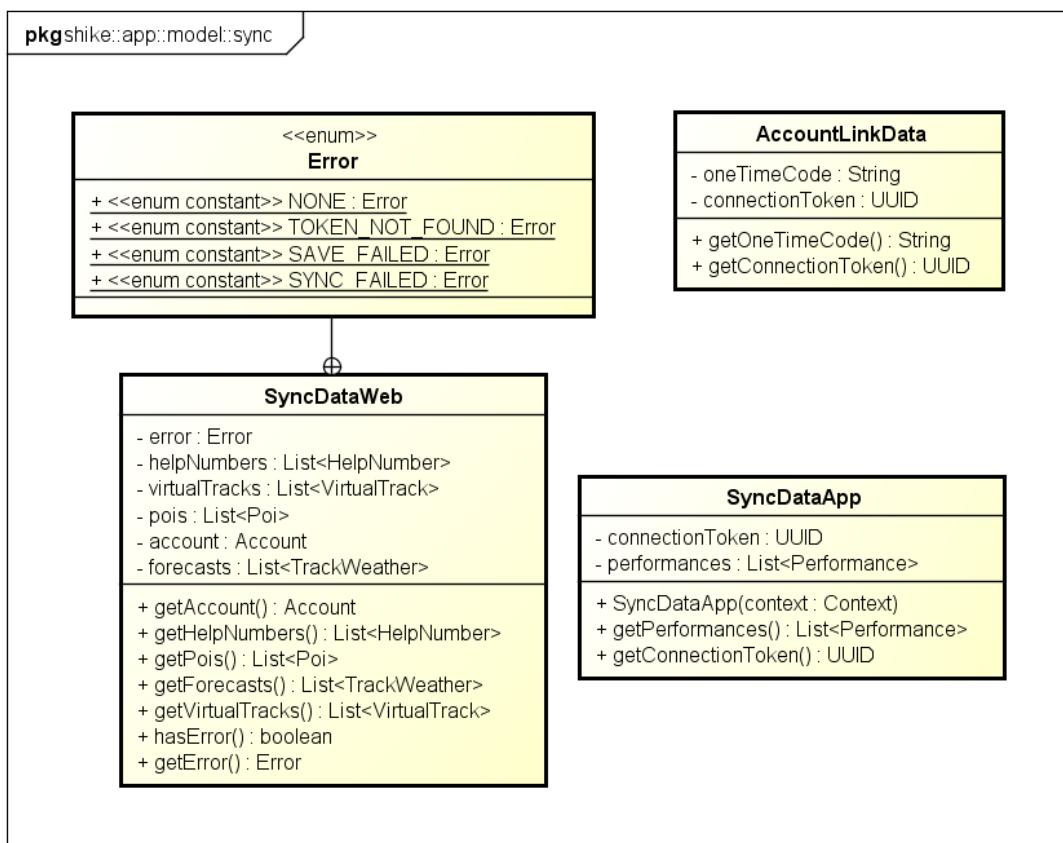


Figura 82: Diagramma di shike::app::model::sync

- **Descrizione:** classe che contiene le classi che modellano i dati che vengono scambiati durante la sincronizzazione e il collegamento account con la piattaforma web.
- **Componente padre:** shike::app::model
- **Interazioni con altri componenti**

- shike::app::helper

### 3.20.2 Classi

#### 3.20.2.1 shike::app::model::sync::SyncDataApp

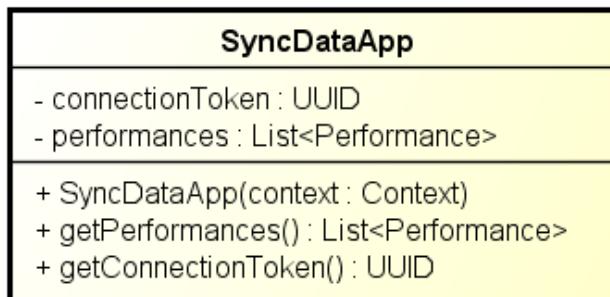


Figura 83: Diagramma di SyncDataApp

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati che l'applicazione invia alla piattaforma web durante una sincronizzazione.
- **Attributi:**
  - **connectionToken** : **UUID**  
*Token* di connessione all'account, necessario per la piattaforma web nell'identificazione dell'account che sta eseguendo la sincronizzazione.
  - **performances** : **List<Performance>**  
Lista di tutte le performance salvate nel dispositivo al momento della sincronizzazione.
- **Metodi:**
  - **+getConnectionToken()** : **UUID**  
Restituisce la variabile **connectionToken**.
  - **+getPerformances()** : **List<Performance>**  
Restituisce la variabile **performances**.
  - **+SyncDataApp( context : Context )** :

**Argomenti:**

  - \* **context** : contesto dell'applicazione, necessario per accedere al database e prelevare i dati.

### 3.20.2.2 shike::app::model::sync::AccountLinkData

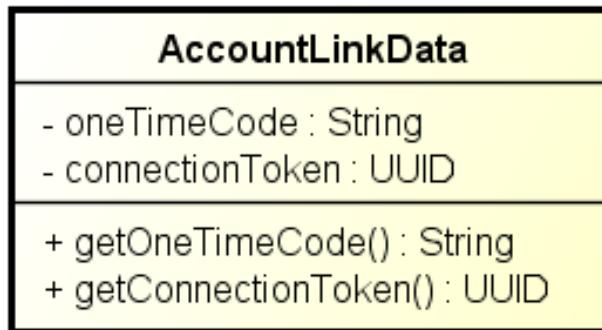


Figura 84: Diagramma di AccountLinkData

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati che vengono ricevuti dalla piattaforma web durante il primo collegamento dell' *account*.
- **Attributi:**
  - `connectionToken` : `UUID`  
`Token` di connessione temporaneo alla piattaforma web. Ha una durata di validità limitata, ma se viene completato il collegamento viene associato all'account e non viene più cancellato (a meno di un eventuali scollegamento manuale dell'account).
  - `oneTimeCode` : `String`  
 Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento.
- **Metodi:**
  - `+getConnectionToken()` : `UUID`  
 Restituisce la variabile `connectionToken`.
  - `+getOneTimeCode()` : `String`  
 Restituisce la variabile `oneTimeCode`.
  - `+AccountLinkData( context : Context )` :  
 Costruttore della classe, si occupa di prelevare autonomamente i dati dal *database* per inizializzare i campi.
- **Argomenti:**
  - \* `context` : contesto dell'applicazione.

### 3.20.2.3 shike::app::model::sync::SyncDataWeb

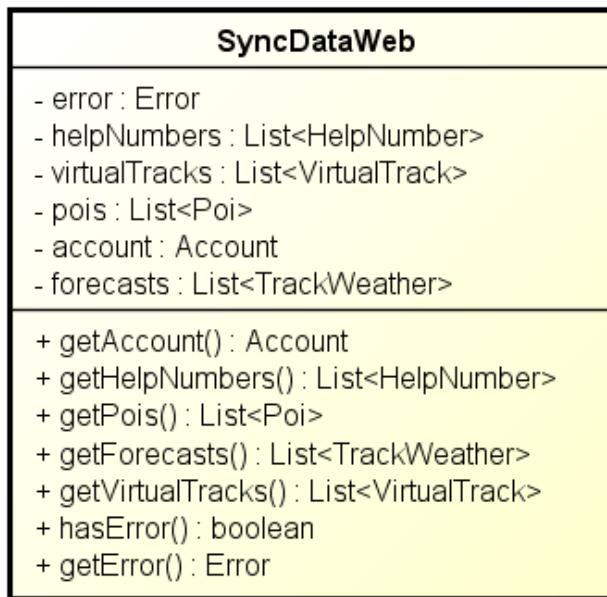


Figura 85: Diagramma di SyncDataWeb

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
- **Attributi:**
  - **error** : SyncDataWeb.Error  
Valore che indica se si sono verificati errori nella piattaforma web durante la sincronizzazione. Vale NONE se invece non si sono verificati errori.
  - **pois** : List<Poi>  
Lista dei POI aggiornati.
  - **account** : Account  
Informazioni aggiornate sull'account associato al dispositivo.
  - **forecasts** : List<TrackWeather>  
Lista con tutte le previsioni meteo per i percorsi selezionati dall'utente nella piattaforma web.
  - **virtualTracks** : List<VirtualTrack>  
Lista dei percorsi selezionati dall'utente nella parte web.
  - **helpNumbers** : List<HelpNumber>  
Lista dei numeri di soccorso dell'account.
- **Metodi:**
  - **+getError()** : SyncDataWeb.Error  
Restituisce la variabile error.
  - **+getPois()** : List<Poi>  
Restituisce la variabile pois.

- `+getAccount() : Account`  
Restituisce la variabile `account`.
- `+getForecasts() : List<TrackWeather>`  
Restituisce la variabile `forecasts`.
- `+getVirtualTracks() : List<VirtualTrack>`  
Restituisce la variabile `virtualTracks`.
- `+getHelpNumbers() : List<HelpNumber>`  
Restituisce la variabile `helpNumbers`.
- `+hasError() : boolean`  
Ritorna true se l'attributo `error` è diverso da `NONE`.

### 3.20.2.4 shike::app::model::sync::SyncDataWeb::Error

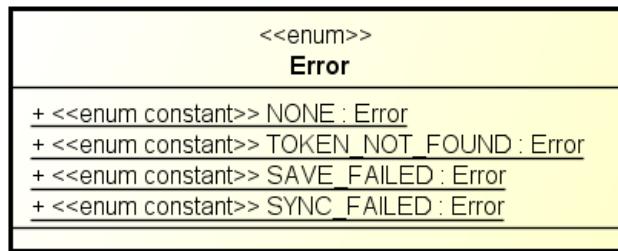


Figura 86: Diagramma di SyncDataWeb::Error

- **Tipo:** enum
- **Descrizione:** enum contenente i possibili errori che la piattaforma web può dare di risposta all'applicazione.

### 3.21 shike::app::view

#### 3.21.1 Informazioni sul package

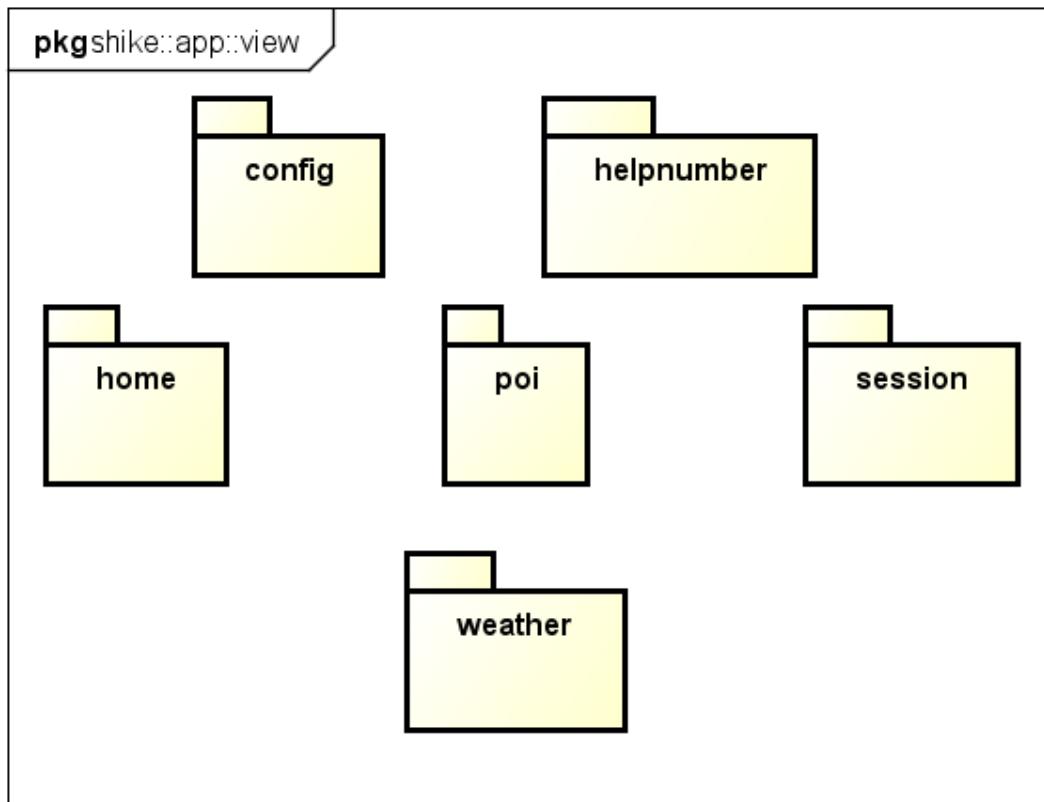


Figura 87: Diagramma di shike::app::view

- **Descrizione:** componente *View* dell'architettura MVP. Essa gestisce la visualizzazione dei dati dell'utente all'interno dell'applicazione.
- **Componenti contenute**
  - shike::app::view::config
  - shike::app::view::helpnumber
  - shike::app::view::home
  - shike::app::view::poi
  - shike::app::view::session
  - shike::app::view::weather
- **Componente padre:** shike::app
- **Interazioni con altri componenti**
  - shike::app::presenter
  - shike::app::model

### 3.22 shike::app::view::config

#### 3.22.1 Informazioni sul package

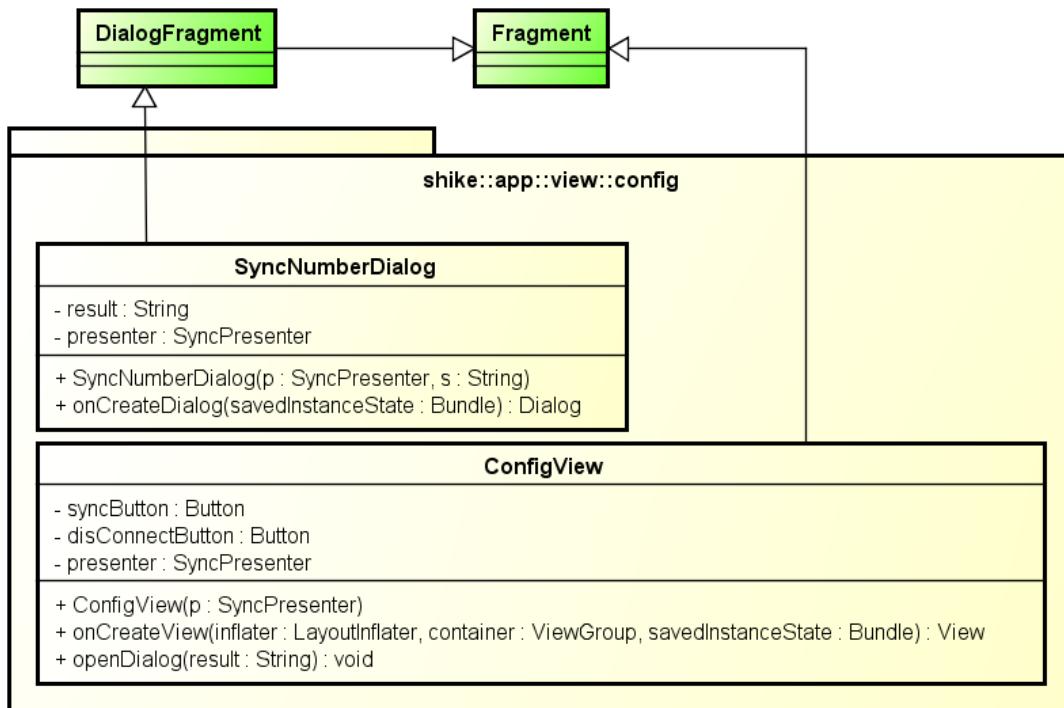


Figura 88: Diagramma di shike::app::view::config

- **Descrizione:** componente contenente le *view* del menù di configurazione dell'applicazione.
- **Componente padre:** shike::app::view
- **Interazioni con altri componenti**
  - shike::app::presenter

#### 3.22.2 Classi

##### 3.22.2.1 shike::app::view::config::ConfigView

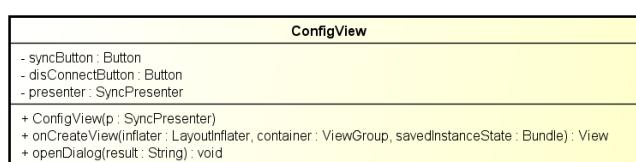


Figura 89: Diagramma di ConfigView

- **Tipo:** concreta

- **Descrizione:** classe che rappresenta la *view* che contiene le impostazioni modificabili dall'utente e la possibilità di scollegare il profilo dalla piattaforma web.

- **Superclassi:**

- android.support.v4.app.Fragment

- **Attributi:**

- `disconnectsButton : Button`  
Bottone che disconnette il profilo dalla piattaforma web.
- `syncButton : Button`  
Bottone che sincronizza i dati con la piattaforma web.
- `syncButton : Button`  
Bottone che sincronizza i dati con la piattaforma web.
- `presenter : SyncPresenter`  
Riferimento al presenter.

- **Metodi:**

- `+onCreateView( inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle ) : View`  
Crea la vista assegnando le varie azioni ai pulsanti incaricati.

**Argomenti:**

- \* `inflater` : l'oggetto inflater viene utilizzato per inserire nuove view nel Fragment.

- \* `container` : se non nullo, indica la view padre a cui il Fragment viene assegnato.

- \* `savedInstanceState` : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.

- `+ConfigView( ) :`

Costruttore della classe.

- `+openDialog( result : String ) : void`

Apre un *dialog* per visualizzare il numero per collegare il dispositivo alla piattaforma web.

**Argomenti:**

- \* `result` :

### 3.22.2.2 shike::app::view::config::SyncNumberDialog

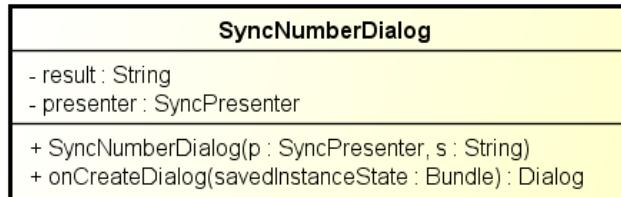


Figura 90: Diagramma di SyncNumberDialog

- **Tipo:** concreta

- **Descrizione:** *Dialog* che visualizza il codice a 5 cifre da inserire nel sito.

- **Attributi:**

- `-presenter : SyncPresenter`  
Riferimento al *presenter* della sincronizzazione.
- `-result : String`  
Stringa con il codice numerico.

- **Metodi:**

- `+onCreateDialog() : Dialog`  
Crea il *dialog* e imposta le funzioni dei pulsanti.
- `+SyncNumberDialog() :`  
Costruttore della classe.

### 3.23 shike::app::view::helpnumber

#### 3.23.1 Informazioni sul package

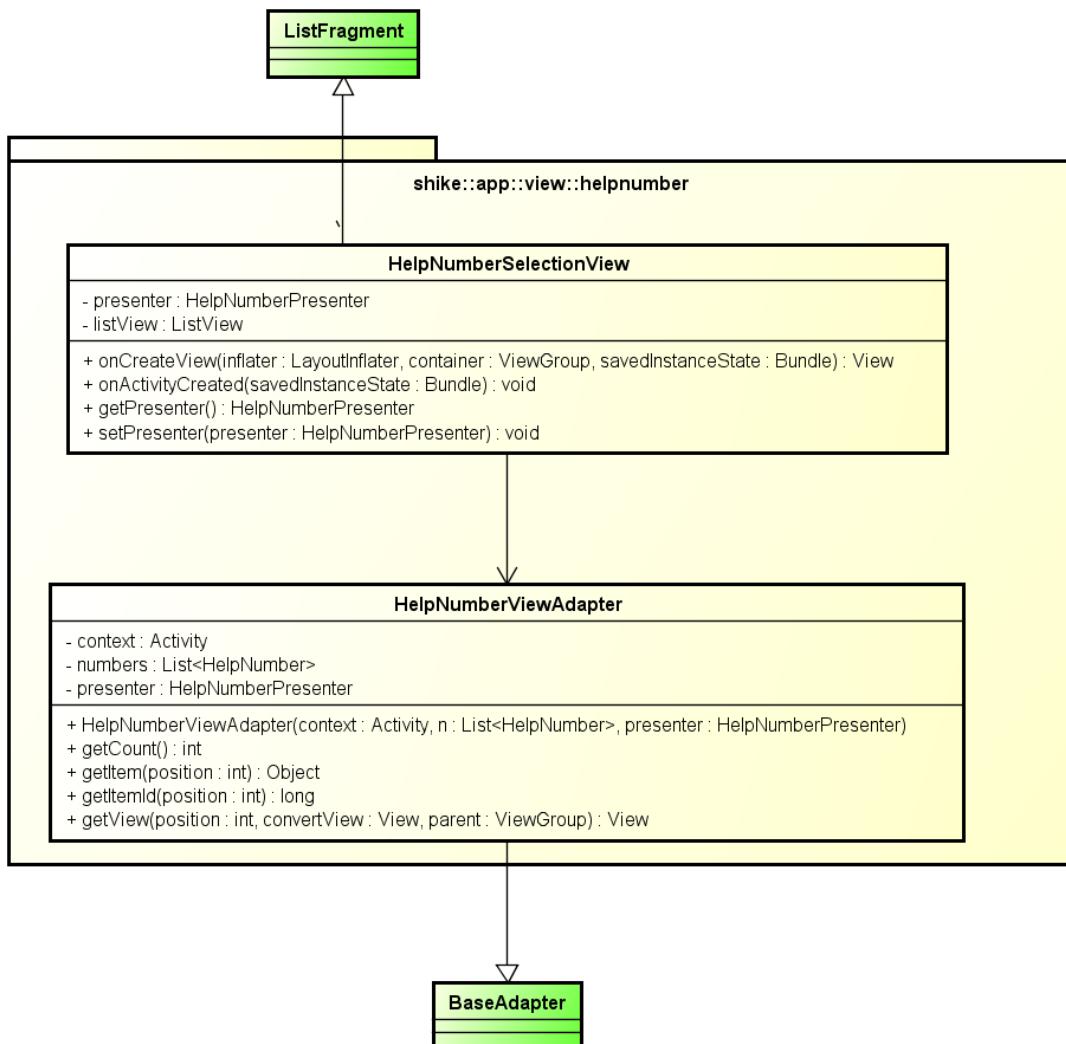


Figura 91: Diagramma di `shike::app::view::helpnumber`

- **Descrizione:** componente contenente le *view* riguardanti la visualizzazione dei numeri di soccorso.
- **Componente padre:** shike::app::view
- **Interazioni con altri componenti**
  - shike::app::presenter

### 3.23.2 Classi

#### 3.23.2.1 shike::app::view::helppnumber::HelpNumberSelectionView

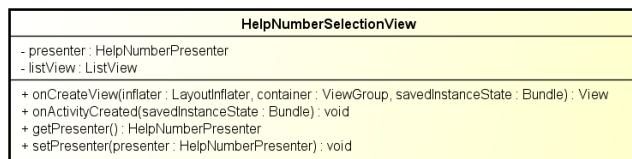


Figura 92: Diagramma di HelpNumberSelectionView

- **Tipo:** concreta
- **Descrizione:** classe che modella la visualizzazione dei numeri di soccorso.
- **Superclassi:**
  - android.support.v4.app.ListFragment
- **Attributi:**
  - **presenter** : HelpNumberPresenter  
Riferimento al presenter.
  - **listView** : ListView  
Lista degli elementi di cui è composta la view.
- **Metodi:**
  - **+onActivityCreated( savedInstanceState : Bundle ) : void**  
Viene utilizzato quando la *view* è già stata creata e serve ad esempio per impostare uno stato precedentemente utilizzato.  
**Argomenti:**
    - \* **savedInstanceState** : se il Fragment viene ricostruito da uno stato precedente, savedInstanceState indica lo stato precedente.
  - **+setMenuVisibility( menuVisible : boolean ) : void**  
Utilizzato per comunicare informazioni quando il fragment è visualizzato a schermo.  
**Argomenti:**
    - \* **menuVisible** : visibilità del fragment.
  - **+onItemClick( arg0 : AdapterView<?>, arg1 : View, position : int, id : long ) : void**  
**Argomenti:**
    - \* **arg0** : riferimento all'adapter.

- \* `arg1` : riferimento alla view.
- \* `position` : posizione della view nell'adapter.
- \* `id` : id della posizione selezionata.
- `+setPresenter( presenter : HelpNumberPresenter ) : void`  
Imposta il riferimento al presenter.
- Argomenti:**
  - \* `presenter` : riferimento al presenter.
- `+getPresenter( ) : HelpNumberPresenter`  
Ritorna il riferimento al presenter.

### 3.23.2.2 shike::app::view::helpnumber::HelpNumberViewAdapter

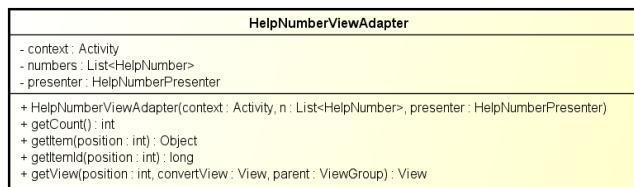


Figura 93: Diagramma di HelpNumberViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei numeri di soccorso.
- **Superclassi:**
  - android.widget.BaseAdapter
- **Attributi:**
  - `_context : Activity`  
Activity di riferimento.
  - `_numbers : List<HelpNumber>`  
Lista dei numeri da visualizzare.
  - `_presenter : HelpNumberPresenter`  
Riferimento al presenter.
- **Metodi:**
  - `+getView( position : int, convertView : View, parent : ViewGroup ) : View`  
Restituisce la visualizzazione dei dati.
  - Argomenti:**
    - \* `position` : la posizione all'interno della lista che vogliamo utilizzare.
    - \* `convertView` : se presenta indica la vecchia View da riutilizzare.
    - \* `parent` : se presente indica la View padre a cui associarsi.
  - `+getCount() : int`  
Indica quanti item sono rappresentati dall' adapter.
  - `+getItem( position : int ) : Object`  
Ritorna l'oggetto presente nella posizione indicata.
  - Argomenti:**

\* `position` : indica la posizione dell'oggetto da ritornare.

- `+getItemId( positon : int ) : long`  
Ritorna l'indice dell'oggetto presente nella posizione indicata.

**Argomenti:**

- \* `positon` : indica la posizione dell'oggetto di cui vogliamo far ritornare l'id.

- `+HelpNumberViewAdapter( n : List<HelpNumber>, context : Activity ) :`  
Costruttore di HelpNumberViewAdapter.

**Argomenti:**

- \* `n` : lista dei numeri di soccorso da visualizzare.
- \* `context` : riferimento all'activity.

### 3.24 shike::app::view::home

#### 3.24.1 Informazioni sul package

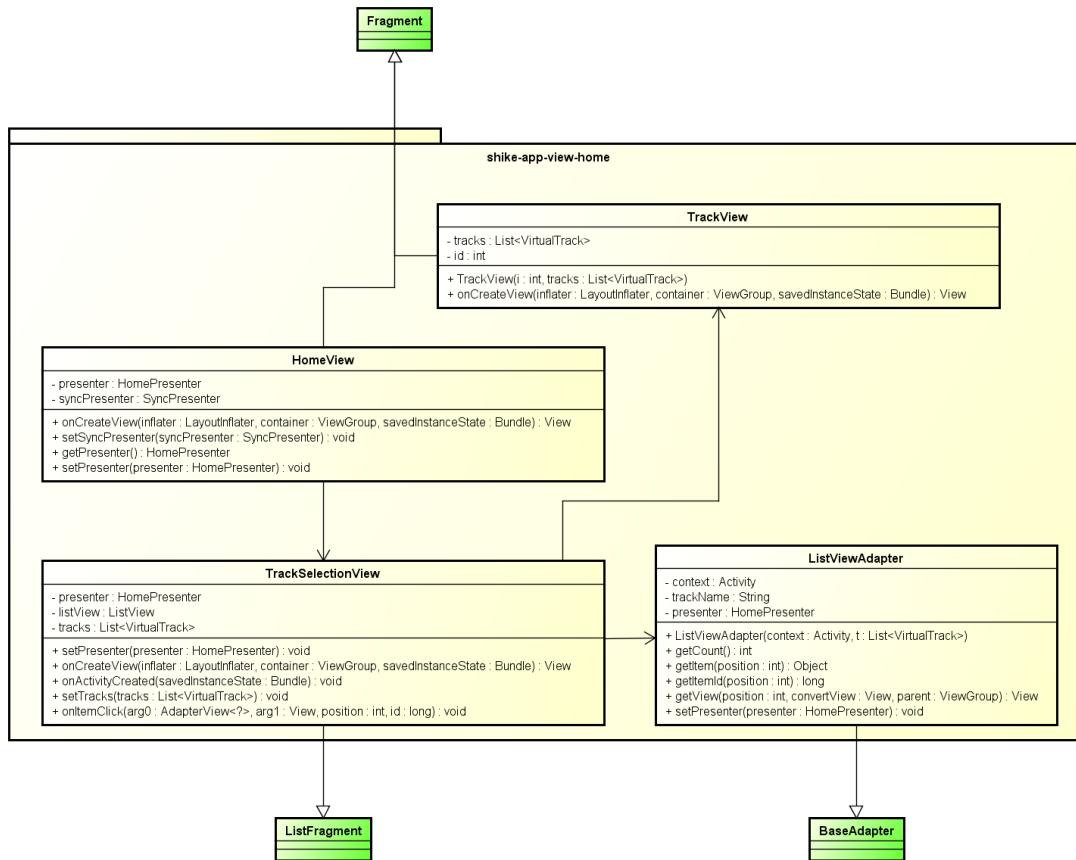


Figura 94: Diagramma di `shike::app::view::home`

- **Descrizione:** componente contenente le *view* del menu principale
- **Componente padre:** `shike::app::view`
- **Interazioni con altri componenti**
  - `shike::app::presenter`

### 3.24.2 Classi

#### 3.24.2.1 shike::app::view::home::TrackView

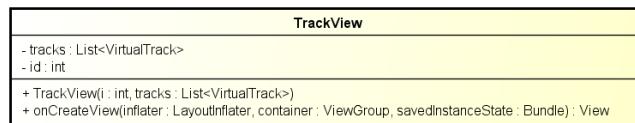


Figura 95: Diagramma di TrackView

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per la visualizzazione dei dati di un singolo percorso
- **Superclassi:**
  - android.support.v4.app.Fragment
- **Attributi:**
  - **–int : id**  
Id del percorso selezionato.
  - **–tracks : List<VirtualTrack>**  
Lista dei percorsi salvati nel dispositivo.
- **Metodi:**
  - **+onCreateView( inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle ) : View**  
Crea la vista assegnando le varie azioni ai pulsanti incaricati.  
**Argomenti:**
    - \* **inflater** : l'oggetto inflater viene utilizzato per inserire nuove view nel Fragment.
    - \* **container** : se non nullo, indica la view padre a cui il Fragment viene assegnato.
    - \* **savedInstanceState** : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.
  - **+TrackView( i : int, tracks : List<VirtualTrack> ) :**  
Costruttore di *TrackView*.  
**Argomenti:**
    - \* **i** : id del tracciato visualizzato.
    - \* **tracks** : lista dei tracciati.

#### 3.24.2.2 shike::app::view::home::HomeView

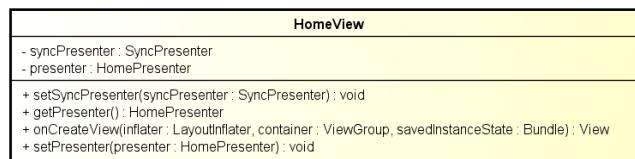


Figura 96: Diagramma di HomeView

- **Tipo:** concreta

- **Descrizione:** classe che rappresenta la visualizzazione della pagina principale dell'applicazione
- **Superclassi:**
  - android.support.v4.app.Fragment
- **Attributi:**
  - `syncPresenter` : SyncPresenter  
Riferimento al presenter utilizzato per gestire la sincronizzazione.
  - `homePresenter` : HomePresenter  
Riferimento al *presenter* del *fragment*.
- **Metodi:**
  - `+setSyncPresenter( syncPresenter : SyncPresenter ) : void`  
**Argomenti:**
    - \* `syncPresenter` : valore da impostare nella variabile `syncPresenter`.
  - `+getHomePresenter() : HomePresenter`  
Restituisce la variabile `homePresenter`.
  - `+setHomePresenter( homePresenter : HomePresenter ) : void`  
**Argomenti:**
    - \* `homePresenter` : valore da impostare nella variabile `homePresenter`.
  - `+onCreateView( inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle ) : View`  
Crea la vista assegnando le varie azioni ai pulsanti incaricati.  
**Argomenti:**
    - \* `inflater` : l'oggetto inflater viene utilizzato per inserire nuove view nel Fragment.
    - \* `container` : se non nullo, indica la view padre a cui il Fragment viene assegnato.
    - \* `savedInstanceState` : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.

### 3.24.2.3 shike::app::view::home::ListViewAdapter

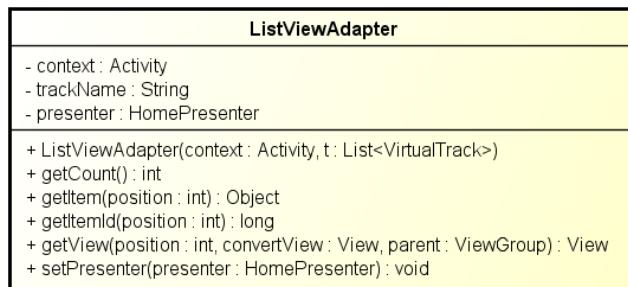


Figura 97: Diagramma di ListViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei tracciati.

- **Superclassi:**

- android.widget.BaseAdapter

- **Attributi:**

- `trackName : String`  
Nome del tracciato.
- `presenter : HomePresenter`  
Riferimento al presenter.
- `context : Activity`  
*Activity* associata alla lista.

- **Metodi:**

- `+getView( position : int, convertView : View, parent : ViewGroup ) : View`  
Restituisce la visualizzazione dei dati.

**Argomenti:**

- \* `position` : la posizione all'interno della lista che vogliamo utilizzare.
- \* `convertView` : se presenta indica la vecchia View da riutilizzare.
- \* `parent` : se presente indica la View padre a cui associarsi.

- `+getCount() : int`

Indica quanti *item* sono rappresentati dall' *adapter*.

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

Ritorna l'oggetto presente nella posizione indicata.

**Argomenti:**

- \* `position` : indica la posizione dell'oggetto da ritornare.

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

Ritorna l'indice dell'oggetto presente nella posizione indicata.

**Argomenti:**

- \* `position` : indica la posizione dell'oggetto di cui vogliamo far ritornare l'id.

- `+ListViewAdapter( context : Activity, t : List<VirtualTrack> ) : ListViewAdapter`

Costruttore di *ListViewAdapter*.

**Argomenti:**

- \* `context` : activity da passare.

- \* `t` : lista dei tracciati salvati nel dispositivo.

- `+setPresenter( presenter : HomePresenter ) : void`

Imposta il riferimento al presenter.

**Argomenti:**

- \* `presenter` : riferimento al presenter.

### 3.24.2.4 shike::app::view::home::TrackSelectionView

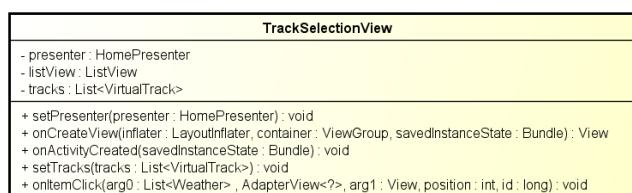


Figura 98: Diagramma di TrackSelectionView

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per la visualizzazione della lista di tutti i *VirtualTrack* presenti nel dispositivo.
- **Superclassi:**
  - android.support.v4.app.ListFragment
- **Attributi:**
  - `–presenter : HomePresenter`  
Riferimento al presenter.
  - `–listView : ListView`  
Vista che visualizza gli oggetti di una lista in modo verticale tramite *scrolling*.
  - `–tracks : List<VirtualTrack>`  
Lista dei tracciati.
- **Metodi:**
  - `+onActivityCreated( savedInstanceState : Bundle ) : void`  
Viene utilizzato quando la *view* è già stata creata e serve ad esempio per impostare uno stato precedentemente utilizzato.  
**Argomenti:**
    - \* `savedInstanceState` : se il Fragment viene ricostruito da uno stato precedente, `savedInstanceState` indica lo stato precedente.
  - `+setMenuVisibility( menuVisible : boolean ) : void`  
Utilizzato per comunicare informazioni quando il fragment è visualizzato a schermo.  
**Argomenti:**
    - \* `menuVisible` : visibilità del fragment.
  - `+onItemClick( arg0 : AdapterView<?>, arg1 : View, position : int, id : long ) : void`  
**Argomenti:**
    - \* `arg0` : riferimento all'adapter.
    - \* `arg1` : riferimento alla view.
    - \* `position` : posizione della view nell'adapter.
    - \* `id` : id della posizione selezionata.
  - `+setPresenter( presenter : HomePresenter ) : void`  
Imposta il riferimento al presenter.  
**Argomenti:**
    - \* `presenter` : riferimento al presenter.
  - `+setTracks( tracks : List<VirtualTrack> ) : void`  
Imposta la lista dei percorsi da visualizzare.  
**Argomenti:**
    - \* `tracks` : lista dei tracciati da visualizzare.

### 3.25 shike::app::view::poi

#### 3.25.1 Informazioni sul package

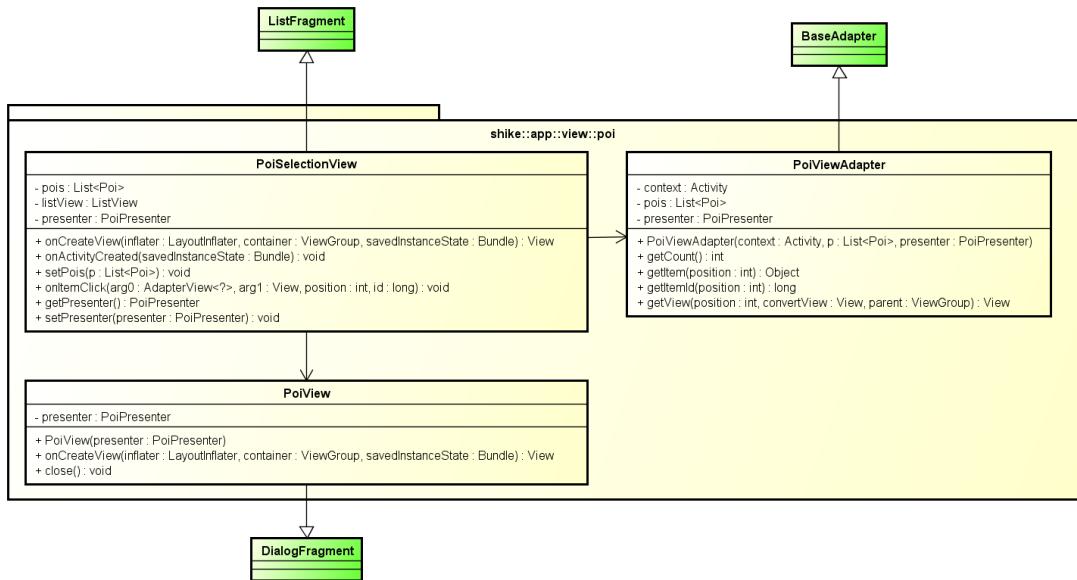


Figura 99: Diagramma di shike::app::view::poi

- **Descrizione:** componente contenente le *view* che si occupano della visualizzazione dei POI.
- **Componente padre:** shike::app::view
- **Interazioni con altri componenti**
  - shike::app::presenter

#### 3.25.2 Classi

##### 3.25.2.1 shike::app::view::poi::PoiSelectionView

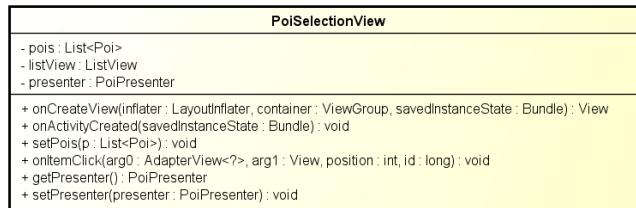


Figura 100: Diagramma di PoiSelectionView

- **Tipo:** concreta
- **Descrizione:** classe che modella la *view* per la visualizzazione e la possibile selezione dei POI da parte dell'utente.
- **Superclassi:**

- android.support.v4.app.ListFragment

- **Attributi:**

- `-pois : List<Poi>`  
Lista dei POI.
- `-listView : ListView`  
Vista che visualizza gli oggetti di una lista in modo verticale tramite *scrolling*.
- `-presenter : PoiPresenter`  
Riferimento al presenter.

- **Metodi:**

- `+onActivityCreated( savedInstanceState : Bundle ) : void`  
Viene utilizzato quando la *view* è già stata creata e serve ad esempio per impostare uno stato precedentemente utilizzato.

**Argomenti:**

- \* `savedInstanceState` : se il Fragment viene ricostruito da uno stato precedente, `savedInstanceState` indica lo stato precedente.

- `+setMenuVisibility( menuVisible : boolean ) : void`  
Utilizzato per comunicare informazioni quando il fragment è visualizzato a schermo.

**Argomenti:**

- \* `menuVisible` : visibilità del fragment.

- `+onItemClick( arg0 : AdapterView<?>, arg1 : View, position : int, id : long ) : void`

**Argomenti:**

- \* `arg0` : riferimento all'adapter.
- \* `arg1` : riferimento alla view.
- \* `position` : posizione della view nell'adapter.
- \* `id` : id della posizione selezionata.

- `+setPois( pois : List<Poi> ) : void`  
Imposta i POI da visualizzare.

**Argomenti:**

- \* `pois` : lista dei POI da visualizzare.

- `+setPresenter( presenter : PoiPresenter ) : void`  
Imposta il riferimento al presenter.

**Argomenti:**

- \* `presenter` : riferimento al presenter.

- `+getPresenter( ) : PoiPresenter`  
Ritorna il riferimento al presenter.

### 3.25.2.2 shike::app::view::poi::PoiView

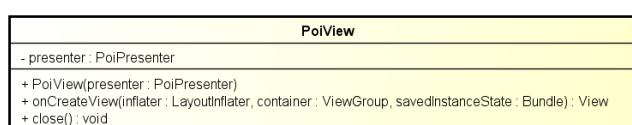


Figura 101: Diagramma di PoiView

- **Tipo:** concreta
- **Descrizione:** classe che modella la visualizzazione delle informazioni globali su un punto di interesse.
- **Superclassi:**
  - android.support.v4.app.DialogFragment
- **Attributi:**
  - `—presenter : PoiPresenter`  
Riferimento al presenter.
- **Metodi:**
  - `+onCreateView( inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle ) : View`  
Crea la vista assegnando le varie azioni ai pulsanti incaricati.  
**Argomenti:**
    - \* `inflater` : l'oggetto inflater viene utilizzato per inserire nuove view nel Fragment.
    - \* `container` : se non nullo, indica la view padre a cui il Fragment viene assegnato.
    - \* `savedInstanceState` : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.
  - `+PoiView( presenter : PoiPresenter )` :  
Costruttore di PoiView.  
**Argomenti:**
    - \* `presenter` : riferimento al presenter.
  - `+close( ) : void`  
Chiude il dialogFragment PoiView.

### 3.25.2.3 shike::app::view::poi::PoiViewAdapter

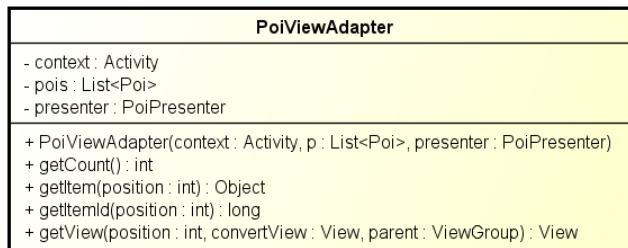


Figura 102: Diagramma di PoiViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei punti di interesse.
- **Superclassi:**
  - android.widget.BaseAdapter

- **Attributi:**

- `_pois : List<Poi>`  
Lista dei POI da visualizzare.
- `_context : Activity`  
Riferimento all'activity associata.
- `_presenter : PoiPresenter`  
Riferimento al presenter.

- **Metodi:**

- `+getView( position : int, convertView : View, parent : ViewGroup ) : View`  
Restituisce la visualizzazione dei dati.

**Argomenti:**

- \* `position` : la posizione all'interno della lista che vogliamo utilizzare.
- \* `convertView` : se presenta indica la vecchia View da riutilizzare.
- \* `parent` : se presente indica la View padre a cui associarsi.

- `+getCount() : int`  
Indica quanti *item* sono rappresentati dall' *adapter*.

- `+getItem( position : int ) : Object`  
Ritorna l'oggetto presente nella posizione indicata.

**Argomenti:**

- \* `position` : indica la posizione dell'oggetto da ritornare.

- `+getItemId( position : int ) : long`  
Ritorna l'indice dell'oggetto presente nella posizione indicata.

**Argomenti:**

- \* `position` : indica la posizione dell'oggetto di cui vogliamo far ritornare l'id.

- `+PoiViewAdapter( context : Activity, pois : List<Poi> ) :`  
Costruttore di PoiViewAdapter.

**Argomenti:**

- \* `context` : riferimento all'activity associata.

- \* `pois` : lista dei POI da visualizzare.

### 3.26 shike::app::view::session

#### 3.26.1 Informazioni sul package

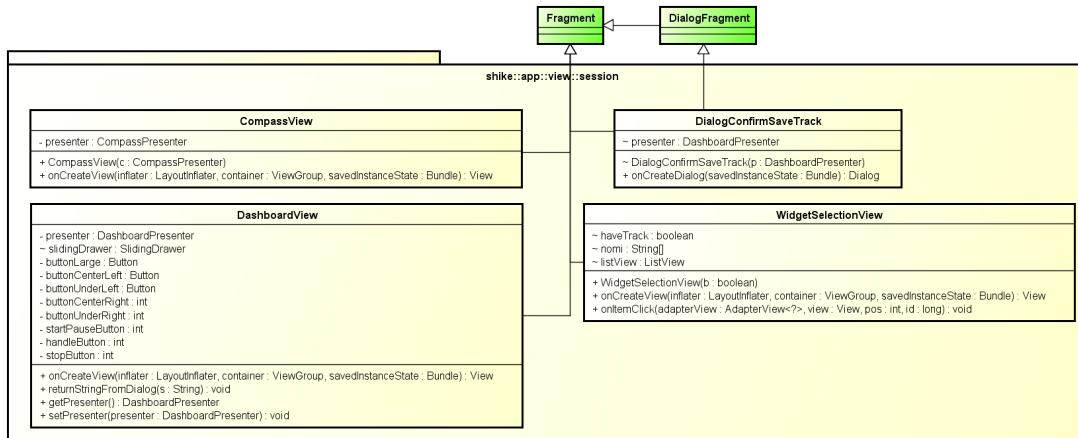


Figura 103: Diagramma di shike::app::view::session

- **Descrizione:** componente contenente le *view* che vengono visualizzate durante una sessione.
- **Componente padre:** `shike::app::view`
- **Interazioni con altri componenti**
  - `shike::app::presenter`

#### 3.26.2 Classi

##### 3.26.2.1 shike::app::view::session::DashboardView

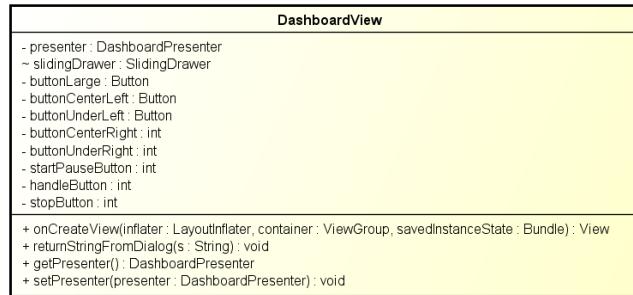


Figura 104: Diagramma di DashboardView

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta la visualizzazione della *dashboard* e dei *widget* in essa contenuti.
- **Superclassi:**
  - `android.support.v4.app.Fragment`

- **Attributi:**

- `buttonCenterRight : Button`  
Bottone medio centrale destro.
- `buttonCenterLeft : Button`  
Bottone medio centrale sinistro.
- `buttonUnderLeft : Button`  
Bottone medio in basso sinistro.
- `buttonLarge : Button`  
Bottone largo centrale.
- `buttonUnderRight : Button`  
Bottone medio in basso destro.
- `startPauseButton : Button`  
Bottone di *start* e pausa utilizzato per il cronometro.
- `stopButton : Button`  
Bottone di stop utilizzato per il cronometro.
- `handleButton : Button`  
Utilizzato per visualizzare la barra di controllo della sessione.
- `slidingDrawer : SlidingDrawer`  
Barra utilizzata per gestire la sessione.
- `presenter : DashboardPresenter`  
*Presenter* utilizzato per effettuare le operazioni.

- **Metodi:**

- `+getButtonLarge() : Button`  
Restituisce la variabile `buttonLarge`.
- `+setButtonLarge( buttonLarge : Button ) : void`  
**Argomenti:**
  - \* `buttonLarge` : valore da impostare nella variabile `buttonLarge`.
- `+onCreateView( inflater : LayoutInflator, container : ViewGroup, savedInstanceState : Bundle ) : View`  
Crea la vista assegnando le varie azioni ai pulsanti incaricati.

**Argomenti:**

- \* `inflater` : l'oggetto `inflater` viene utilizzato per inserire nuove view nel Fragment.
- \* `container` : se non nullo, indica la view padre a cui il Fragment viene assegnato.
- \* `savedInstanceState` : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.
- `+returnStringFromDialog( s : String ) : void`  
Passa la stringa tra due *Fragment*.

**Argomenti:**

- \* `s` : messaggio da passare tra i Fragment.

### 3.26.2.2 shike::app::view::session::CompassView

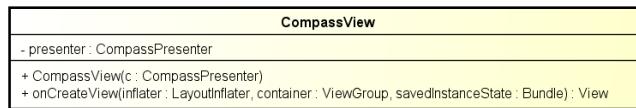


Figura 105: Diagramma di CompassView

- **Tipo:** concreta
- **Descrizione:** classe che visualizza la bussola e se si sta seguendo un percorso salvato nel dispositivo indica anche il prossimo punto da raggiungere.
- **Superclassi:**
  - android.support.v4.app.Fragment
- **Attributi:**
  - `_imageCompass : ImageView`  
Riferimento all'immagine che mostra la bussola.
  - `_imageArrow : ImageView`  
Riferimento all'immagine che mostra la freccia che indica il prossimo punto da raggiungere nel caso si stia seguendo un percorso.
- **Metodi:**
  - `+onCreateView( inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle ) : View`  
Crea la vista assegnando le varie azioni ai pulsanti incaricati.
  - Argomenti:**
    - \* `inflater` : l'oggetto inflater viene utilizzato per inserire nuove view nel Fragment.
    - \* `container` : se non nullo, indica la view padre a cui il Fragment viene assegnato.
    - \* `savedInstanceState` : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.

### 3.26.2.3 shike::app::view::session::WidgetSelectionView

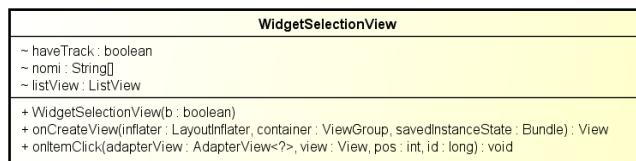


Figura 106: Diagramma di WidgetSelectionView

- **Tipo:** concreta
- **Descrizione:** vista che visualizza i possibili *widget* utilizzabili.
- **Superclassi:**

- android.support.v4.app.DialogFragment

- **Attributi:**

- `~mylist : ListView`  
Lista delle *View* utilizzate.

- `~nomi : String[]`  
Contiene i nomi dei possibili *widget* utilizzabili.

- **Metodi:**

- `+onCreateView( inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle ) : View`  
Crea la vista assegnando le varie azioni ai pulsanti incaricati.

**Argomenti:**

- \* `inflater` : l'oggetto inflater viene utilizzato per inserire nuove view nel Fragment.
- \* `container` : se non nullo, indica la view padre a cui il Fragment viene assegnato.
- \* `savedInstanceState` : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.

### 3.26.2.4 shike::app::view::session::DialogConfirmSaveTrack



Figura 107: Diagramma di DialogConfirmSaveTrack

- **Tipo:** concreta

- **Descrizione:** crea un *dialog* chiede conferma per salvare il percorso.

- **Attributi:**

- `~presenter : DashboardPresenter`  
Riferimento al *presenter* della *dashboard*.

- **Metodi:**

- `+DialogConfirmSaveTrack( ) :`  
Costruttore della classe.
- `+onCreateDialog( ) : Dialog`  
Imposta i messaggi e le relative azioni ai pulsanti Ok e Annulla.

### 3.27 shike::app::view::weather

#### 3.27.1 Informazioni sul package

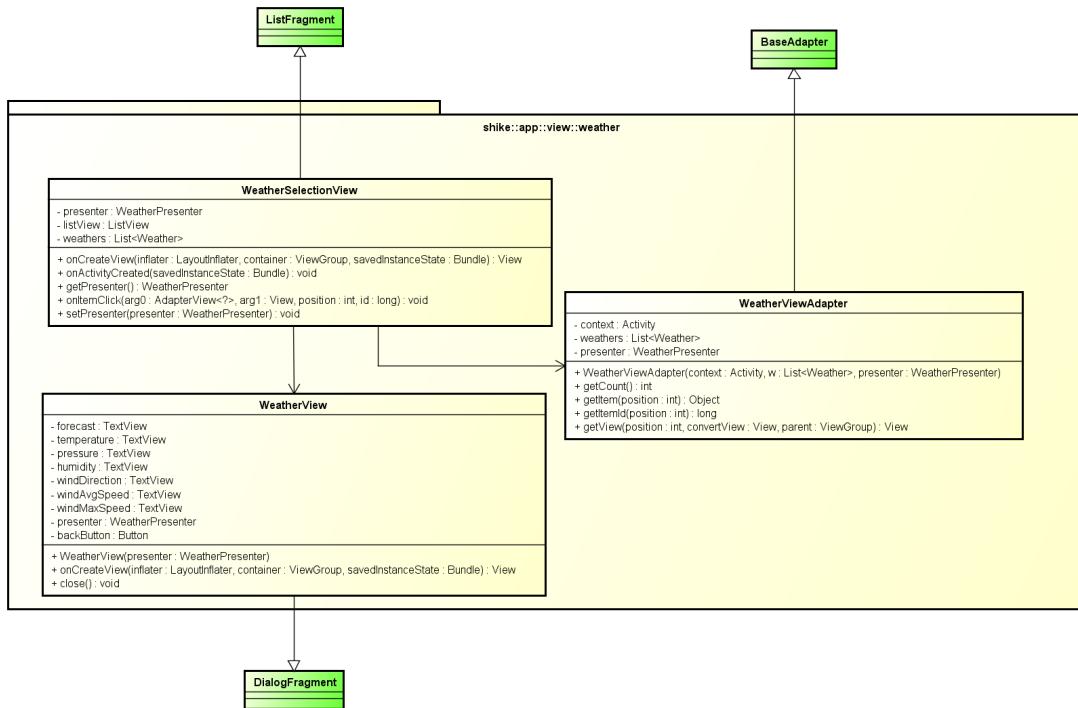


Figura 108: Diagramma di `shike::app::view::weather`

- Descrizione:** componente contenente le *view* che si occupano di visualizzare le informazioni sul meteo.
- Componente padre:** `shike::app::view`
- Interazioni con altri componenti**
  - `shike::app::presenter`

#### 3.27.2 Classi

##### 3.27.2.1 shike::app::view::weather::WeatherView

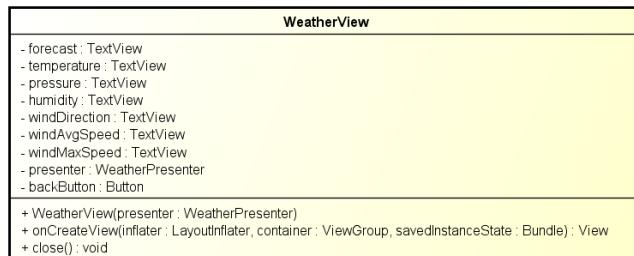


Figura 109: Diagramma di `WeatherView`

- **Tipo:** concreta

- **Descrizione:** classe che modella la visualizzazione delle informazioni meteo salvate.

- **Superclassi:**

- android.support.v4.app.DialogFragment

- **Attributi:**

- `–forecast : TextView`

Tipologia di meteo prevista.

- `–temperature : TextView`

Temperatura prevista sul tracciato.

- `–pressure : TextView`

Pressione prevista sul tracciato.

- `–humidity : TextView`

Percentuale di umidità prevista sul tracciato.

- `–windDirection : TextView`

Direzione del vento prevista sul tracciato.

- `–windAvgSpeed : TextView`

Velocità media del vento prevista sul tracciato.

- `–windMaxSpeed : TextView`

Velocità massima del vento prevista sul tracciato.

- `–presenter : WeatherPresenter`

Riferimento al presenter.

- `–backButton : Button`

Bottone per tornare alla lista dei meteo.

- **Metodi:**

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

Crea la vista assegnando le varie azioni ai pulsanti incaricati.

**Argomenti:**

- \* `inflater` : l'oggetto inflater viene utilizzato per inserire nuove view nel Fragment.

- \* `container` : se non nullo, indica la view padre a cui il Fragment viene assegnato.

- \* `savedInstanceState` : se non nullo, il Fragment viene ricostruito dal precedente stato salvato.

- `+WeatherView( presenter : WeatherPresenter ) :`

Costruttore di WeatherView.

**Argomenti:**

- \* `presenter` : riferimento al presenter.

- `+close( ) : void`

Chiude il dialogFragment WeatherView.

### 3.27.2.2 shike::app::view::weather::WeatherSelectionView

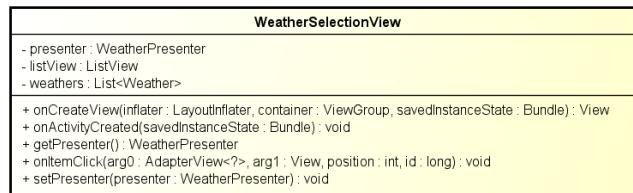


Figura 110: Diagramma di WeatherSelectionView

- **Tipo:** concreta
- **Descrizione:** classe che modella la visualizzazione della lista delle informazioni meteo previste.
- **Superclassi:**
  - android.support.v4.app.ListFragment
- **Attributi:**
  - **presenter** : WeatherPresenter  
Riferimento al presenter.
  - **listView** : ListView  
Lista degli elementi da visualizzare.
  - **weathers** : List<Weather>  
Lista delle previsioni meteo da visualizzare.
- **Metodi:**
  - **+onActivityCreated( savedInstanceState : Bundle ) : void**  
Viene utilizzato quando la view è già stata creata e serve ad esempio per impostare uno stato precedentemente utilizzato.  
**Argomenti:**
    - \* **savedInstanceState** : se il Fragment viene ricostruito da uno stato precedente, savedInstanceState indica lo stato precedente.
  - **+setMenuVisibility( menuVisible : boolean ) : void**  
Utilizzato per comunicare informazioni quando il fragment è visualizzato a schermo.  
**Argomenti:**
    - \* **menuVisible** : visibilità del fragment.
  - **+onItemClick( arg0 : AdapterView<?>, arg1 : View, position : int, id : long ) : void**  
**Argomenti:**
    - \* **arg0** : riferimento all'adapter.
    - \* **arg1** : riferimento alla view.
    - \* **position** : posizione della view nell'adapter.
    - \* **id** : id della posizione selezionata.
  - **+getPresenter( ) : WeatherPresenter**  
Ritorna il riferimento al presenter.

- `+setPresenter( presenter : WeatherPresenter ) : void`  
Imposta il riferimento al presenter.

**Argomenti:**

- \* `presenter` : riferimento al presenter.

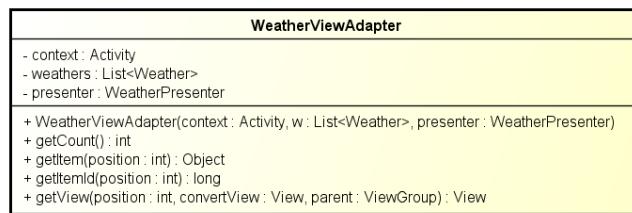
**3.27.2.3 shike::app::view::weather::WeatherViewAdapter**

Figura 111: Diagramma di WeatherViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei meteo.
- **Superclassi:**
  - android.widget.BaseAdapter
- **Attributi:**
  - `–presenter : WeatherPresenter`  
Riferimento al presenter.
  - `–context : Activity`  
Riferimento all'activity.
  - `–weathers : List<Weather>`  
Lista dei meteo da visualizzare.
- **Metodi:**
  - `+getView( position : int, convertView : View, parent : ViewGroup ) : View`  
Restituisce la visualizzazione dei dati.  
**Argomenti:**
    - \* `position` : la posizione all'interno della lista che vogliamo utilizzare.
    - \* `convertView` : se presenta indica la vecchia View da riutilizzare.
    - \* `parent` : se presente indica la View padre a cui associarsi.
  - `+getCount( ) : int`  
Indica quanti *item* sono rappresentati dall' *adapter*.
  - `+getItem( position : int ) : Object`  
Ritorna l'oggetto presente nella posizione indicata.  
**Argomenti:**
    - \* `position` : indica la posizione dell'oggetto da ritornare.
  - `+getItemId( position : int ) : long`  
Ritorna l'indice dell'oggetto presente nella posizione indicata.  
**Argomenti:**
    - \* `position` : indica la posizione dell'oggetto da ritornare.

- \* `position` : indica la posizione dell'oggetto di cui vogliamo far ritornare l'id.
- `+WeatherViewAdapter( context : Activity, weathers : List<Weather> )` :  
Costruttore di WeatherViewAdapter.

**Argomenti:**

- \* `context` : riferimento all'activity.
- \* `weathers` : lista dei meteo da visualizzare.

### 3.28 shike::app::helper

#### 3.28.1 Informazioni sul package

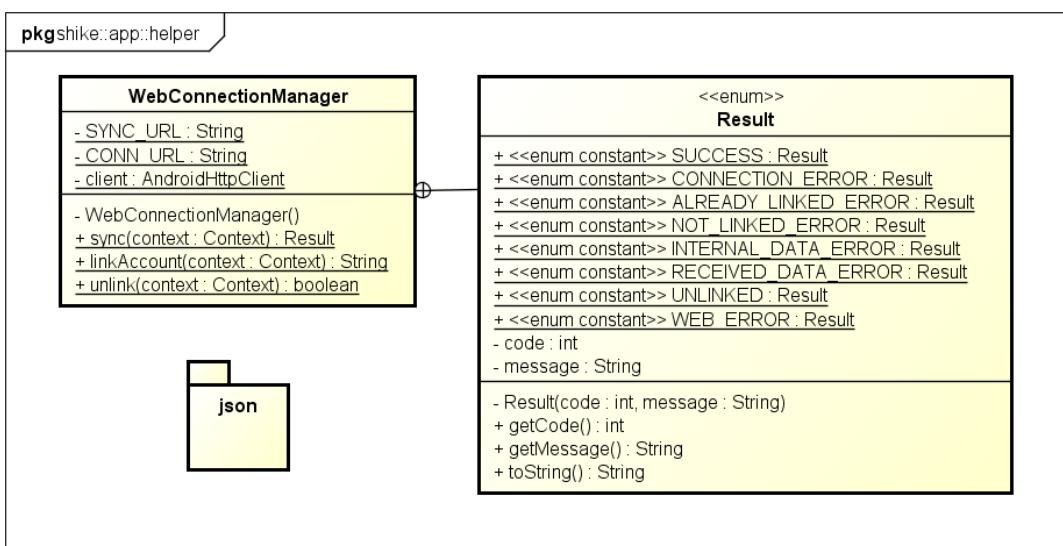


Figura 112: Diagramma di shike::app::helper

- **Descrizione:** componente che raccoglie le componenti di supporto all'applicazione.
- **Componenti contenute**
  - `shike::app::helper::json`
- **Componente padre:** `shike::app`
- **Interazioni con altri componenti**
  - `shike::app::presenter`
  - `shike::app::model`

### 3.28.2 Classi

#### 3.28.2.1 shike::app::helper::WebConnectionManager

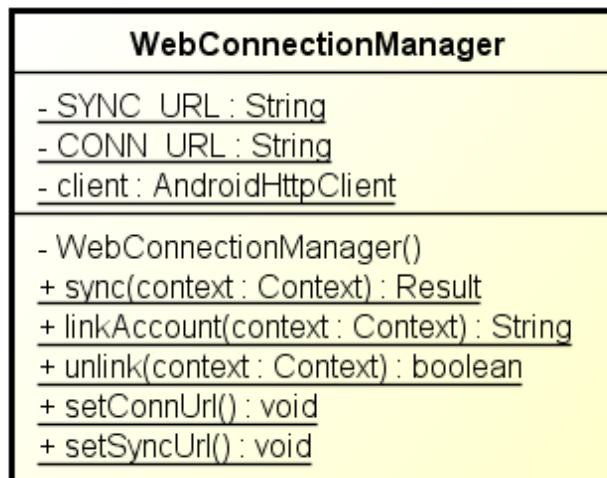


Figura 113: Diagramma di WebConnectionManager

- **Tipo:** concreta
- **Descrizione:** classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.

- **Attributi:**

- –CONN\_URL : String URL di connessione per il collegamento account.
- –SYNC\_URL : String URL di connessione per la sincronizzazione.
- –client : AndroidHttpClient  
Client HTTP su cui vengono eseguite le richieste verso la piattaforma web.

- **Metodi:**

- –WebConnectionManager( ) :  
Costruttore privato della classe.
- +sync( context : Context ) : Result  
Effettua tutte le azioni necessarie per sincronizzare l'applicazione con la piattaforma web. Viene utilizzato il token di connessione per confermare l'identità dell'utente che richiede la sincronizzazione. Se il token è errato (perché l'account è stato scollegato dall'app o il dispositivo è stato scollegato dal web), i dati memorizzati nel database dell'applicazione vengono cancellati (token compreso), riportando l'app allo stato iniziale.

**Argomenti:**

- \* context : contesto dell'applicazione, necessario per l'accesso al database.
- +linkAccount( context : Context ) : String  
Richiede al server il codice da utilizzare per effettuare il collegamento dell'account al dispositivo, e un token di connessione temporaneo. Il codice verrà mostrato a schermo durante la prima connessione e l'utente dovrà immetterlo nella sua pagina

della piattaforma web. Se il codice inserito è corretto, il collegamento è completato e il token di connessione ricevuto dal web viene associato all'account da cui è stato inserito. Se il codice è errato, il token viene cancellato dopo un certo periodo di tempo. Ritorna il codice usa e getta numerico di 5 cifre. Se si sono verificati problemi nella comunicazione con la piattaforma web, ritorna una stringa alfanumerica di errore.

**Argomenti:**

- \* `context` : contesto dell'applicazione, necessario per l'accesso al database.
- `+unlinkAccount( context : Context ) : boolean`  
Scollega l'account dall'applicazione, cancellando di conseguenza tutti i dati utente. Ritorna `true` se la cancellazione è avvenuta con successo, `false` altrimenti.

**Argomenti:**

- \* `context` : contesto dell'applicazione, necessario per l'accesso al database.

### 3.28.2.2 shike::app::helper::WebConnectionManager::Result

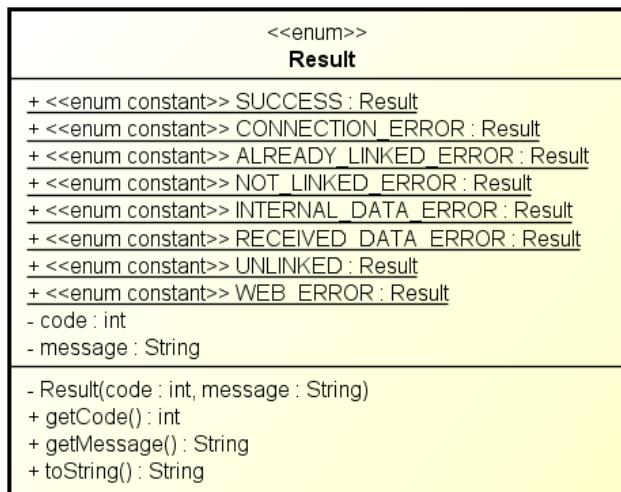


Figura 114: Diagramma di WebConnectionManager::Result

- **Tipo:** concreta
- **Descrizione:** enum interna a shike::app::helper::WebConnectionManager che indica i possibili risultati ottenibili da un'operazione di sincronizzazione o collegamento *account*.
- **Attributi:**
  - `_code : int`  
Codice numerico che identifica il risultato.
  - `_message : String`  
Messaggio che definisce il risultato.
- **Metodi:**
  - `+getCode() : int`  
Restituisce la variabile `code`.

- **+getMessage() : String**  
Restituisce la variabile `message`.
  - **-WebConnectionManager.Result( code : int, message : String ) :**  
Costruttore privato della classe.
- Argomenti:**
- \* `code` : codice identificativo del risultato.
  - \* `message` : messaggio del risultato.
- **+toString() : String**  
Fornisce una rappresentazione in stringa dell'oggetto nel seguente modo: `code: message`.

### 3.29 shike::app::helper::json

#### 3.29.1 Informazioni sul package

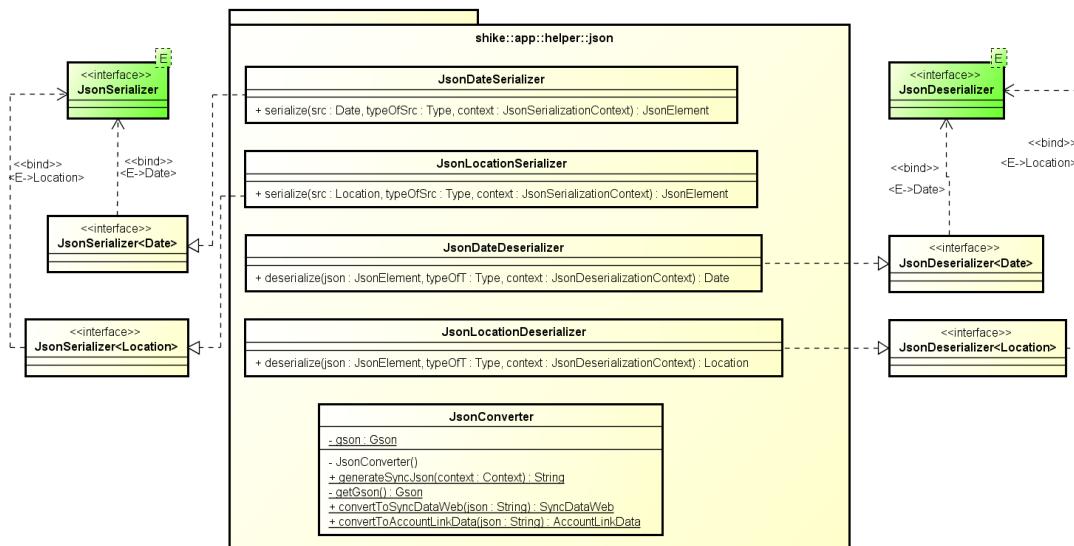


Figura 115: Diagramma di `shike::app::helper::json`

- **Descrizione:** componente contenente le classi utilizzate per la conversione da e verso JSON
- **Componente padre:** `shike::app::helper`

#### 3.29.2 Classi

##### 3.29.2.1 shike::app::helper::json::JsonDateSerializer



Figura 116: Diagramma di `JsonDateSerializer`

- **Tipo:** concreta

- **Descrizione:** serializzatore personalizzato per il tipo `Date` di Java. È stato creato perché la serializzazione di default comporta perdita di informazioni (i millisecondi vengono troncati). Implementa `JsonSerializer` utilizzando `Date` come tipo T.

- **Implementa:**

- `com.google.gson.JsonSerializer`

- **Metodi:**

- `+serialize( src : T, typeOfSrc : Type, context : JsonSerializationContext ) : JsonElement`

Metodo invocato automaticamente da `Gson` quando, durante la serializzazione, incontra un oggetto di tipo T.

**Argomenti:**

- \* `src` : oggetto da serializzare.

- \* `typeOfSrc` : tipo dell'oggetto sorgente.

- \* `context` : contesto di serializzazione in `JSON`, utile per invocare la serializzazione di default per le parti dell'oggetto che non richiedono serializzazioni personalizzate.

### 3.29.2.2 shike::app::helper::json::JsonDateDeserializer



Figura 117: Diagramma di JsonDateDeserializer

- **Tipo:** concreta

- **Descrizione:** deserializzatore personalizzato per il tipo `Date` di Java. Creato per via del serializzatore personalizzato, `shike::app::helper::json::JsonDateSerializer`, che serializza la data in maniera diversa da quella di default. Implementa `JsonDeserializer` utilizzando `Date` come tipo T.

- **Implementa:**

- `com.google.gson.JsonDeserializer`

- **Metodi:**

- `+deserialize( json : JsonElement, typeOfSrc : Type, context : JsonDeserializationContext ) : T`

Metodo invocato automaticamente da `Gson` quando, durante la deserializzazione, deve convertire un elemento JSON in un oggetto di tipo T.

**Argomenti:**

- \* `json` : elemento `JSON` da convertire in un oggetto di tipo T.

- \* `typeOfSrc` : tipo dell'oggetto da ritornare.

- \* `context` : contesto di deserializzazione in `JSON`, utile per invocare la deserializzazione di default per le parti dell'oggetto che non richiedono implementazioni personalizzate.

### 3.29.2.3 shike::app::helper::JsonConverter

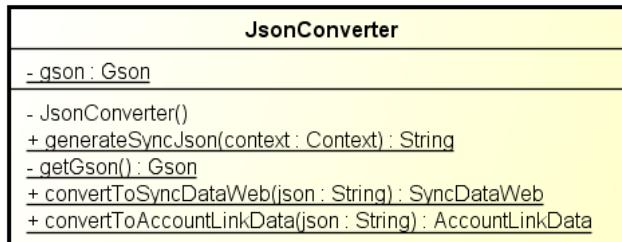


Figura 118: Diagramma di JsonConverter

- **Tipo:** concreta
- **Descrizione:** classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa. Utilizza la libreria Gson di Google per effettuare la conversione in entrambi i sensi.
- **Attributi:**
  - –gson : Gson Riferimento al convertitore Gson che si occupa di effettuare tutte le conversioni.
- **Metodi:**
  - –getGson( ) : Gson  
Inizializzatore *lazy* dell'attributo `gson`. Utilizza la classe `GsonBuilder` per creare un convertitore personalizzato con i de/serializzatori definiti nel package corrente.
  - +generateSyncJson( ) : String  
Genera una stringa in formato JSON contenente tutti i dati da inviare alla piattaforma web durante la sincronizzazione.
  - –JsonConverter( ) :  
Costruttore privato della classe.
  - +convertToSyncDataWeb( json : String ) : SyncDataWeb  
Genera un oggetto `SyncDataWeb` a partire dal JSON ricevuto dalla piattaforma web durante la sincronizzazione.
- **Argomenti:**
  - \* `json` : stringa JSON contenente la serializzazione di un oggetto di tipo `SyncDataWeb`.
- +convertToAccountLinkData( json : String ) : AccountLinkData  
Genera un oggetto `AccountLinkData` a partire dal JSON ricevuto dalla piattaforma web durante il collegamento dell'account.
- **Argomenti:**
  - \* `json` : stringa JSON contenente la serializzazione di un oggetto di tipo `AccountLinkData`.

### 3.29.2.4 shike::app::helper::json::JsonLocationSerializer



Figura 119: Diagramma di JsonLocationSerializer

- **Tipo:** concreta
- **Descrizione:** serializzatore personalizzato per il tipo `Location` di Android. È stato creato perché le classi `Location` utilizzate nella parte app e nella parte web hanno implementazioni diverse.
- **Implementa:**
  - `com.google.gson.JsonSerializer`
- **Metodi:**
  - `+serialize( src : T, typeOfSrc : Type, context : JsonSerializationContext ) : JsonElement`  
Metodo invocato automaticamente da `Gson` quando, durante la serializzazione, incontra un oggetto di tipo `T`.  
**Argomenti:**
    - \* `src` : oggetto da serializzare.
    - \* `typeOfSrc` : tipo dell'oggetto sorgente.
    - \* `context` : contesto di serializzazione in `JSON`, utile per invocare la serializzazione di default per le parti dell'oggetto che non richiedono serializzazioni personalizzate.

### 3.29.2.5 shike::app::helper::json::JsonLocationDeserializer



Figura 120: Diagramma di JsonLocationDeserializer

- **Tipo:** concreta
- **Descrizione:** deserializzatore personalizzato per il tipo `Location` di Android. Si occupa di convertire le posizioni ricevute dalla parte web in `Location`, in quanto l'implementazione della classe omonima nella parte web è diversa da quella della parte app.
- **Implementa:**
  - `com.google.gson.JsonDeserializer`
- **Metodi:**
  - `+ deserialize(json : JsonElement, typeOfT : Type, context : JsonDeserializationContext) : Location`

– `+deserialize( json : JsonElement, typeOfSrc : Type, context : JsonDeserializationContext ) : T`

Metodo invocato automaticamente da `Gson` quando, durante la deserializzazione, deve convertire un elemento JSON in un oggetto di tipo T.

#### Argomenti:

- \* `json` : elemento JSON da convertire in un oggetto di tipo T.
- \* `typeOfSrc` : tipo dell'oggetto da ritornare.
- \* `context` : contesto di deserializzazione in JSON, utile per invocare la deserializzazione di default per le parti dell'oggetto che non richiedono implementazioni personalizzate.

### 3.30 shike::web

#### 3.30.1 Informazioni sul package

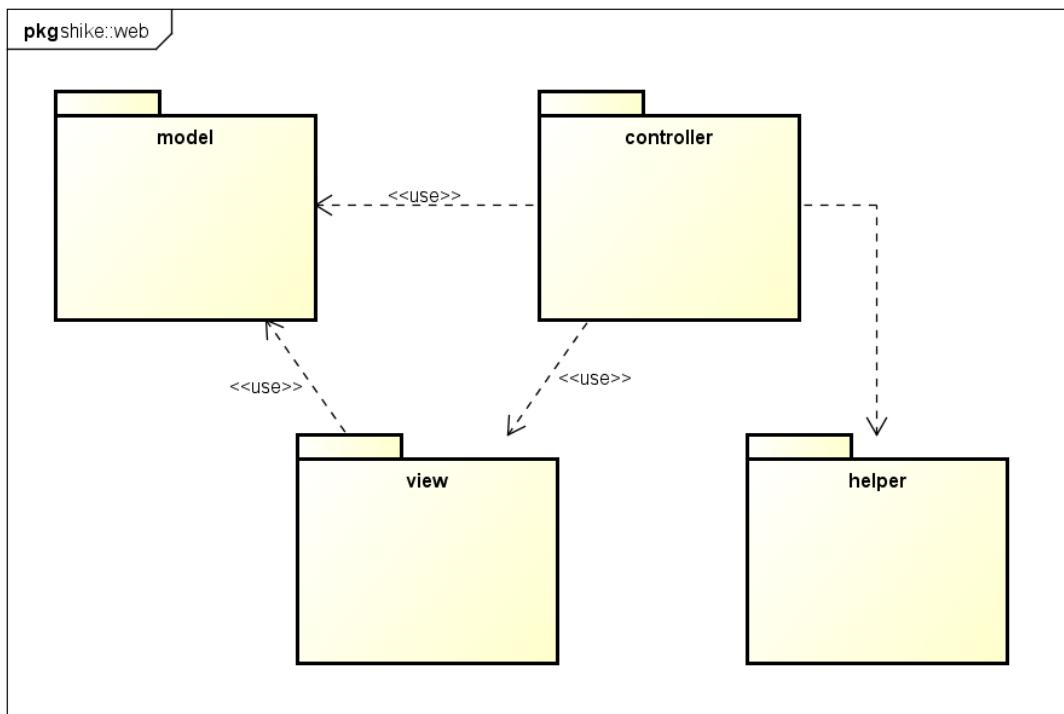


Figura 121: Diagramma di shike::web

- **Descrizione:** componente che contiene la piattaforma web del prodotto sHike.
- **Componenti contenute**
  - shike::web::controller
  - shike::web::view
  - shike::web::helper
  - shike::web::model
- **Componente padre:** shike

### 3.31 shike::web::controller

#### 3.31.1 Informazioni sul package

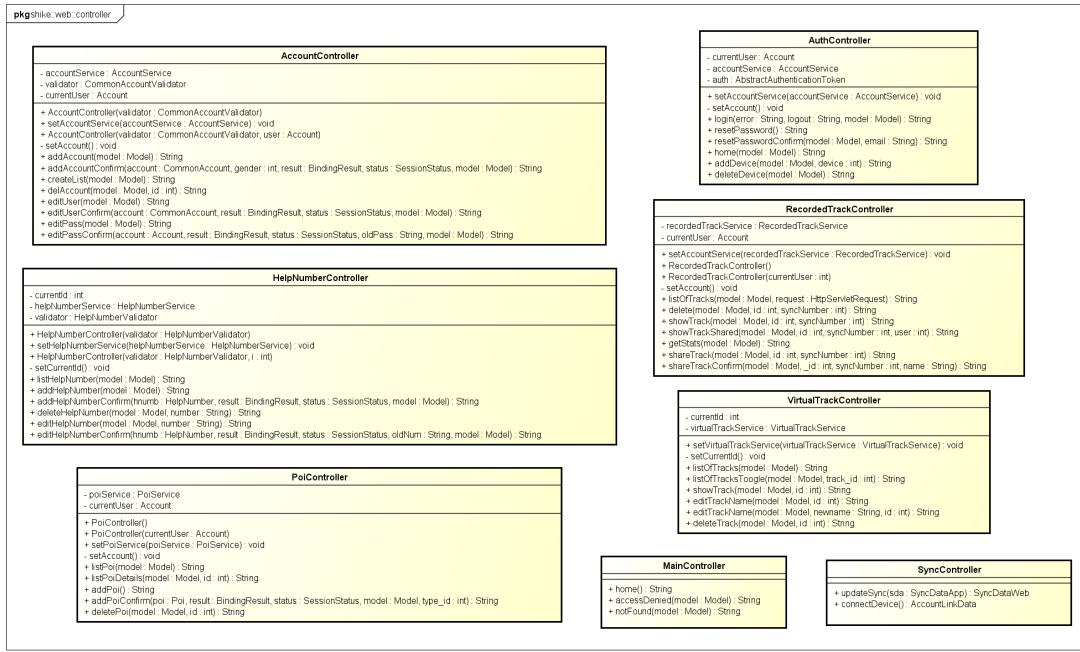


Figura 122: Diagramma di shike::web::controller

- Descrizione:** componente *Controller* dell'architettura MVC della parte web. Essa esegue le operazioni che l'utente ha richiesto tramite la *View* agendo se necessario sul *Model*.
- Componente padre:** shike::web
- Interazioni con altri componenti**
  - shike::web::model
  - shike::web::view
  - shike::web::helper

#### 3.31.2 Classi

##### 3.31.2.1 shike::web::controller::MainController

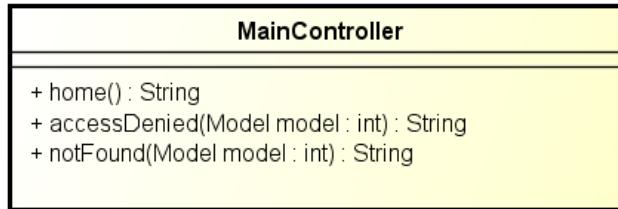


Figura 123: Diagramma di MainController

- Tipo:** concreta

- **Descrizione:** Controller base che permette il *mapping* delle pagine principali della piattaforma come la *home* e quelle di errore (403 e 404).

- **Metodi:**

- `+notFound( model : Model ) : String`

*Mapping* per la pagina di errore 404. Se una pagina non viene trovata all'interno del server viene visualizzata questa pagina.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- `+home() : String`

*Mapping* della pagina home.

- `+accessDenied( model : Model ) : String`

*Mapping* per la pagina di errore 403. Se un utente non ha il permesso di accedere ad una certa sezione viene visualizzata questa pagina.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

### 3.31.2.2 shike::web::controller::RecordedTrackController

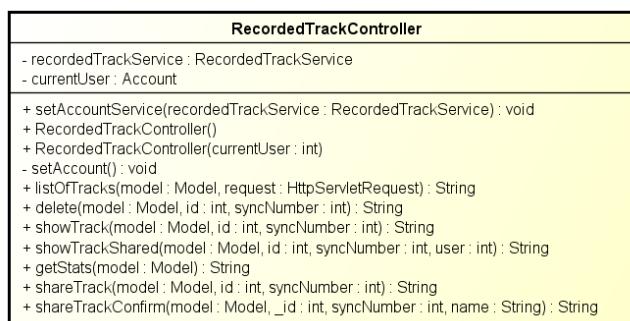


Figura 124: Diagramma di RecordedTrackController

- **Tipo:** concreta

- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.

- **Attributi:**

- `-currentUser : Account`  
Account dell'utente corrente.

- `-recordedTrackService : RecordedTrackService`  
Fornisce un oggetto di tipo *RecordedTrackService* che permette l'accesso ai dati del *database*.

- **Metodi:**

- `+setAccount() : void`

Metodo che permette di leggere e salvare l'*account* dell'utente che ha effettuato il *login*.

- `+listOfTracks( model : Model, request : HttpServletRequest ) : String`  
Restituisce la lista dei tracciati associati ad un utente.  
**Argomenti:**
    - \* `model` : modello dati del pattern MVC di Spring.
    - \* `request` : richiesta HTTP del server.
  - `+delete( syncNumber : int, model : Model, id : int ) : String`  
Permette all'utente di eliminare un proprio tracciato.  
**Argomenti:**
    - \* `syncNumber` : numero di sincronizzazione del tracciato.
    - \* `model` : modello dati del pattern MVC di Spring.
    - \* `id` : viene passato come variabile d'indirizzo l'id del percorso da eliminare.
  - `+shareTrack( model : Model, type : int, id : int ) : String`  
Permette la condivisione di un tracciato nei principali *social* o nella piattaforma web.  
**Argomenti:**
    - \* `model` : modello dati del pattern MVC di Spring.
    - \* `type` : tipo di condivisione, passato come variabile d'indirizzo.
    - \* `id` : id del tracciato da condividere passata come variabile d'indirizzo.
  - `+showTrack( syncNumber : int, model : Model, id : int ) : String`  
Visualizza i dettagli di un percorso mostrando il tracciato sulla mappa e le *performance* di quel percorso.  
**Argomenti:**
    - \* `syncNumber` : numero di sincronizzazione del tracciato.
    - \* `model` : modello dati del pattern MVC di Spring.
    - \* `id` : id del tracciato del quale si vogliono visualizzare le informazioni.
  - `+showTrackShared( model : Model, id : int, syncNumber : int, user : int ) : String`  
Visualizza i dettagli di un percorso che è stato condiviso da un utente, mostrando il tracciato sulla mappa e le *performance* di quel percorso.  
**Argomenti:**
    - \* `model` : modello dati del pattern MVC di Spring.
    - \* `id` : id del tracciato del quale si vogliono visualizzare le informazioni.
    - \* `syncNumber` : numero di sincronizzazione del tracciato.
    - \* `user` : utente che ha condiviso il tracciato.
  - `+shareTrackConfirm( model : Model, _id : int, syncNumber : int, name : String ) : String`
- Argomenti:**
- \* `model` : modello dati del pattern MVC di Spring.
  - \* `_id` : id del tracciato.
  - \* `syncNumber` : numero di sincronizzazione del tracciato.
  - \* `name` : nome del percorso condiviso.
- `+getStats( model : Model ) : String`  
Visualizza le statistiche globali dell'utente.  
**Argomenti:**
    - \* `model` : modello dati del pattern MVC di Spring.
  - `+setAccountService( recordedTrackService : RecordedTrackService ) : void`  
Metodo che permette di impostare il `RecordedTrackService`.  
**Argomenti:**
    - \* `recordedTrackService` : `RecordedTrackService`.

- \* `recordedTrackService` : recordedTrackService da impostare.
- `+RecordedTrackController()` :
  - Costruttore della classe.
- `+RecordedTrackController( currentUser : int )` :
  - Costruttore della classe ad un parametro, utile a livello di test.

**Argomenti:**

- \* `currentUser` : id dell'utente corrente.

### 3.31.2.3 shike::web::controller::PoiController

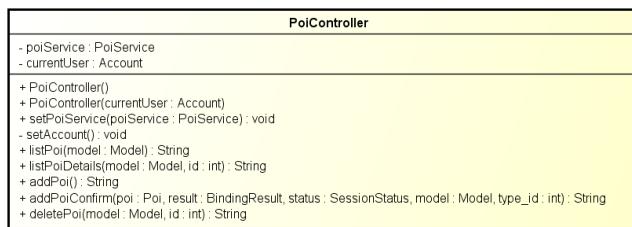


Figura 125: Diagramma di PoiController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
- **Attributi:**
  - `-currentUser : Account`  
Se l'utente ha fatto il *login*, il campo contiene i suoi dati.
  - `-poiService : PoiService`  
Fornisce un oggetto di tipo *PoiService* che permette l'accesso ai dati del *database*.
- **Metodi:**
  - `+setAccount() : void`  
Imposta l'utente corrente nella variabile.
  - `+listPoi( model : Model ) : String`  
Restituisce la lista dei POI.

**Argomenti:**

  - \* `model` : modello dati del pattern MVC di Spring.
  - `+addPoi() : String`  
Fornisce il *form* per l'aggiunta di un nuovo POI all'utente.
  - `+addPoiConfirm( type_id : int, poi : Poi, result : BindingResult, status : SessionStatus, model : Model ) : String`  
Il metodo permette di confermare i dati ricevuti dal *form* di aggiunta di un POI.  
Se non sono riscontrati errori i dati vengono salvati nel *database* e successivamente viene fornito un messaggio di conferma all'utente.

**Argomenti:**

  - \* `type_id` : id del tipo del POI.
  - \* `pai` : fornisce il modello dati riguardante il POI che viene passato dal form.

- \* `result` : contiene i risultati della validazione che viene fatta dal Validator. Se è tutto corretto non saranno presenti errori.
  - \* `status` : stato della sessione corrente.
  - \* `model` : modello dati del pattern MVC di Spring.
  - `+deletePoi( model : Model, id : int ) : String`  
Permette all'utente di eliminare un proprio POI. Se l'utente è un amministratore gli è possibile eliminare qualsiasi POI.
- Argomenti:**
- \* `model` : modello dati del pattern MVC di Spring.
  - \* `id` : id del POI da cancellare.
- `+listPoiDetails( model : Model, int : id ) : String`  
Restituisce il dettaglio di un POI.
- Argomenti:**
- \* `model` : modello dati del pattern MVC di Spring.
  - \* `int` : id del POI di cui si desidera vedere i dettagli.
- `+setPoiService( poiService : PoiService ) :`  
Metodo che permette di impostare il PoiService.
- Argomenti:**
- \* `poiService` : poiService da impostare.
- `+PoiController() :`  
Costruttore della classe.
- `+PoiController( currentUser : Account ) :`  
Costruttore della classe ad un parametro, utile a livello di test.
- Argomenti:**
- \* `currentUser` : id dell'utente che si desidera impostare.

### 3.31.2.4 shike::web::controller::AuthController

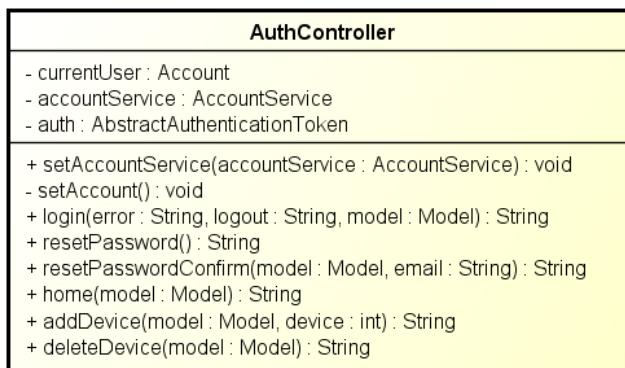


Figura 126: Diagramma di AuthController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
- **Attributi:**

- **`–auth : AbstractAuthenticationToken`**  
Token di autenticazione.
- **`–currentUser : Account`**  
Se l'utente ha fatto il *login*, il campo contiene i suoi dati.
- **`–accountService : AccountService`**  
Se l'utente ha fatto il *login*, il campo contiene i suoi dati.

• **Metodi:**

- **`+login( error : String, logout : String, model : Model ) : String`**  
Fornisce il modulo di *login*. Se è presente il parametro *error* o *logout* imposto il messaggio apposito prima di ritornare la pagina di accesso.

**Argomenti:**

- \* `error` : se viene fornito questo parametro significa che si sono verificati degli errori nel login.
- \* `logout` : se viene fornito questo parametro significa che l'utente vuole fare il logout.
- \* `model` : modello dati del pattern MVC di Spring.

- **`+home( model : Model ) : String`**

Fornisce la *homepage* relativa all'utente che ha eseguito l'accesso, a seconda che sia l'amministratore del portale o un utente semplice.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- **`+resetPasswordConfirm( model : Model, email : String ) : String`**

Riceve i dati dal *form* di recupero *password*. Se l'`email` viene trovata viene reimposta la *password* ed inviata quella nuova all'utente che ne ha fatto richiesta.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.
- \* `email` : email dell'account.

- **`+resetPassword() : String`**

Fornisce il *form* per recuperare la *password* di un *account*.

- **`+setAccountService( accountService : AccountService ) : void`**

Metodo che permette di impostare il `AccountService`.

**Argomenti:**

- \* `accountService` : accountService da impostare.

- **`+setAccount( ) : void`**

Metodo che permette di leggere e salvare l'*account* dell'utente che ha effettuato il *login*.

- **`+addDevice( device : int, model : Model ) : String`**

Permette di effettuare la connessione del dispositivo dal proprio *account*.

**Argomenti:**

- \* `device` : codice fornito dal dispositivo.
- \* `model` : modello dati del pattern MVC di Spring.

- **`+deleteDevice( model : Model ) : String`**

Permette di effettuare la disconnessione del dispositivo dal proprio *account*.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

### 3.31.2.5 shike::web::controller::AccountController

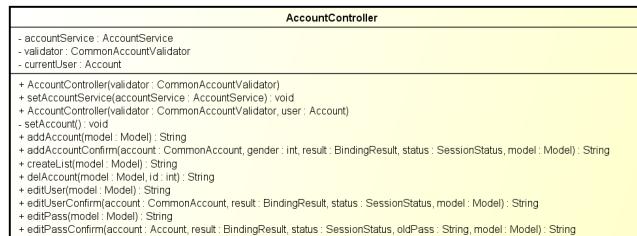


Figura 127: Diagramma di Account Controller

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
- **Attributi:**
  - **–accountService : AccountService**  
Service che permette di recuperare gli Account dal database.
  - **–validator : CommonAccountValidator**  
Validatore dei dati inseriti all'interno dei *form*.
  - **–currentUser : Account**  
Se l'utente ha fatto il *login*, il campo contiene i suoi dati.
- **Metodi:**
  - **+setAccountService( accountService : AccountService ) : void**  
**Argomenti:**
    - \* **accountService** : valore da impostare nella variabile **accountService**.
  - **+addAccount( model : Model ) : String**  
Permette di visualizzare il *form* di registrazione.  
**Argomenti:**
    - \* **model** : modello dati del pattern MVC di Spring.
  - **+addAccountConfirm( account : CommonAccount, result : BindingResult, status : SessionStatus, model : Model, gender : int ) : String**  
Il metodo permette di confermare i dati ricevuti dal *form* di registrazione. Se non sono riscontrati errori i dati vengono salvati nel *database* e successivamente viene fornito un messaggio di conferma all'utente.  
**Argomenti:**
    - \* **account** : fornisce il modello dati riguardante l'account che viene passato dal form.
    - \* **result** : contiene i risultati della validazione che viene fatta dal Validator. Se è tutto corretto non saranno presenti errori.
    - \* **status** : stato della sessione corrente.
    - \* **model** : modello dati del pattern MVC di Spring.
    - \* **gender** : intero che rappresenta il genere dell'utente.

- **+createList( model : Model ) : String**  
Crea una lista di tutti gli utenti, consultabile solo dall'amministratore della piattaforma.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- **+delAccount( model : Model, id : int ) : String**  
Permette all'amministratore della piattaforma di eliminare un *account* di un utente comune.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `id` : viene passato come variabile d'indirizzo l'id dell'account da eliminare.

- **-setAccount( ) : void**

Imposta l'utente corrente nella variable.

- **+editUser( model : Model, account : Account, result : BindingResult, status : SessionStatus ) : String**

Permette di visualizzare il *form* di modifica dati utente.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `account` : account contenente i nuovi dati.

- \* `result` : contiene i risultati della validazione che viene fatta dal Validator. Se è tutto corretto non saranno presenti errori.

- \* `status` : stato della sessione corrente.

- **+editUserConfirm( model : Model ) : String**

Il metodo permette di confermare i dati ricevuti dal *form* di modifica dati. Se non sono riscontrati errori i dati vengono salvati nel *database* e successivamente viene fornito un messaggio di conferma all'utente.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- **+editPass( model : Model, account : Account, result : BindingResult, status : SessionStatus, oldPass : String ) : String**

Permette di visualizzare il *form* di modifica *password*.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `account` : account contenente la nuova password.

- \* `result` : contiene i risultati della validazione che viene fatta dal Validator. Se è tutto corretto non saranno presenti errori.

- \* `status` : stato della sessione corrente.

- \* `oldPass` : stringa contenente la vecchia password.

- **+editPassConfirm( model : Model ) : String**

Il metodo permette di confermare i dati ricevuti dal *form* di modifica *password*. Se non sono riscontrati errori i dati vengono salvati nel *database* e successivamente viene fornito un messaggio di conferma all'utente.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- **+AccountController( validator : CommonAccountValidator ) :**

Viene ridefinito il costruttore di *AccountController*, aggiungendo la gestione del validatore per i *form*.

**Argomenti:**

- \* `validator` : validatore del form.
- `+AccountController( validator : CommonAccountValidator, user : Account )` :

Viene ridefinito il costruttore di `AccountController` a due parametri, aggiungendo la gestione del validatore per i *form* e l'utente corrente. Utile a livello di test.

**Argomenti:**

- \* `validator` : validatore del form.
- \* `user` : utente corrente.

### 3.31.2.6 shike::web::controller::SyncController



Figura 128: Diagramma di SyncController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di sincronizzazione tra dispositivo e piattaforma web.
- **Metodi:**
  - `+updateSync( sda : SyncDataApp ) : SyncDataWeb`  
Permette di effettuare la sincronizzazione dei dati.
  - Argomenti:**
    - \* `sda` : dati ricevuti dall'applicazione.
  - `+connectDevice() : AccountLinkData`  
Permette la connessione alla piattaforma del dispositivo.

### 3.31.2.7 shike::web::controller::HelpNumberController

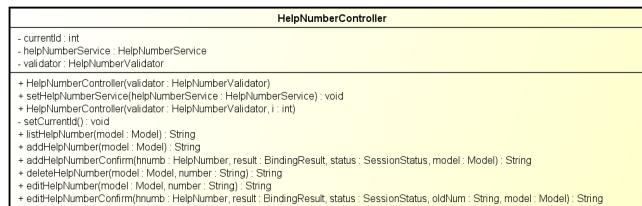


Figura 129: Diagramma di HelpNumberController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numeri di pronto soccorso.
- **Attributi:**

- `validator : HelpNumberValidator`  
Validatore dei dati inseriti all'interno dei *form*.
- `currentId : int`  
Id dell'utente corrente.
- `helpNumberService : HelpNumberService`  
Service che permette di recuperare gli HelpNumber dal database.

• **Metodi:**

- `setCurrentId( ) : void`  
Metodo che permette di leggere e salvare l'id corrente dell' *account* che ha effettuato il *login*.
- `+listHelpNumber( model : Model ) : String`  
Restituisce la lista dei numeri di soccorso associati ad un utente.

**Argomenti:**

\* `model` : modello dati del pattern MVC di Spring.

- `+addHelpNumber( model : Model ) : String`  
Fornisce il *form* per l'aggiunta di un nuovo numero utile all'utente.

**Argomenti:**

\* `model` : modello dati del pattern MVC di Spring.

- `+addHelpNumberConfirm( hnumb : HelpNumber, result : BindingResult, status : SessionStatus, model : Model ) : String`  
Il metodo permette di confermare i dati ricevuti dal *form* di aggiunta del numero utile. Se non sono riscontrati errori i dati vengono salvati nel *database* e successivamente viene fornito un messaggio di conferma all'utente.

**Argomenti:**

\* `hnumb` : fornisce il modello dati riguardante il numero utile che viene passato dal form.

\* `result` : contiene i risultati della validazione che viene fatta dal Validator. Se è tutto corretto non saranno presenti errori.

\* `status` : stato della sessione corrente.

\* `model` : modello dati del pattern MVC di Spring.

- `+deleteHelpNumber( model : Model, number : String ) : String`  
Permette all'utente di eliminare un proprio numero utile.

**Argomenti:**

\* `model` : modello dati del pattern MVC di Spring.

\* `number` : viene passato come variabile d'indirizzo il numero da eliminare.

- `+editHelpNumber( number : String, model : Model ) : String`  
Permette all'utente di modificare un proprio numero utile.

**Argomenti:**

\* `number` : viene passato come variabile d'indirizzo il numero da eliminare.

\* `model` : modello dati del pattern MVC di Spring.

- `+editHelpNumberConfirm( hnumb : HelpNumber, result : BindingResult, status : SessionStatus, oldNum : String, model : Model ) : String`  
Il metodo permette di confermare i dati ricevuti dal *form* di modifica del numero utile. Se non sono riscontrati errori i dati vengono salvati nel *database* e successivamente viene fornito un messaggio di conferma all'utente.

**Argomenti:**

- \* `hnumb` : fornisce il modello dati riguardante il numero utile che viene passato dal form.
- \* `result` : contiene i risultati della validazione che viene fatta dal Validator. Se è tutto corretto non saranno presenti errori.
- \* `status` : stato della sessione corrente.
- \* `oldNum` : viene passato come variabile d'indirizzo il vecchio numero, in modo da avere il riferimento per l'update del database.
- \* `model` : modello dati del pattern MVC di Spring.
- `+HelpNumberController( validator : HelpNumberValidator )` :  
Viene ridefinito il costruttore di `HelpNumberController`, aggiungendo la gestione del validatore per i *form*.
- Argomenti:**
  - \* `validator` : validatore del form.
- `+HelpNumberController( validator : HelpNumberValidator, id : int )` :  
Viene definito un costruttore a due parametri per `HelpNumberController`.
- Argomenti:**
  - \* `validator` : validatore del form.
  - \* `id` : id corrente dell'utente che ha effettuato l'accesso.
- `+setHelpNumberService( helpNumberService : HelpNumberService ) : void`  
Metodo che permette di impostare `HelpNumberService`.
- Argomenti:**
  - \* `helpNumberService` : `helpNumberService` da impostare.

### 3.31.2.8 shike::web::controller::VirtualTrackController

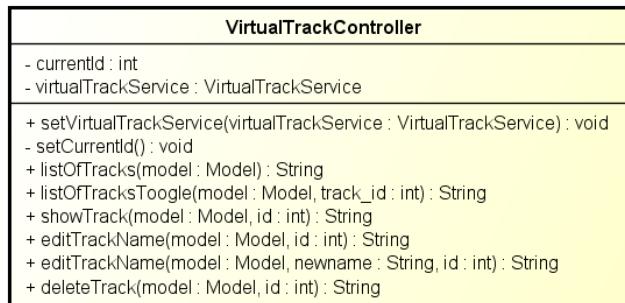


Figura 130: Diagramma di `VirtualTrackController`

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.
- **Attributi:**
  - `-currentId : int`  
Id dell'utente che ha effettuato il login.
  - `-virtualTrackService : VirtualTrackService`  
Fornisce un oggetto di tipo `VirtualTrackService` che permette l'accesso ai dati del *database*.

- Metodi:

- `+listOfTracks( model : Model ) : String`

Restituisce la lista dei percorsi condivisi nella piattaforma.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- `+showTrack( model : Model, id : int ) : String`

Visualizza i dettagli di un percorso mostrando il tracciato sulla mappa.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `id` : id del tracciato del quale si vogliono visualizzare le informazioni.

- `+deleteTrack( model : Model, id : int ) : String`

Elimina un percorso dal database, selezionandolo tramite l'id.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `id` : id del tracciato da eliminare.

- `+editTrackName( model : Model, id : int ) : String`

Visualizza la pagina con le informazioni attuali del tracciato.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `id` : id del tracciato a cui cambiare nome.

- `+editTrackNameConfirm( model : Model, newname : String, id : int ) : String`

Indica all'utente se il nome del tracciato è stato modificato con successo o se c'è stato un errore.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `newname` : nuovo nome del percorso selezionato.

- \* `id` : id del tracciato a cui cambiare nome.

- `+listOfTracksToogle( model : Model, track_id : int ) : String`

Permette di effettuare il cambio di impostazioni per singolo tracciato, decidendo se è da sincronizzare o meno.

**Argomenti:**

- \* `model` : modello dati del pattern MVC di Spring.

- \* `track_id` : id del tracciato da impostare.

- `+setCurrent( ) : void`

Imposta l'id dell'utente corrente.

- `+setVirtualTrackService( virtualTrackService : VirtualTrackService ) : void`

Metodo che permette di impostare il `VirtualTrackService`.

**Argomenti:**

- \* `virtualTrackService` : `virtualTrackService` da impostare.

### 3.32 shike::web::view

#### 3.32.1 Informazioni sul package

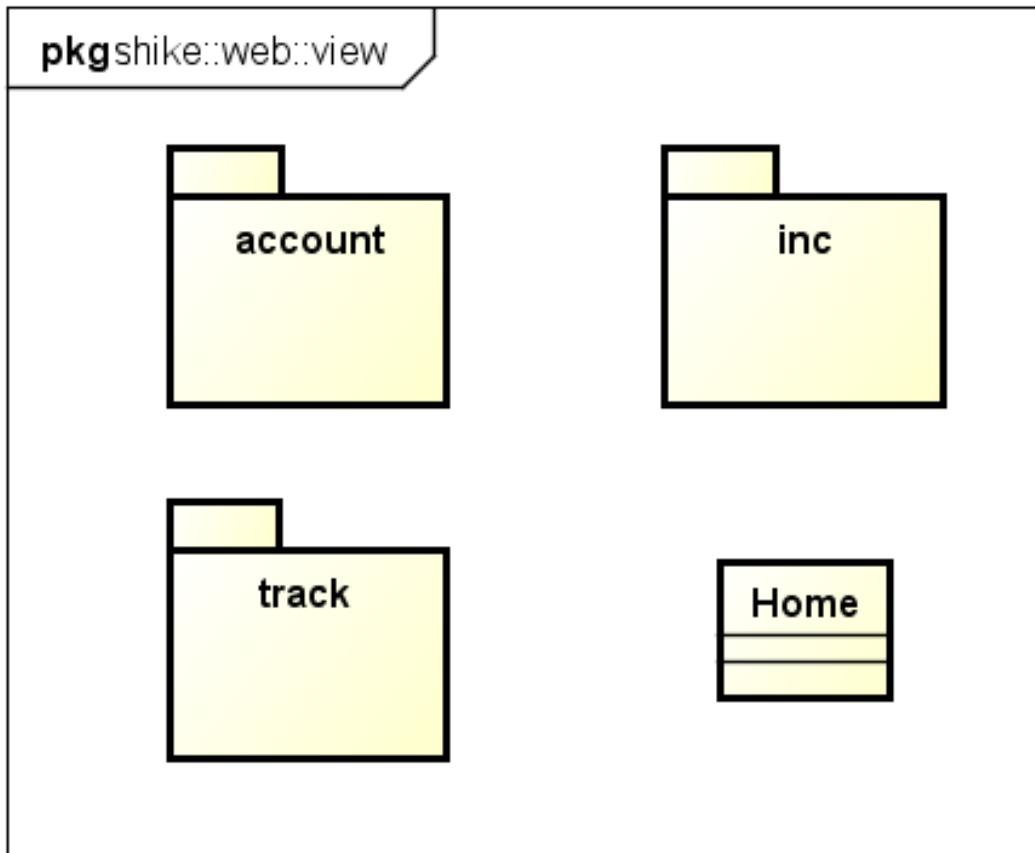


Figura 131: Diagramma di shike::web::view

- **Descrizione:** componente *View* dell'architettura MVC della parte web. Essa gestisce la visualizzazione dei dati delle pagine web all'interno della piattaforma. I file contenuti in essa non sono propriamente della classi ma quanto più delle pagine JSP che ricevono i dati dal *Controller*, il quale a sua volta li legge dal *Model*.
- **Componenti contenute**
  - shike::web::view::account
  - shike::web::view::track
  - shike::web::view::inc
- **Componente padre:** shike::web
- **Interazioni con altri componenti**
  - shike::web::model
  - shike::web::controller
  - shike::web::view::inc

### 3.32.2 Classi

#### 3.32.2.1 shike::web::view::Home



Figura 132: Diagramma di Home

- **Tipo:** concreta
- **Descrizione:** classe descrittiva del portale web, visualizzabile anche senza aver fatto il *login*.

### 3.33 shike::web::view::account

#### 3.33.1 Informazioni sul package

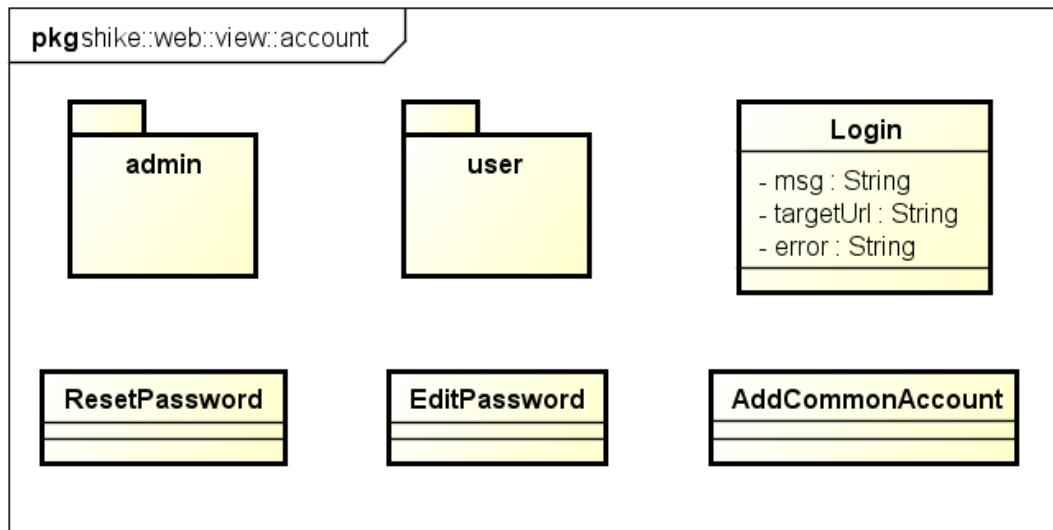


Figura 133: Diagramma di shike::web::view::account

- **Descrizione:** parte della *view* riguardante gli utenti, le classi al suo interno riguardano sia gli amministratori che gli utenti semplici.
- **Componenti contenute**
  - shike::web::view::account::user
  - shike::web::view::account::admin
- **Componente padre:** shike::web::view
- **Interazioni con altri componenti**

- shike::web::controller
- shike::web::view::inc

### 3.33.2 Classi

#### 3.33.2.1 shike::web::view::account::ResetPassword

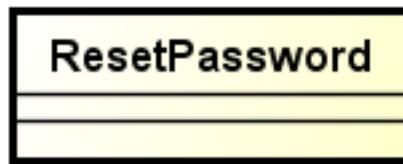


Figura 134: Diagramma di ResetPassword

- **Tipo:** concreta
- **Descrizione:** View che permette l'inserimento della propria *email* alla quale verrà inviata la nuova *password*.

#### 3.33.2.2 shike::web::view::account::Login

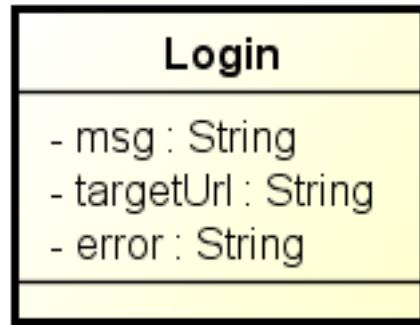


Figura 135: Diagramma di Login

- **Tipo:** concreta
- **Descrizione:** pagina che permette l'accesso al portale. Vengono visualizzati anche messaggi di errore o successo riguardanti la procedura di *login*.
- **Attributi:**
  - `-targetUrl : String`  
Contiene l' *url* alla quale si viene reindirizzate successivamente al *login*.
  - `-msg : String`  
Visualizza i messaggi di successo riguardanti il *login*.
  - `-error : String`  
Visualizza i messaggi di errore riguardanti il *login*.

### 3.33.2.3 shike::web::view::account::AddCommonAccount

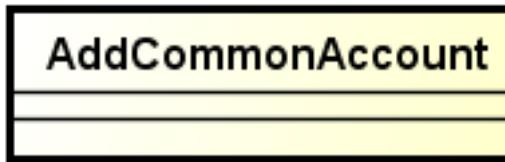


Figura 136: Diagramma di AddCommonAccount

- **Tipo:** concreta
- **Descrizione:** *View* che permette la registrazione di un nuovo utente.

### 3.33.2.4 shike::web::view::account::EditPassword

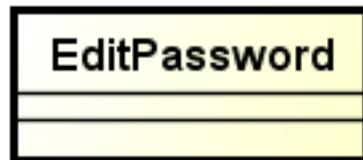


Figura 137: Diagramma di EditPassword

- **Tipo:** concreta
- **Descrizione:** permette di cambiare la *password*. La *view* è la stessa sia per amministratori che per utenti.

### 3.34 shike::web::view::account::user

#### 3.34.1 Informazioni sul package

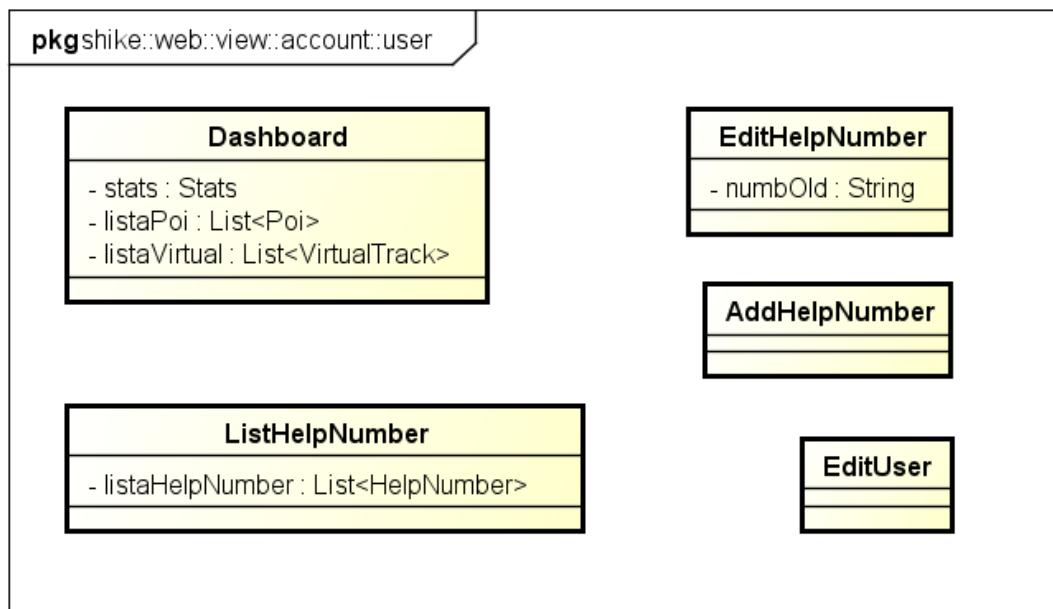


Figura 138: Diagramma di shike::web::view::account::user

- **Descrizione:** componente che contiene le pagine JSP che permettono di interagire con i dati dell' *account* utente.
- **Componente padre:** shike::web::view::account
- **Interazioni con altri componenti**
  - shike::web::controller
  - shike::web::view::inc

#### 3.34.2 Classi

##### 3.34.2.1 shike::web::view::account::user::Dashboard

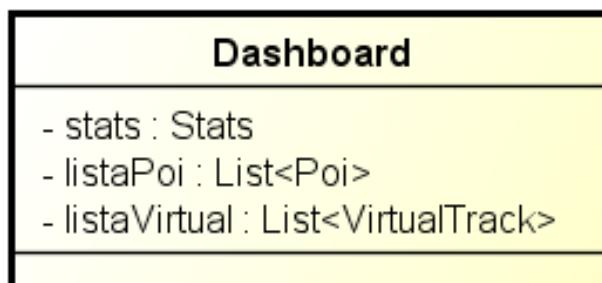


Figura 139: Diagramma di Dashboard

- **Tipo:** concreta
- **Descrizione:** *Dashboard* di riepilogo alla quale si viene reindirizzati dopo aver fatto il *login* se si è utenti semplici. Da qui è possibile decidere che azione compiere grazie alla barra di navigazione.
- **Attributi:**
  - `_listaVirtual : List<VirtualTrack>`  
Lista degli ultimi tracciati condivisi.
  - `_stats : Stats`  
Statistiche globali dell'utente.
  - `_listaPoi : List<Poi>`  
Lista degli ultimi POI aggiunti.

### 3.34.2.2 shike::web::view::account::user::AddHelpNumber

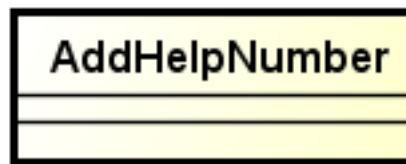


Figura 140: Diagramma di AddHelpNumber

- **Tipo:** concreta
- **Descrizione:** form che permette l'inserimento di un nuovo numero utile.

### 3.34.2.3 shike::web::view::account::user::EditHelpNumber

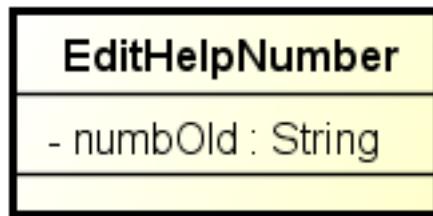


Figura 141: Diagramma di EditHelpNumber

- **Tipo:** concreta
- **Descrizione:** permette di modificare un numero utile aggiunto in precedenza.
- **Attributi:**
  - `_numbOld : String`  
Numero di telefono che si sta modificando.

### 3.34.2.4 shike::web::view::account::user::EditUser



Figura 142: Diagramma di EditUser

- **Tipo:** concreta
- **Descrizione:** View che permette la modifica dei dati del proprio profilo. Disponibile solo per gli utenti.

### 3.34.2.5 shike::web::view::account::user::ListHelpNumbers



Figura 143: Diagramma di ListHelpNumbers

- **Tipo:** concreta
- **Descrizione:** lista dei numeri utili associati al proprio profilo.
- **Attributi:**
  - `-listaHelpNum : List<HelpNumber>`  
Lista dei numeri utili del proprio profilo.

### 3.35 shike::web::view::account::admin

#### 3.35.1 Informazioni sul package

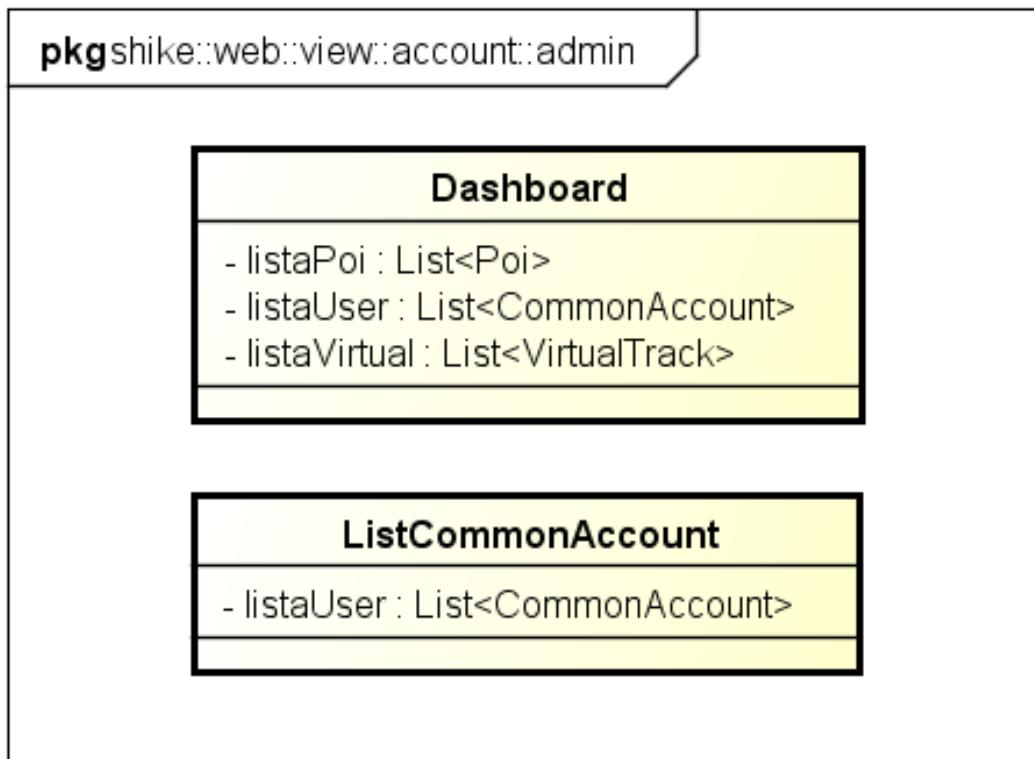


Figura 144: Diagramma di shike::web::view::account::admin

- **Descrizione:** componente che contiene la *view* della sezione del sito accessibile solo dall'amministratore.
- **Componente padre:** shike::web::view::account
- **Interazioni con altri componenti**
  - shike::web::controller
  - shike::web::view::inc

### 3.35.2 Classi

#### 3.35.2.1 shike::web::view::account::admin::Dashboard

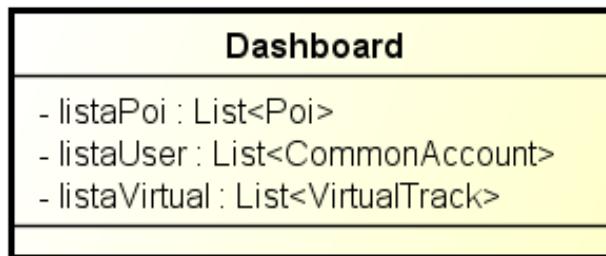


Figura 145: Diagramma di Dashboard

- **Tipo:** concreta
- **Descrizione:** *Dashboard* di riepilogo alla quale si viene reindirizzati dopo aver fatto il *login* se si è amministratori. Da qui è possibile decidere che azione compiere grazie alla barra di navigazione.
- **Attributi:**
  - `-listaPoi : List<Poi>`  
Lista degli ultimi POI aggiunti.
  - `-listaUser : List<CommonAccount>`  
Lista degli ultimi utenti iscritti.
  - `-listaVirtual : List<VirtualTrack>`  
Lista degli ultimi tracciati condivisi.

#### 3.35.2.2 shike::web::view::account::admin::ListCommonAccount



Figura 146: Diagramma di ListCommonAccount

- **Tipo:** concreta
- **Descrizione:** viene visualizzata la lista degli utenti registrati al portale permettendo all'amministratore di eliminare gli *account*.
- **Attributi:**
  - `+listaUser : List<CommonAccount>`  
Lista degli utenti registrati.

### 3.36 shike::web::view::track

#### 3.36.1 Informazioni sul package

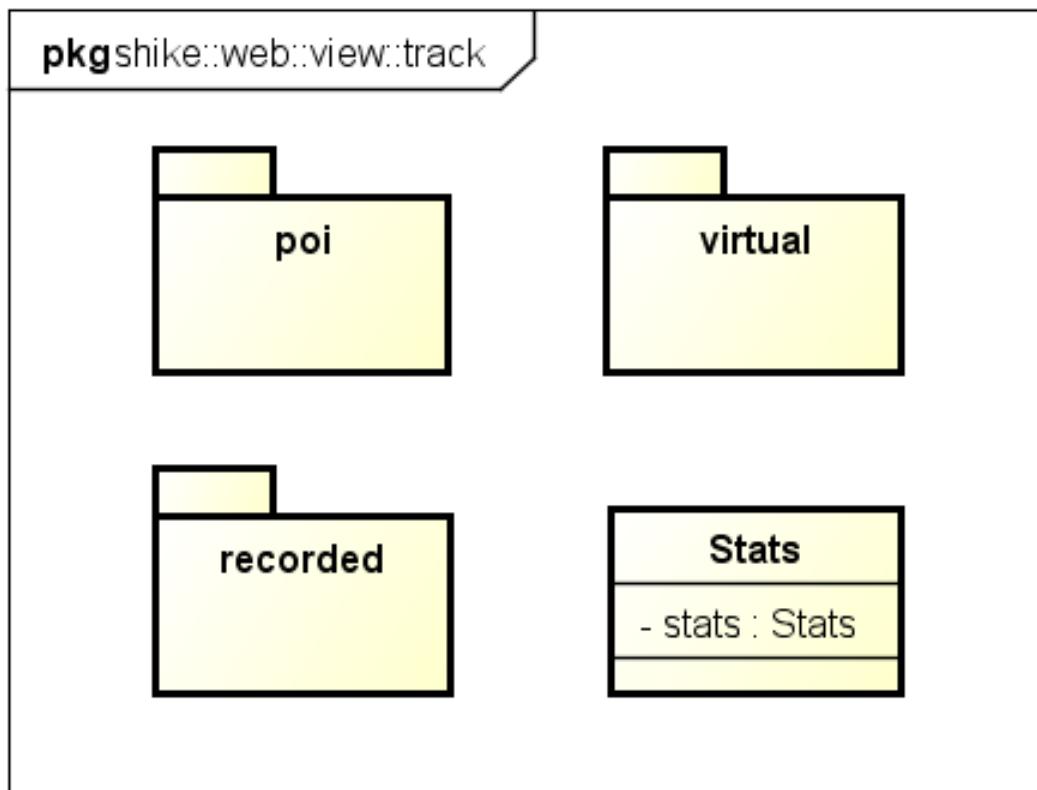


Figura 147: Diagramma di shike::web::view::track

- **Descrizione:** componente che contiene i file JSP della *view* del portale web.
- **Componenti contenute**
  - shike::web::view::track::poi
  - shike::web::view::track::virtual
  - shike::web::view::track::recorded
- **Componente padre:** shike::web::view
- **Interazioni con altri componenti**
  - shike::web::controller
  - shike::web::view::inc

### 3.36.2 Classi

#### 3.36.2.1 shike::web::view::track::Stats



Figura 148: Diagramma di Stats

- **Tipo:** concreta
- **Descrizione:** schermata di riepilogo contenente le statistiche globali dell'utente.
- **Attributi:**
  - `-stats : Stats`  
Statistiche globali dell'utente.

### 3.37 shike::web::view::track::poi

#### 3.37.1 Informazioni sul package

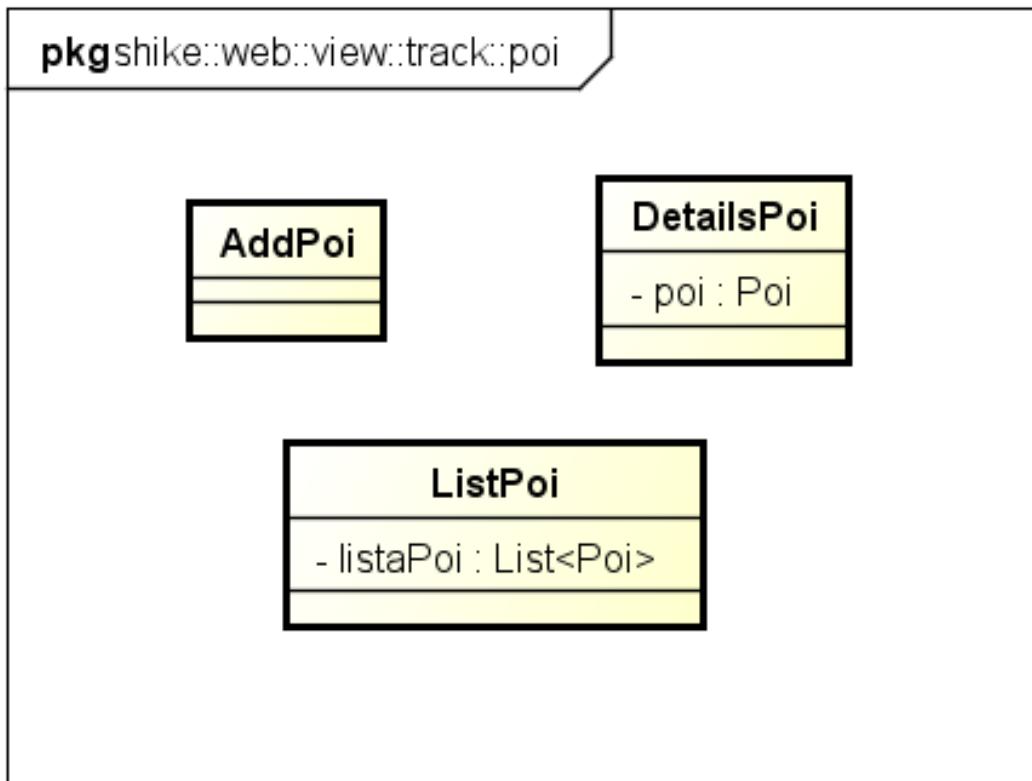


Figura 149: Diagramma di shike::web::view::track::poi

- **Descrizione:** componente che contiene i file JSP della *view* del portale web riguardanti i POI.
- **Componente padre:** shike::web::view::track

#### 3.37.2 Classi

##### 3.37.2.1 shike::web::view::track::poi::DetailsPoi

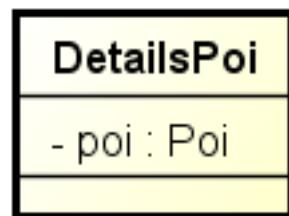


Figura 150: Diagramma di DetailsPoi

- **Tipo:** concreta
- **Descrizione:** permette di visualizzare i dettagli di un POI.
- **Attributi:**
  - -poi : Poi  
POI del quale si desidera visualizzare i dettagli.

### 3.37.2.2 shike::web::view::track::poi::AddPoi



Figura 151: Diagramma di AddPoi

- **Tipo:** concreta
- **Descrizione:** permette di aggiungere un POI nel sistema.

### 3.37.2.3 shike::web::view::track::poi::ListPoi



Figura 152: Diagramma di ListPoi

- **Tipo:** concreta
- **Descrizione:** vista generica dei POI che visualizza l'elenco di tutti i POI presenti nel *database*.
- **Attributi:**
  - -listaPoi : List<Poi>  
Lista dei POI presenti.

### 3.38 shike::web::view::track::virtual

#### 3.38.1 Informazioni sul package

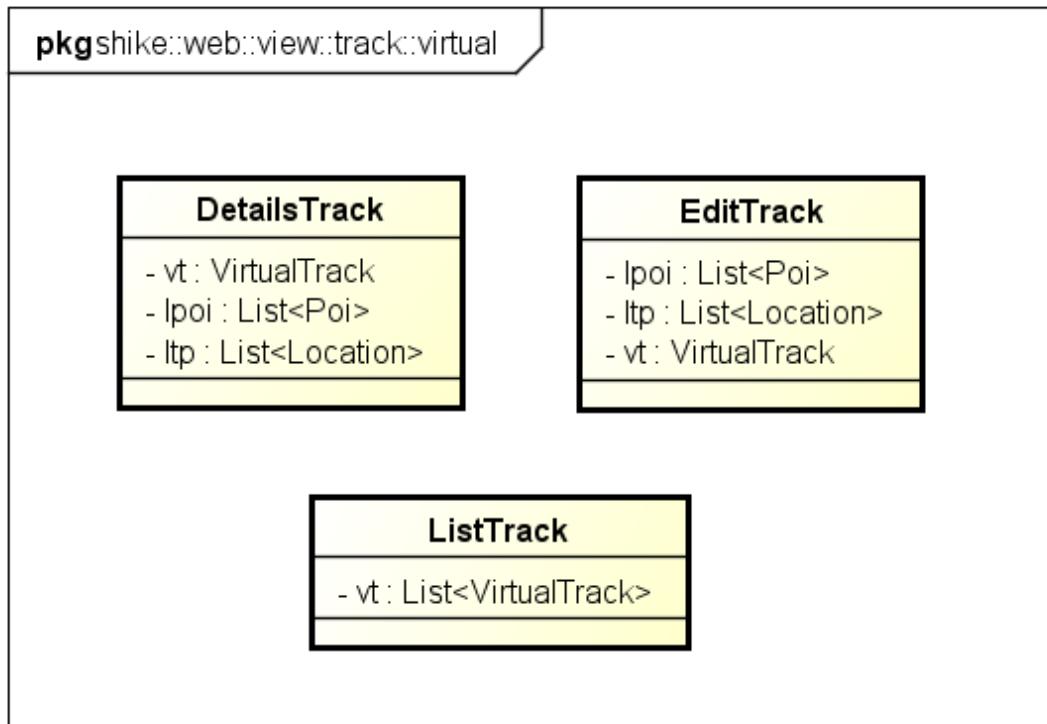


Figura 153: Diagramma di shike::web::view::track::virtual

- **Descrizione:** componente che contiene i file JSP della *view* del portale web riguardanti i tracciati condivisi.
- **Componente padre:** shike::web::view::track

#### 3.38.2 Classi

##### 3.38.2.1 shike::web::view::track::virtual::ListTrack



Figura 154: Diagramma di ListTrack

- **Tipo:** concreta

- **Descrizione:** visualizza la lista dei tracciati presenti.

- **Metodi:**

- `-vt() : List<VirtualTrack>`  
Lista dei tracciati disponibili.

### 3.38.2.2 shike::web::view::track::virtual::EditTrack

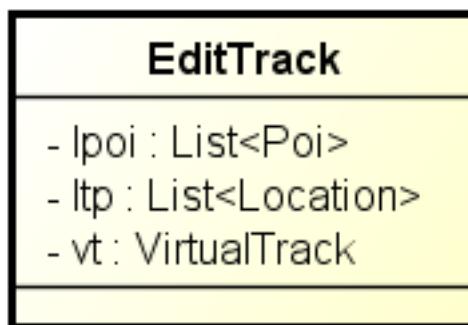


Figura 155: Diagramma di EditTrack

- **Tipo:** concreta

- **Descrizione:** permette di modificare il nome di un tracciato condiviso.

- **Attributi:**

- `-vt : VirtualTrack`  
VirtualTrack da visualizzare.
- `-lpoi : List<Poi>`  
Lista dei POI.
- `-ltp : List<Location>`  
Lista dei punti di Location.

### 3.38.2.3 shike::web::view::track::virtual::DetailsTrack

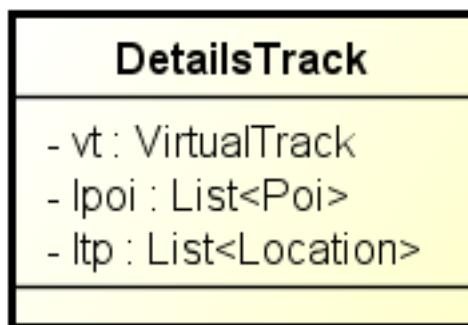


Figura 156: Diagramma di DetailsTrack

- **Tipo:** concreta

- **Descrizione:** permette di visualizzare in dettaglio i dati di un tracciato.

- **Attributi:**

- `-vt : VirtualTrack`  
VirtualTrack da visualizzare.
- `-lpoi : List<Poi>`  
Lista dei POI.
- `-ltp : List<Location>`  
Lista dei punti di Location.

### 3.39 shike::web::view::track::recorded

#### 3.39.1 Informazioni sul package

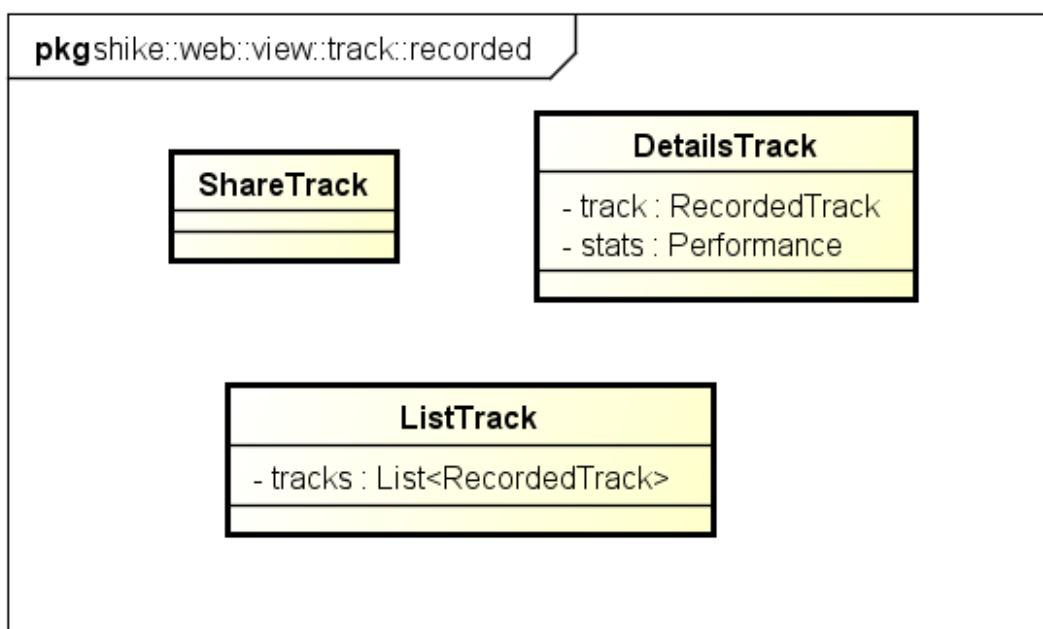


Figura 157: Diagramma di shike::web::view::track::recorded

- **Descrizione:** componente che contiene i file JSP della *view* del portale web dei tracciati registrati.
- **Componente padre:** shike::web::view::track

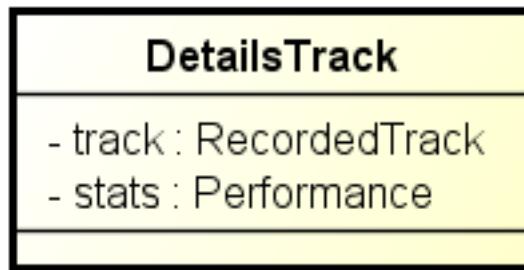
**3.39.2 Classi****3.39.2.1 shike::web::view::track::recorded::DetailsTrack**

Figura 158: Diagramma di DetailsTrack

- **Tipo:** concreta
- **Descrizione:** permette di visualizzare in dettaglio i dati di un tracciato.
- **Attributi:**
  - **–track :** RecordedTrack  
Tracciato che si vuole visualizzare.
  - **–stats :** Performance  
*performance* riguardanti il tracciato visualizzato.

**3.39.2.2 shike::web::view::track::recorded::ShareTrack**

Figura 159: Diagramma di ShareTrack

- **Tipo:** concreta
- **Descrizione:** vista dei tracciati condivisi.

### 3.39.2.3 shike::web::view::track::recorded::ListTrack



Figura 160: Diagramma di ListTrack

- **Tipo:** concreta
- **Descrizione:** visualizza la lista dei tracciati presenti.
- **Attributi:**
  - `-tracks : List<RecordedTrack>`  
Lista dei tracciati.

### 3.40 shike::web::view::inc

#### 3.40.1 Informazioni sul package

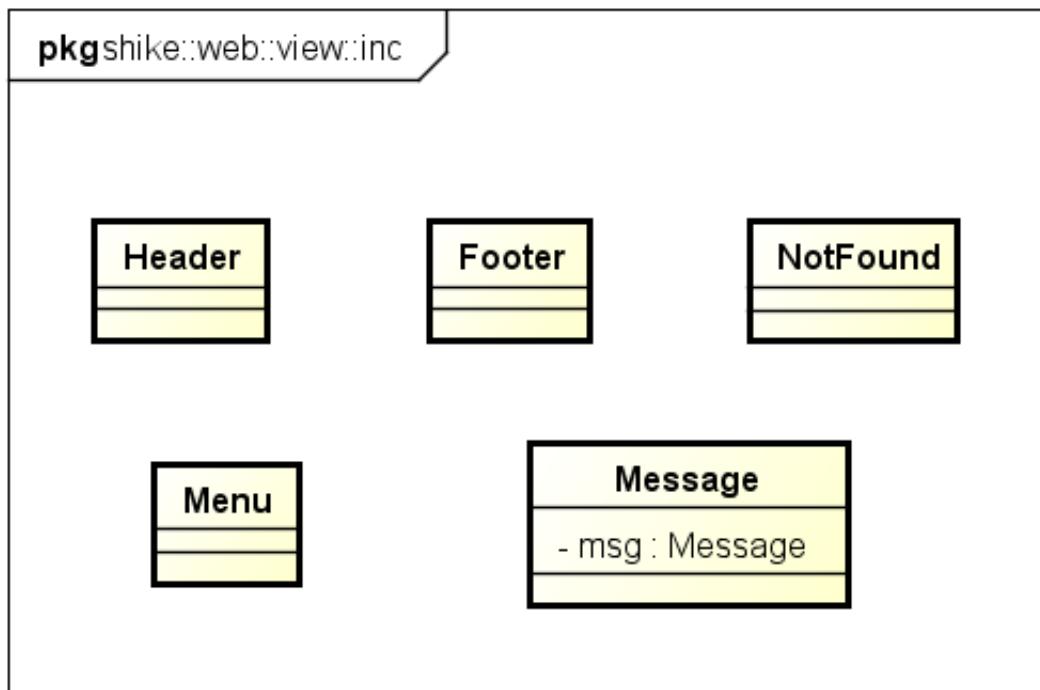


Figura 161: Diagramma di shike::web::view::inc

- **Descrizione:** componente che contiene le parti JSP del portale web di supporto alle altre pagine.

- **Componente padre:** shike::web::view

### 3.40.2 Classi

#### 3.40.2.1 shike::web::view::inc::Footer



Figura 162: Diagramma di Footer

- **Tipo:** concreta
- **Descrizione:** parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

#### 3.40.2.2 shike::web::view::inc::NotFound



Figura 163: Diagramma di NotFound

- **Tipo:** concreta
- **Descrizione:** *Redirect* verso l'indirizzo contenente la *view* della pagina 404.

#### 3.40.2.3 shike::web::view::inc::Header

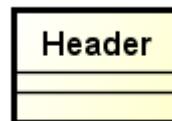


Figura 164: Diagramma di Header

- **Tipo:** concreta
- **Descrizione:** pagina JSP contenente l'header delle pagine web.

#### 3.40.2.4 shike::web::view::inc::Menu

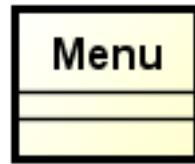


Figura 165: Diagramma di Menu

- **Tipo:** concreta
- **Descrizione:** pagina JSP contenente il menu delle pagine web.

#### 3.40.2.5 shike::web::view::inc::Message



Figura 166: Diagramma di Message

- **Tipo:** concreta
- **Descrizione:** View generica che permette la visualizzazione di un messaggio.
- **Attributi:**
  - `-msg : Message`  
*Model* dati di un messaggio.

### 3.41 shike::web::helper

#### 3.41.1 Informazioni sul package

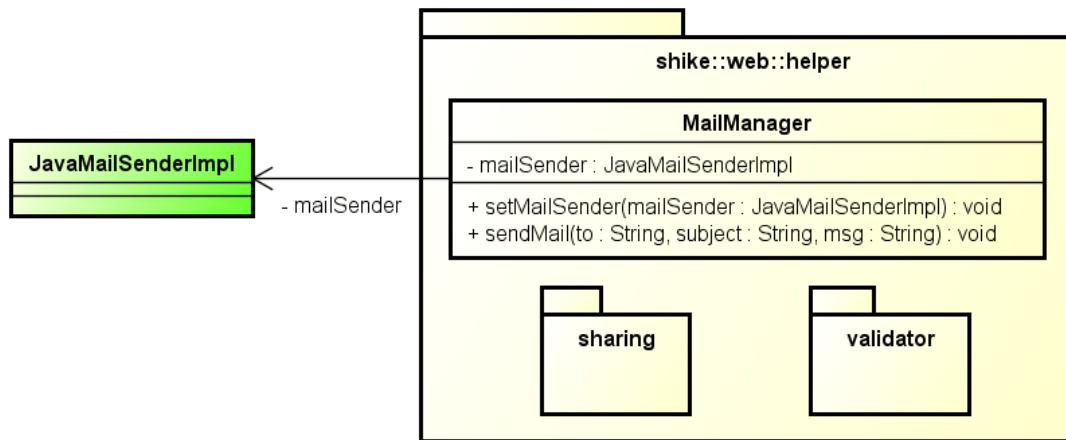


Figura 167: Diagramma di `shike::web::helper`

- **Descrizione:** componente che raccoglie le componenti di supporto all'applicazione web.
- **Componenti contenute**
  - `shike::web::helper::sharing`
  - `shike::web::helper::validator`
- **Componente padre:** `shike::web`
- **Interazioni con altri componenti**
  - `shike::web::controller`

#### 3.41.2 Classi

##### 3.41.2.1 `shike::web::helper::MailManager`

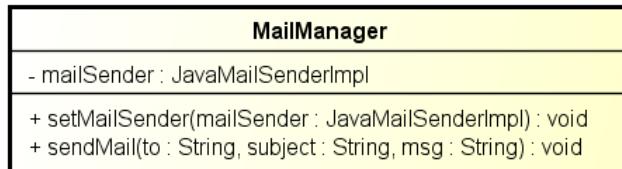


Figura 168: Diagramma di `MailManager`

- **Tipo:** concreta
- **Descrizione:** classe di supporto che permette l'invio di *email*.
- **Attributi:**

- `mailSender : JavaMailSenderImpl`  
Oggetto che permette l'invio dell'email.

- **Metodi:**

- `+sendMail( to : String, subject : String, msg : String ) : void`  
Invia l'email con i parametri selezionati.

**Argomenti:**

- \* `to` : destinatario dell'email.
- \* `subject` : oggetto dell'email.
- \* `msg` : messaggio da inviare.

- `+setMailSender() : void`  
Permette di impostare il `mailSender`.

### 3.42 shike::web::helper::sharing

#### 3.42.1 Informazioni sul package

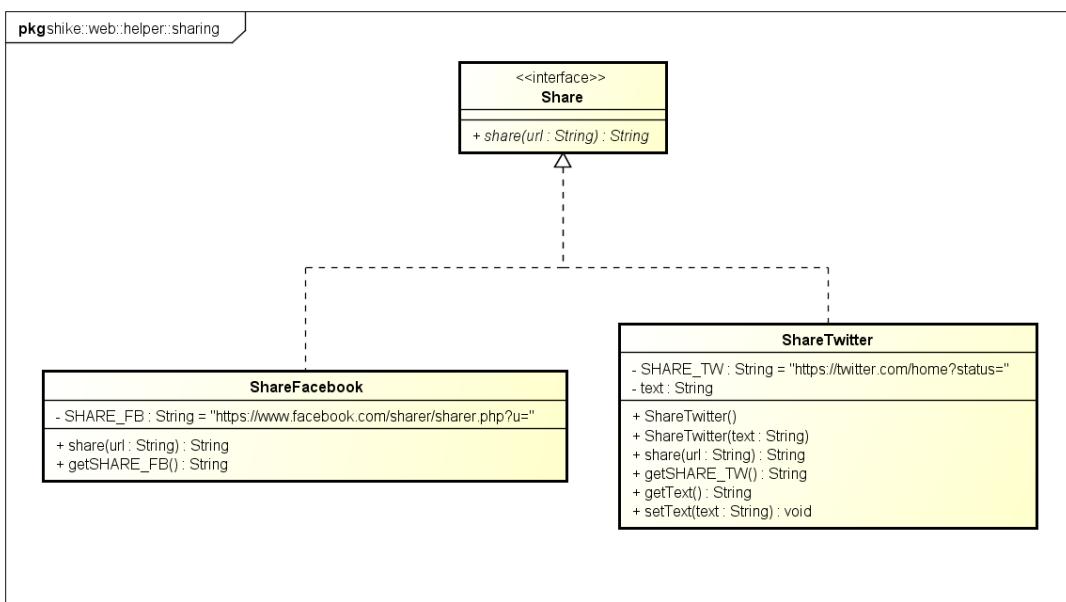


Figura 169: Diagramma di shike::web::helper::sharing

- **Descrizione:** componente che raccoglie le classi di supporto all'applicazione web per la condivisione delle informazioni sui *Social network*.
- **Componente padre:** shike::web::helper
- **Interazioni con altri componenti**
  - shike::web::controller

### 3.42.2 Classi

#### 3.42.2.1 shike::web::helper::sharing::Share

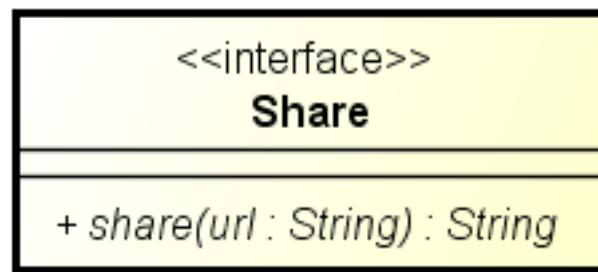


Figura 170: Diagramma di Share

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che implementa il *pattern Strategy* relativo alla funzionalità di condivisione di una *performance* e del relativo tracciato sui *social network* (Facebook e Twitter).
- **Implementata da:**
  - shike::web::helper::sharing::ShareFacebook;
  - shike::web::helper::sharing::ShareTwitter;
- **Metodi:**
  - **+share( url : String ) : String**  
Effettua la condivisione sui *social* di un link.  
**Argomenti:**  
\* **url** : link da condividere.

#### 3.42.2.2 shike::web::helper::sharing::ShareFacebook



Figura 171: Diagramma di ShareFacebook

- **Tipo:** concreta
- **Descrizione:** classe che modella la funzionalità di condivisione di una *performance* e del relativo tracciato sul *social network Facebook*.
- **Implementa:**
  - shike::web::helper::sharing::Share;
- **Attributi:**

- `SHARE_FB : String`

Costante contenente il link per poter condividere su Facebook.

- **Metodi:**

- `+getSHARE_FB() : String`

Restituisce la variabile SHARE\_FB.

### 3.42.2.3 shike::web::helper::sharing::ShareTwitter

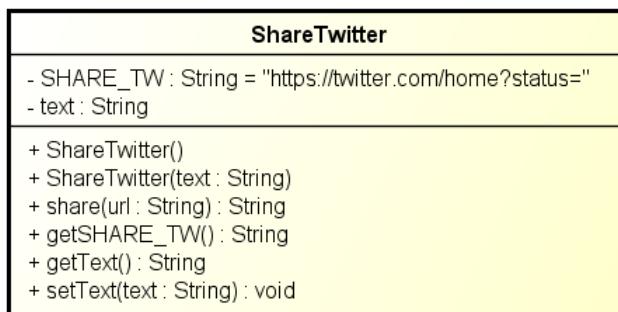


Figura 172: Diagramma di ShareTwitter

- **Tipo:** concreta

- **Descrizione:** classe che modella la funzionalità di condivisione di una *performance* e del relativo tracciato sul *social network Twitter*.

- **Implementa:**

- shike::web::helper::sharing::**Share**:

- **Attributi:**

- `text : String`

Descrizione contenuta nel tweet.

- `SHARE_TW : String`

Costante contenente il link per poter condividere su Twitter.

- **Metodi:**

- `+getText() : String`

Restituisce la variabile text.

- `+getSHARE_TW() : String`

Restituisce la variabile SHARE\_TW.

- `+setSHARE_TW( SHARE_TW : String ) : void`

**Argomenti:**

\* `SHARE_TW` : valore da impostare nella variabile SHARE\_TW.

- `+ShareTwitter( ) :`

Costruttore ridefinito, il messaggio del tweet viene inizializzato di default.

- `+ShareTwitter( texr : String ) :`

Costruttore ad un parametro, permette di assegnare un messaggio personalizzato al tweet.

**Argomenti:**

\* `texr` : testo da impostare.

### 3.43 shike::web::helper::validator

#### 3.43.1 Informazioni sul package

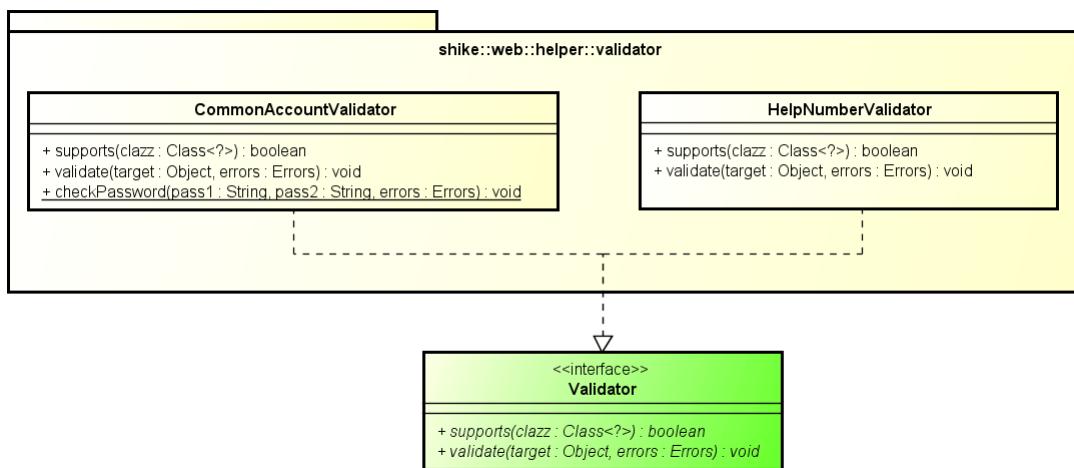


Figura 173: Diagramma di shike::web::helper::validator

- **Descrizione:** il *package* contiene tutte le classi necessarie per la validazione dei *form* presenti nel sito, come ad esempio la registrazione di un nuovo utente.
- **Componente padre:** shike::web::helper
- **Interazioni con altri componenti**
  - shike::web::model::user
  - shike::web::model::session::track

#### 3.43.2 Classi

##### 3.43.2.1 shike::web::helper::validator::HelpNumberValidator

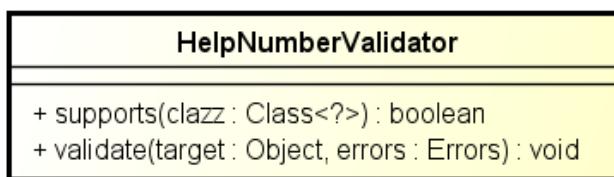


Figura 174: Diagramma di HelpNumberValidator

- **Tipo:** concreta
- **Descrizione:** classe che permette di validare i dati che vengono inseriti nel *form* per l'aggiunta di un numero utile. Vengono effettuate delle chiamate ai metodi della classe statica *ValidationUtils* e in alcuni casi vengono effettuati dei controlli appositi.
- **Implementa:**
  - org.springframework.validation.Validator

- **Metodi:**

- `+supports( clazz : Class<?> ) : boolean`

Viene verificato che la classe passata come parametro sia assegnabile.

**Argomenti:**

- `* clazz` : classe da controllare.

- `+validate( target : Object, errors : Errors ) : void`

Valida l'oggetto che viene passato come parametro, se si verificano errori questi vengono salvati su `errors`.

**Argomenti:**

- `* target` : oggetto da validare, successivamente viene eseguito un cast per ottenere l'oggetto apposito.

- `* errors` : dove salvare gli errori se presenti.

### 3.43.2.2 shike::web::helper::validator::CommonAccountValidator

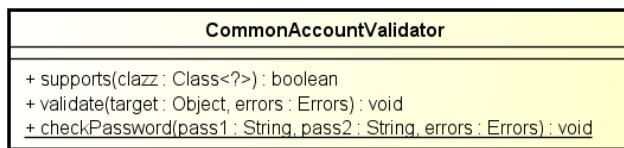


Figura 175: Diagramma di CommonAccountValidator

- **Tipo:** concreta

- **Descrizione:** classe che permette di validare i dati che vengono inseriti nel *form* di registrazione. Vengono effettuate delle chiamate ai metodi della classe statica *ValidationUtils* e in alcuni casi vengono effettuati dei controlli appositi.

- **Implementa:**

- `org.springframework.validation.Validator`

- **Metodi:**

- `+supports( clazz : Class<?> ) : boolean`

Viene verificato che la classe passata come parametro sia assegnabile.

**Argomenti:**

- `* clazz` : classe da controllare.

- `+validate( target : Object, errors : Errors ) : void`

Valida l'oggetto che viene passato come parametro, se si verificano errori questi vengono salvati su `errors`.

**Argomenti:**

- `* target` : oggetto da validare, successivamente viene eseguito un cast per ottenere l'oggetto apposito.

- `* errors` : dove salvare gli errori se presenti.

- `+checkPassword( pass1 : String, pass2 : String, errors : Errors ) : void`

Permette di controllare la validità di una `password`.

**Argomenti:**

- `* pass1` : password 1 da controllare.

- `* pass2` : password 2 da controllare.

- `* errors` : se presenti errori, vengono salvati qui.

### 3.44 shike::web::model

#### 3.44.1 Informazioni sul package

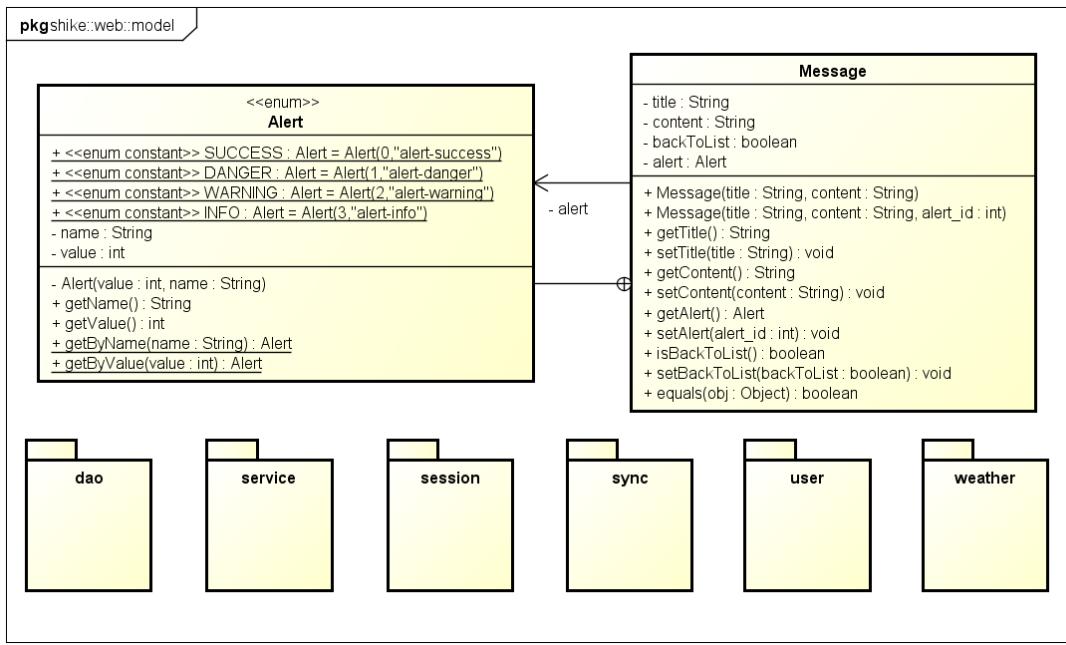


Figura 176: Diagramma di shike::web::model

- **Descrizione:** componente *Model* dell'architettura MVC della parte web. Essa memorizza tutti i dati dell'utente su cui la *View* si basa.
- **Componenti contenute**
  - shike::web::model::weather
  - shike::web::model::user
  - shike::web::model::session
  - shike::web::model::dao
  - shike::web::model::service
  - shike::web::model::sync
- **Componente padre:** shike::web
- **Interazioni con altri componenti**
  - shike::web::view

### 3.44.2 Classi

#### 3.44.2.1 shike::web::model::Message::Alert

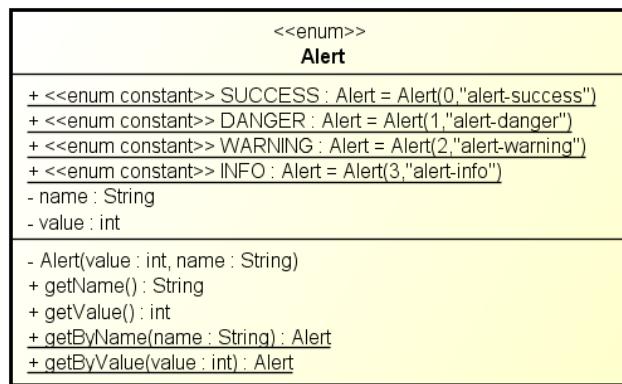


Figura 177: Diagramma di Message::Alert

- **Tipo:** enum
- **Descrizione:** enumeratore interno alla classe shike::web::model::Message contenente le varie tipologie di *alert* disponibili.
- **Attributi:**
  - **`value`** : int  
Identificatore univoco della tipologia di *alert*.
  - **`name`** : String  
Nome della tipologia di *alert*.
- **Metodi:**
  - **`+getValue()`** : int  
Restituisce la variabile `value`.
  - **`+getName()`** : String  
Restituisce la variabile `name`.
  - **`+getById( id : int )`** : Alert  
Ritorna il tipo di *alert* con il nome inserito, o *null* se tale tipologia non esiste. Se ci sono più tipologie con lo stesso nome, viene ritornata la prima dichiarata.  
**Argomenti:**
    - \* **`id`** : id del tipo di *alert* desiderato.
  - **`+getByName( name : String )`** : Alert  
Ritorna il tipo di *alert* con l'id inserito, o *null* se tale tipologia non esiste.  
**Argomenti:**
    - \* **`name`** : nome della tipologia di *alert* desiderata.

### 3.44.2.2 shike::web::model::Message

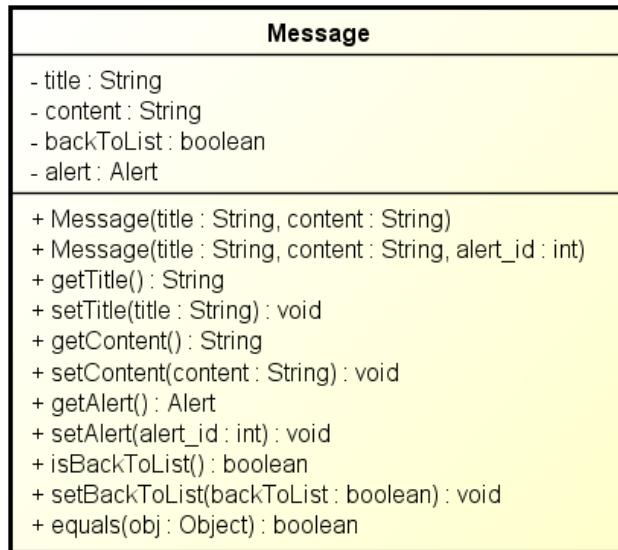


Figura 178: Diagramma di Message

- **Tipo:** concreta
- **Descrizione:** classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
- **Attributi:**

- **`backToList` : boolean**  
Boleano che permette di cambiare il significato del tasto “Torna Indietro” del messaggio.
- **`title` : String**  
Il titolo del messaggio.
- **`content` : String**  
Il contenuto del messaggio.
- **`alert` : Alert**  
Classe che si vuole far adottare al box del message.

- **Metodi:**

- **`isbackToList()` : boolean**  
Restituisce la variabile `backToList`.
- **`setBackToList( backToList : boolean )` : void**  
**Argomenti:**  
\* `backToList` : valore da impostare nella variabile `backToList`.
- **`getTitle()` : String**  
Restituisce la variabile `title`.
- **`setTitle( title : String )` : void**  
**Argomenti:**

- \* **title** : valore da impostare nella variabile **title**.
- **+getContent()** : **String**  
Restituisce la variabile **content**.
- **+setContent( content : String )** : **void**  
**Argomenti:**
  - \* **content** : valore da impostare nella variabile **content**.
- **+getAlert()** : **Alert**  
Restituisce la variabile **alert**.
- **+setAlert( alert : Alert )** : **void**  
**Argomenti:**
  - \* **alert** : valore da impostare nella variabile **alert**.
- **+Message( title : String, content : String )** :  
Costruttore della classe a due parametri.  
**Argomenti:**
  - \* **title** : titolo del messaggio.
  - \* **content** : contenuto del messaggio.
- **+Message( title : String, content : String, alert\_id : int )** :  
Costruttore della classe a tre parametri.  
**Argomenti:**
  - \* **title** : titolo del messaggio.
  - \* **content** : contenuto del messaggio.
  - \* **alert\_id** : tipo di errore.
- **+equals( o : Object )** : **boolean**  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
  - \* **o** : oggetto da confrontare con **this**.

### 3.45 shike::web::model::weather

#### 3.45.1 Informazioni sul package

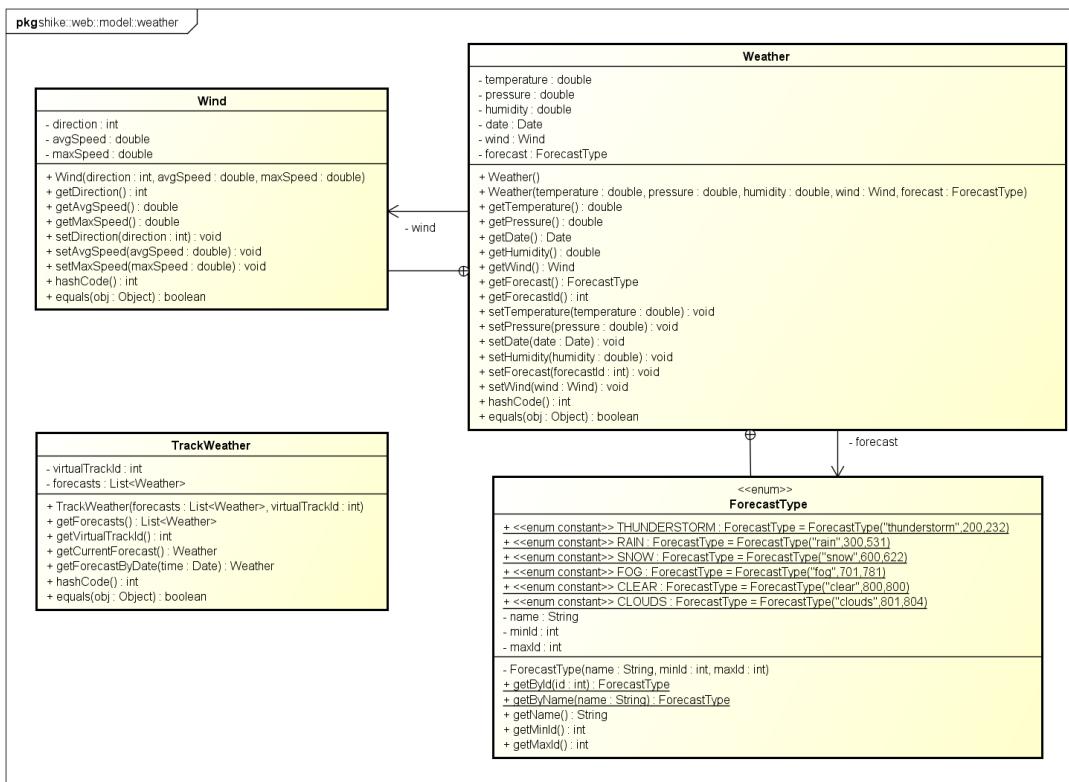


Figura 179: Diagramma di shike::web::model::weather

- **Descrizione:** componente del *Model* utilizzata per modellare i dati delle condizioni climatiche riguardanti i percorsi caricati all'interno della piattaforma Web.
- **Componente padre:** shike::web::model
- **Interazioni con altri componenti**
  - shike::web::model::session::track

### 3.45.2 Classi

#### 3.45.2.1 shike::web::model::weather::TrackWeather

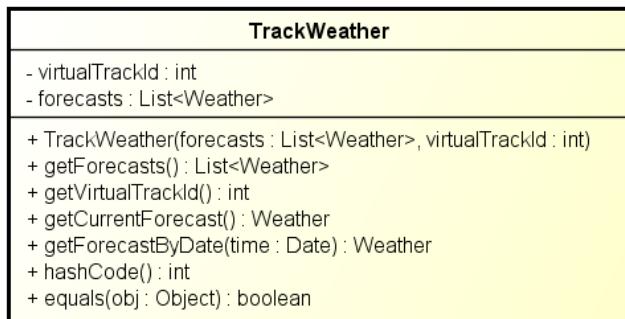


Figura 180: Diagramma di TrackWeather

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni relative alle previsioni meteo previste nel percorso. Contiene un riferimento alla classe di Android Location (android.location.Location), che indica il luogo per cui la previsione meteo è valida.
- **Attributi:**
  - `-forecast : List<Weather>`  
Lista delle previsioni corrispondente ad un tracciato.
  - `-virtualTrackId : int`  
Id del percorso associato alle previsioni.
- **Metodi:**
  - `+getForecast() : List<Weather>`  
Restituisce la variabile `forecast`.
  - `+getVirtualTrackId() : int`  
Restituisce la variabile `virtualTrackId`.
  - `+getCurrentForecast( ) : Weather`  
Previsione per la data odierna.
  - `+getForecastByDate( date : Date ) : Weather`  
Previsioni per una certa data passata come parametro.

**Argomenti:**

  - \* `date` : data per la quale si vogliono le previsioni.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.
- Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+hashCode( ) : int`  
*Override* che genera un codice `hash` considerando tutti gli attributi della classe.
- `+TrackWeather( virtualTrackId : int, forecasts : List<Weather> ) :`  
Costruttore della classe.
- Argomenti:**

- \* **virtualTrackId** : id del tracciato al quale si riferiscono le previsioni.
- \* **forecasts** : lista delle previsioni.

### 3.45.2.2 shike::web::model::weather::Weather

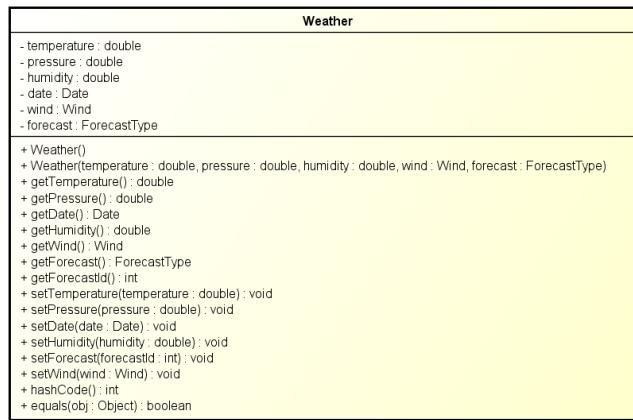


Figura 181: Diagramma di Weather

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni sulle condizione meteo possibili.
- **Attributi:**
  - **forecast** : **ForecastType**  
Id della previsione meteo.
  - **wind** : **Wind**  
Vento della previsione meteo.
  - **temperature** : **double**  
Temperatura della previsione meteo.
  - **pressure** : **double**  
Pressione della previsione meteo.
  - **humidity** : **double**  
Umidità della previsione meteo.
  - **date** : **Date**  
Data della previsione meteo.
- **Metodi:**
  - **+getForecast()** : **ForecastType**  
Restituisce la variabile **forecast**.
  - **+setForecast( forecast : ForecastType )** : **void**  
**Argomenti:**
    - \* **forecast** : valore da impostare nella variabile **forecast**.
  - **+getWind()** : **Wind**  
Restituisce la variabile **wind**.
  - **+setWind( wind : Wind )** : **void**  
**Argomenti:**

- \* `wind` : valore da impostare nella variabile `wind`.
- `+getTemperature() : double`  
Restituisce la variabile `temperature`.
- `+setTemperature( temperature : double ) : void`  
**Argomenti:**
  - \* `temperature` : valore da impostare nella variabile `temperature`.
- `+getPressure() : double`  
Restituisce la variabile `pressure`.
- `+setPressure( pressure : double ) : void`  
**Argomenti:**
  - \* `pressure` : valore da impostare nella variabile `pressure`.
- `+getHumidity() : double`  
Restituisce la variabile `humidity`.
- `+setHumidity( humidity : double ) : void`  
**Argomenti:**
  - \* `humidity` : valore da impostare nella variabile `humidity`.
- `+getDate() : Date`  
Restituisce la variabile `date`.
- `+ setDate( date : Date ) : void`  
**Argomenti:**
  - \* `date` : valore da impostare nella variabile `date`.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.
- `+Weather( ) :`  
Costruttore della classe.
- `+Weather( temperature : double, pressure : double, humidity : double, wind : Wind, forecast : ForecastType ) :`  
Costruttore della classe a 5 parametri.  
**Argomenti:**
  - \* `temperature` : temperatura della previsione.
  - \* `pressure` : pressione della previsione.
  - \* `humidity` : umidità della previsione.
  - \* `wind` : vento della previsione.
  - \* `forecast` : tipo del meteo della previsione.

### 3.45.2.3 shike::web::model::weather::Weather::Wind

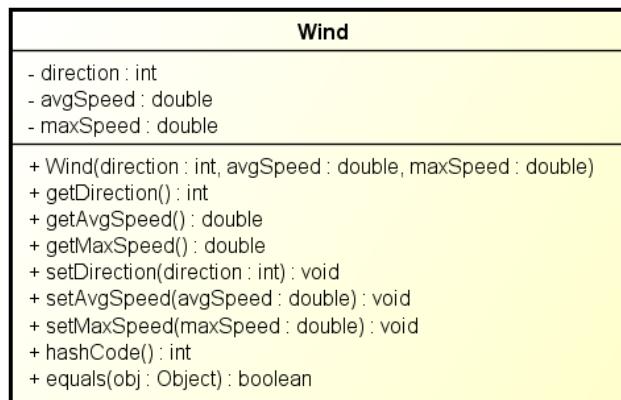


Figura 182: Diagramma di Weather::Wind

- **Tipo:** concreta
- **Descrizione:** classe interna di Weather che rappresenta il vento.
- **Attributi:**
  - `–direction : int`  
Direzione del vento.
  - `–avgSpeed : double`  
Velocità media del vento.
  - `–maxSpeed : double`  
Velocità massima del vento.
- **Metodi:**
  - `+getDirection() : int`  
Restituisce la variabile `direction`.
  - `+setDirection( direction : int ) : void`  
**Argomenti:**
    - \* `direction` : valore da impostare nella variabile `direction`.
  - `+getAvgSpeed() : double`  
Restituisce la variabile `avgSpeed`.
  - `+setAvgSpeed( avgSpeed : double ) : void`  
**Argomenti:**
    - \* `avgSpeed` : valore da impostare nella variabile `avgSpeed`.
  - `+getMaxSpeed() : double`  
Restituisce la variabile `maxSpeed`.
  - `+setMaxSpeed( maxSpeed : double ) : void`  
**Argomenti:**
    - \* `maxSpeed` : valore da impostare nella variabile `maxSpeed`.
  - `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**

- \* `o` : oggetto da confrontare con `this`.
- `+hashcode() : int`  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe.
- `+Wind( direction : int, avgSpeed : double, maxSpeed : double ) :`  
Costruttore della classe.  
**Argomenti:**
  - \* `direction` : direzione del vento.
  - \* `avgSpeed` : velocità media del vento.
  - \* `maxSpeed` : velocità massima del vento.

### 3.45.2.4 shike::web::model::weather::Weather::ForecastType

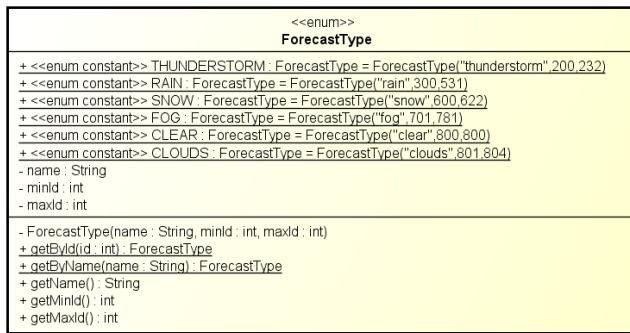


Figura 183: Diagramma di Weather::ForecastType

- **Tipo:** enum
- **Descrizione:** enum che indica il tipo di condizioni meteo previste. I codici indicati nelle condizioni si riferiscono agli identificatori di <http://openweathermap.org/weather-conditions>
- **Attributi:**
  - `-maxId : int`  
Limite superiore (incluso) dei numeri che identificano la condizione meteo.
  - `-minId : int`  
Limite inferiore (incluso) dei numeri che identificano la condizione meteo.
  - `-name : String`  
Nome della condizione meteo.
- **Metodi:**
  - `+getMaxId() : int`  
Restituisce la variabile `maxId`.
  - `+getMinId() : int`  
Restituisce la variabile `minId`.
  - `+getName() : String`  
Restituisce la variabile `name`.

- `+ForecastType( name : String, minId : int, maxId : int ) :`  
Ritorna la prima condizione meteo che contiene nel suo intervallo degli id accettabili quello indicato.

**Argomenti:**

- \* `name` : nome della condizione meteo.
- \* `minId` : id minimo della condizione.
- \* `maxId` : id massimo della condizione.

- `+getByName( name : String ) : ForecastType`  
Ritorna la prima condizione meteo che ha il nome uguale a quello passato per parametro.

**Argomenti:**

- \* `name` : nome della condizione da cercare.

- `+getId( id : int ) : ForecastType`  
Ritorna la prima condizione meteo che contiene nel suo intervallo degli id accettabili quello indicato.

**Argomenti:**

- \* `id` : id della condizione meteo ricercata.

### 3.46 shike::web::model::user

#### 3.46.1 Informazioni sul package

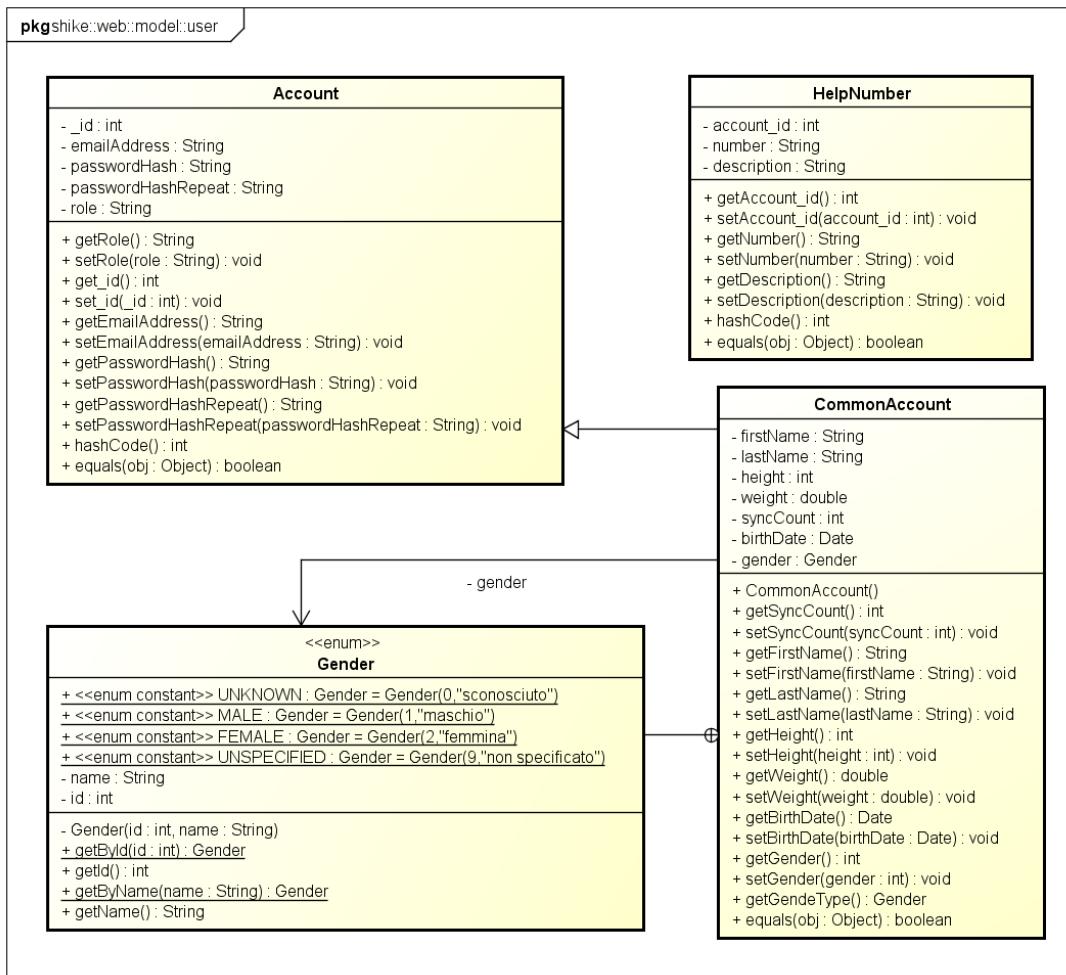


Figura 184: Diagramma di shike::web::model::user

- **Descrizione:** componente del *Model* per la raccolta delle informazioni degli utenti registrati all'interno della piattaforma Web.
- **Componente padre:** `shike::web::model`

### 3.46.2 Classi

#### 3.46.2.1 shike::web::model::user::Account

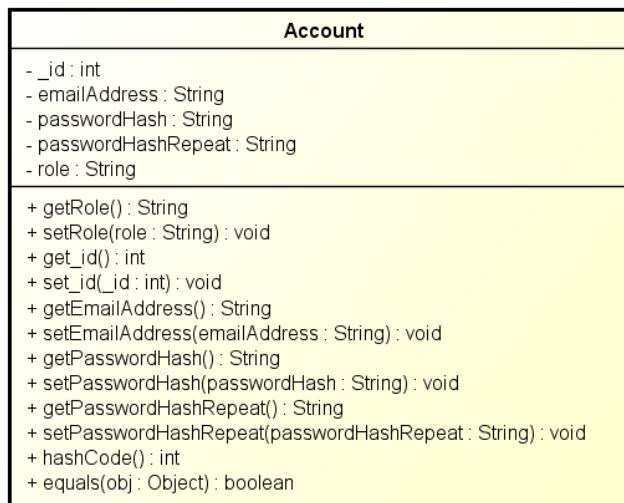


Figura 185: Diagramma di Account

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un utente comune, tale classe è identica alla controparte lato applicazione.
- **Sottoclassi:**
  - shike::web::model::user::CommonAccount
- **Attributi:**
  - `_id` : int  
Id dell'utente.
  - `emailAddress` : String  
Indirizzo *email* dell'utente.
  - `passwordHash` : String  
*Password* dell'utente.
  - `passwordHashRepeat` : String  
Ripetizione della *password* dell'utente. Il campo è usato durante la registrazione.
  - `role` : String  
Ruolo dell'utente, se amministratore o utente semplice.
- **Metodi:**
  - `+get_id()` : int  
Restituisce la variabile `_id`.
  - `+set_id( _id : int )` : void  
**Argomenti:**
    - \* `_id` : valore da impostare nella variabile `_id`.

- `+getEmailAddress() : String`  
Restituisce la variabile `emailAddress`.
- `+setEmailAddress( emailAddress : String ) : void`  
**Argomenti:**
  - \* `emailAddress` : valore da impostare nella variabile `emailAddress`.
- `+getPasswordHash() : String`  
Restituisce la variabile `passwordHash`.
- `+setPasswordHash( passwordHash : String ) : void`  
**Argomenti:**
  - \* `passwordHash` : valore da impostare nella variabile `passwordHash`.
- `+getPasswordHashRepeat() : String`  
Restituisce la variabile `passwordHashRepeat`.
- `+setPasswordHashRepeat( passwordHashRepeat : String ) : void`  
**Argomenti:**
  - \* `passwordHashRepeat` : valore da impostare nella variabile `passwordHashRepeat`.
- `+getRole() : String`  
Restituisce la variabile `role`.
- `+setRole( role : String ) : void`  
**Argomenti:**
  - \* `role` : valore da impostare nella variabile `role`.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+hashcode( ) : int`  
*Override* che genera un codice `hash` considerando tutti gli attributi della classe.

### 3.46.2.2 shike::web::model::user::CommonAccount

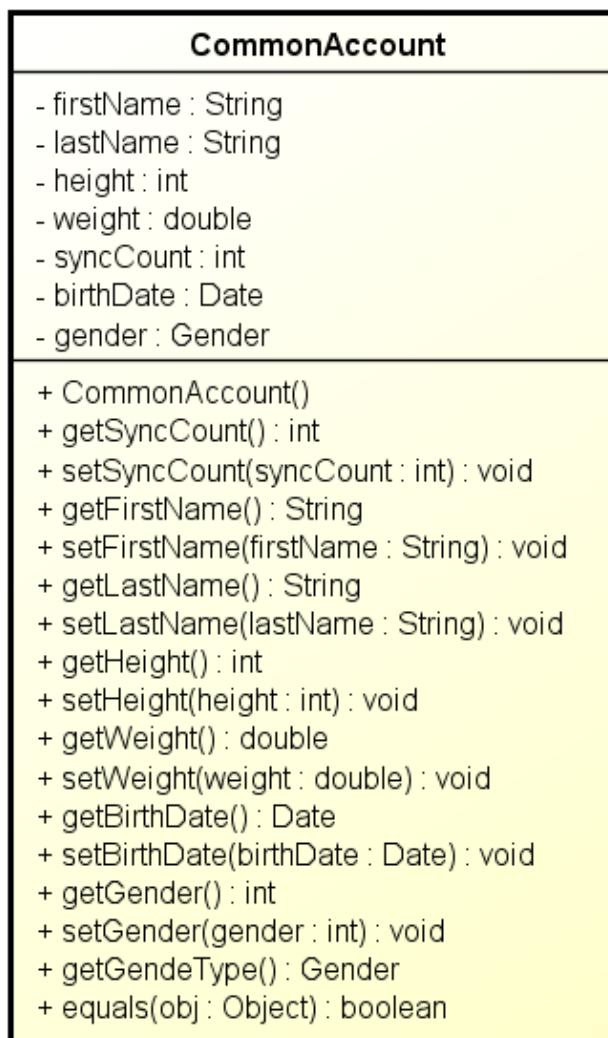


Figura 186: Diagramma di CommonAccount

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.
- **Superclassi:**
  - shike::web::model::user::Account
- **Attributi:**
  - **`firstName`** : String  
Nome dell'utente.
  - **`lastName`** : String  
Cognome dell'utente.

- `height : int`  
Altezza dell'utente.
- `syncCount : int`  
Sincronizzazioni effettuate dell'utente.
- `weight : float`  
Peso dell'utente.
- `birthDate : Date`  
Data di nascita dell'utente.
- `gender : Gender`  
Genere dell'utente.

- **Metodi:**

- `+getFirstName() : String`  
Restituisce la variabile `firstName`.  
**Argomenti:**  
\* `firstName` : valore da impostare nella variabile `firstName`.
- `+setFirstName( firstName : String ) : void`  
**Argomenti:**  
\* `lastName` : valore da impostare nella variabile `lastName`.
- `+getLastname() : String`  
Restituisce la variabile `lastName`.
- `+setLastName( lastName : String ) : void`  
**Argomenti:**  
\* `lastName` : valore da impostare nella variabile `lastName`.
- `+getHeight() : int`  
Restituisce la variabile `height`.
- `+setHeight( height : int ) : void`  
**Argomenti:**  
\* `height` : valore da impostare nella variabile `height`.
- `+getSyncCount() : int`  
Restituisce la variabile `syncCount`.
- `+setSyncCount( syncCount : int ) : void`  
**Argomenti:**  
\* `syncCount` : valore da impostare nella variabile `syncCount`.
- `+getWeight() : float`  
Restituisce la variabile `weight`.
- `+setWeight( weight : float ) : void`  
**Argomenti:**  
\* `weight` : valore da impostare nella variabile `weight`.
- `+getBirthDate() : Date`  
Restituisce la variabile `birthDate`.
- `+setBirthDate( birthDate : Date ) : void`  
**Argomenti:**  
\* `birthDate` : valore da impostare nella variabile `birthDate`.
- `+getGender() : Gender`  
Restituisce la variabile `gender`.
- `+setGender( gender : Gender ) : void`  
**Argomenti:**

\* `gender` : valore da impostare nella variabile `gender`.

- `+CommonAccount()` :  
Costruttore della classe.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**  
\* `o` : oggetto da confrontare con `this`.

### 3.46.2.3 shike::web::model::user::HelpNumber

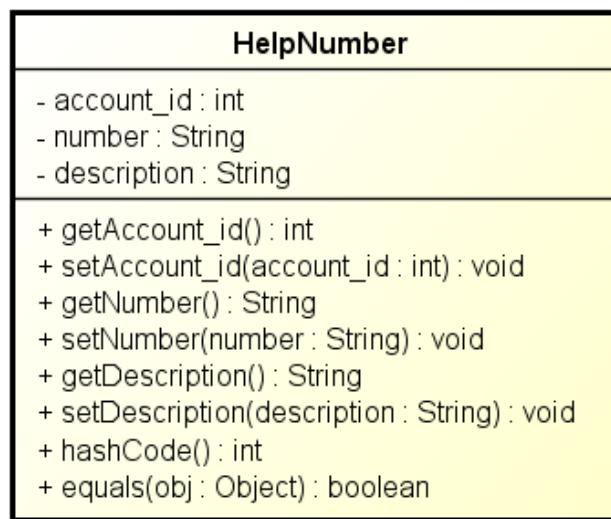


Figura 187: Diagramma di HelpNumber

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.
- **Attributi:**
  - `-account_id : int`  
Id dell'utente al quale appartiene il numero.
  - `-number : String`  
Numero utile.
  - `-description : String`  
Descrizione del numero utile.
- **Metodi:**
  - `+getAccount_id() : int`  
Restituisce la variabile `account_id`.
  - `+setAccount_id( account_id : int ) : void`  
**Argomenti:**  
\* `account_id` : valore da impostare nella variabile `account_id`.

- **+getNumber() : String**  
Restituisce la variabile `number`.
- **+setNumber( number : String ) : void**  
**Argomenti:**
  - \* `number` : valore da impostare nella variabile `number`.
- **+getDescription() : String**  
Restituisce la variabile `description`.
- **+setDescription( description : String ) : void**  
**Argomenti:**
  - \* `description` : valore da impostare nella variabile `description`.
- **+equals( o : Object ) : boolean**  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.
- **+hashcode( ) : int**  
*Override* che genera un codice *hash* considerando tutti gli attributi della classe.

#### 3.46.2.4 shike::web::model::user::CommonAccount::Gender

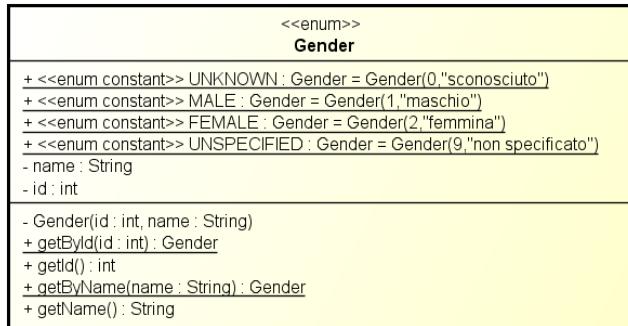


Figura 188: Diagramma di CommonAccount::Gender

- **Tipo:** enum
- **Descrizione:** enum che indica i possibili sessi indicati dallo standard ISO 5218
- **Attributi:**
  - **-id : int**  
Identificatore del sesso (in accordo con ISO 5218).
  - **-name : String**  
Nome del sesso.
- **Metodi:**
  - **+getId() : int**  
Restituisce la variabile `id`.
  - **+getName() : String**  
Restituisce la variabile `name`.

- `+Gender( id : int, name : String ) :`  
Costruttore dell'enum.  
**Argomenti:**
  - \* `id` : id del sesso.
  - \* `name` : nome del sesso.
- `+getById( id : int ) : Gender`  
Ritorna il sesso con l'id cercato.  
**Argomenti:**
  - \* `id` : id del sesso.
- `+getByName( name : String ) : Gender`  
Ritorna il sesso con il nome cercato.  
**Argomenti:**
  - \* `name` : nome del sesso.

### 3.47 shike::web::model::session

#### 3.47.1 Informazioni sul package

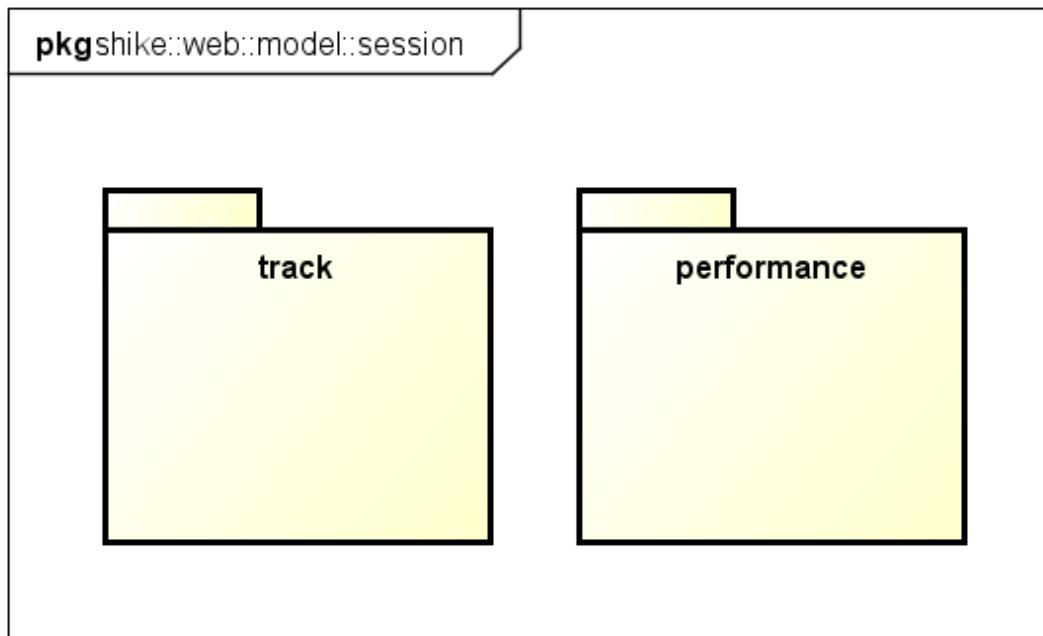


Figura 189: Diagramma di shike::web::model::session

- **Descrizione:** componente del *Model* che raccoglie le informazioni delle sessioni effettuate dagli utenti.
- **Componenti contenute**
  - shike::web::model::session::track
  - shike::web::model::session::performance
- **Componente padre:** shike::web::model

- Interazioni con altri componenti
  - shike::web::model::weather

### 3.48 shike::web::model::session::track

#### 3.48.1 Informazioni sul package

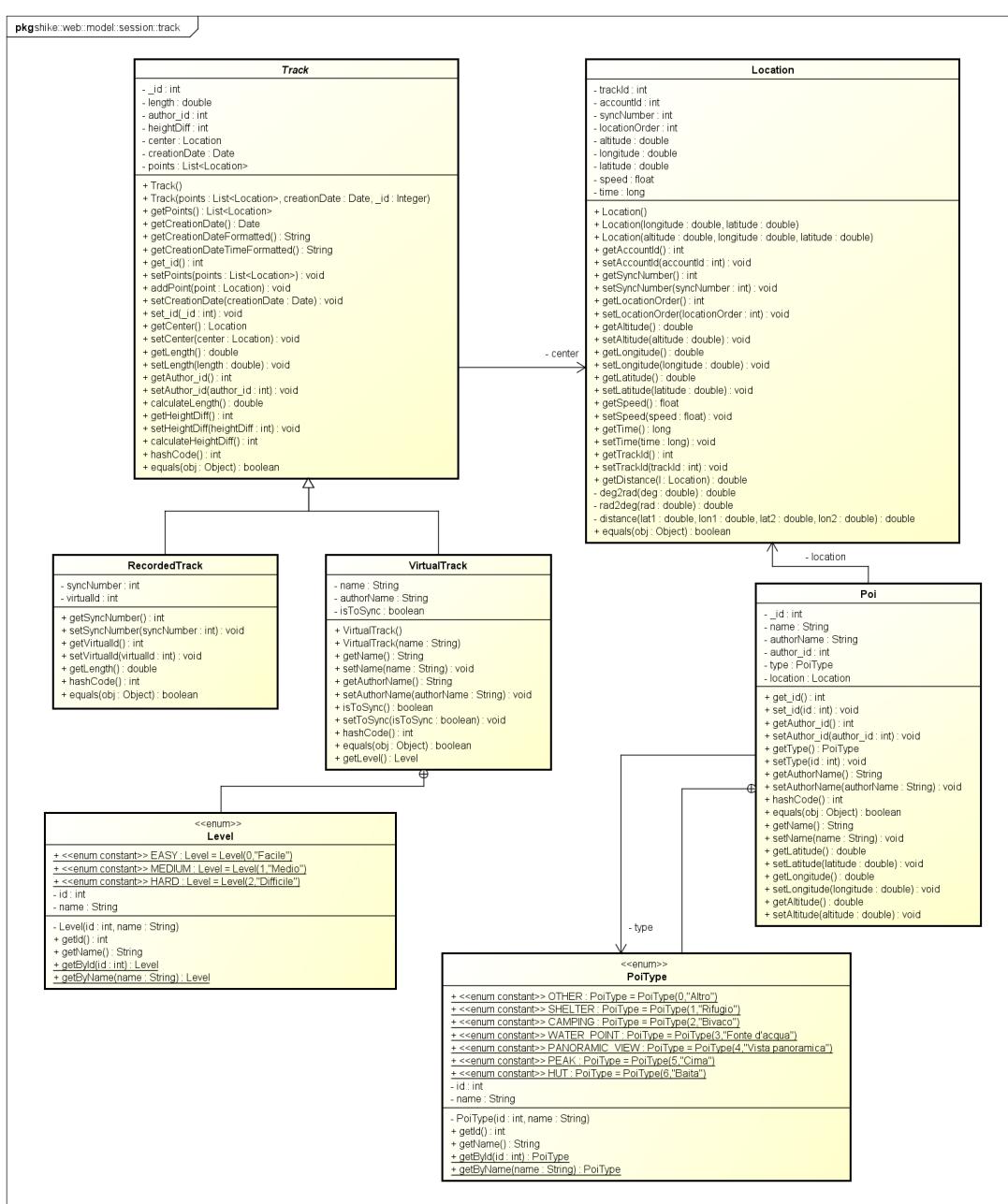


Figura 190: Diagramma di shike::web::model::session::track

- **Descrizione:** componente del *Model* che raccoglie le informazioni dei tracciati sia generati dagli utenti durante le uscite, che quelli condivisi per essere scaricati nel *device*.
- **Componente padre:** shike::web::model::session
- **Interazioni con altri componenti**
  - shike::web::model::user

### 3.48.2 Classi

#### 3.48.2.1 shike::web::model::session::track::Track

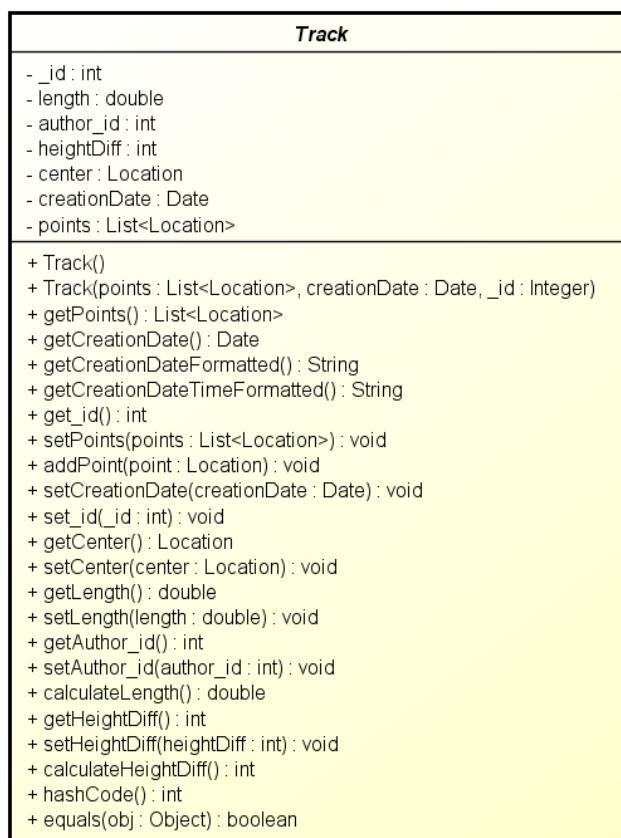


Figura 191: Diagramma di Track

- **Tipo:** astratta
- **Descrizione:** interfaccia che modella la struttura dei percorsi.
- **Sottoclassi:**
  - shike::web::model::session::track::RecordedTrack
  - shike::web::model::session::track::VirtualTrack
- **Attributi:**

- `-points : List<Location>`  
I punti di cui è composto il percorso.
- `-creationDate : Date`  
Data di creazione del percorso.
- `-_id : int`  
Identificatore del percorso.
- `-center : Location`  
Centro della mappa in base al percorso.
- `-length : double`  
Lunghezza del percorso.
- `-author_id : int`  
Autore del tracciato.
- `-heightDiff : int`  
Dislivello del tracciato.
- `-heightDiff : int`  
Dislivello del tracciato.
- `-heightDiff : int`  
Dislivello del tracciato.

- **Metodi:**

- `+getPoints() : List<Location>`  
Restituisce la variabile `points`.  
**Argomenti:**  
\* `points` : valore da impostare nella variabile `points`.
- `+setPoints( points : List<Location> ) : void`  
**Argomenti:**  
\* `points` : valore da impostare nella variabile `points`.
- `+getCreationDate() : Date`  
Restituisce la variabile `creationDate`.  
**Argomenti:**  
\* `creationDate` : valore da impostare nella variabile `creationDate`.
- `+setCreationDate( creationDate : Date ) : void`  
**Argomenti:**  
\* `creationDate` : valore da impostare nella variabile `creationDate`.
- `+get_id() : int`  
Restituisce la variabile `_id`.  
**Argomenti:**  
\* `_id` : valore da impostare nella variabile `_id`.
- `+set_id( _id : int ) : void`  
**Argomenti:**  
\* `_id` : valore da impostare nella variabile `_id`.
- `+getCenter() : Location`  
Restituisce la variabile `center`.  
**Argomenti:**  
\* `center` : valore da impostare nella variabile `center`.
- `+setCenter( center : Location ) : void`  
**Argomenti:**  
\* `center` : valore da impostare nella variabile `center`.
- `+getLength() : double`  
Restituisce la variabile `length`.  
**Argomenti:**  
\* `length` : valore da impostare nella variabile `length`.
- `+setLength( length : double ) : void`  
**Argomenti:**  
\* `length` : valore da impostare nella variabile `length`.

- `+getAuthor_id() : int`  
Restituisce la variabile `author_id`.
- `+setAuthor_id( author_id : int ) : void`  
**Argomenti:**
  - \* `author_id` : valore da impostare nella variabile `author_id`.
- `+getHeightDiff() : int`  
Restituisce la variabile `heightDiff`.
- `+setHeightDiff( heightDiff : int ) : void`  
**Argomenti:**
  - \* `heightDiff` : valore da impostare nella variabile `heightDiff`.
- `+getHeightDiff() : int`  
Restituisce la variabile `heightDiff`.
- `+setHeightDiff( heightDiff : int ) : void`  
**Argomenti:**
  - \* `heightDiff` : valore da impostare nella variabile `heightDiff`.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+Track( ) :`  
Costruttore della classe.
- `+hashcode( ) : int`  
*Override* che genera un codice `hash` considerando tutti gli attributi della classe.
- `+Track( points : List<Location> , creationDate : Date, _id : Integer ) :`  
Costruttore della classe.  
**Argomenti:**
  - \* `points` : lista dei punti.
  - \* `creationDate` : data del tracciato.
  - \* `_id` : id dell'utente.
- `+calculateHeighDiff( ) : int`  
Permette di calcolare il dislivello.
- `+calculateLength( ) : double`  
Permette di calcolare la lunghezza del tracciato.

### 3.48.2.2 shike::web::model::session::track::RecordedTrack

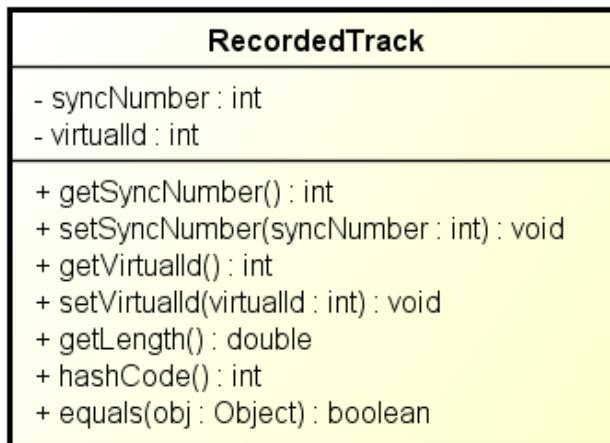


Figura 192: Diagramma di RecordedTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un percorso registrato dall'utente e carico sul web ma non condiviso.
- **Superclassi:**
  - shike::web::model::session::track::Track:
- **Attributi:**
  - **`syncNumber`** : int  
Numero sync del tracciato registrato.
  - **`virtualId`** : int  
Percorso virtuale corrispondente.
- **Metodi:**
  - **`+getSyncNumber()`** : int  
Restituisce la variabile syncNumber.
  - **`+setSyncNumber( syncNumber : int )`** : void  
**Argomenti:**
    - \* **`syncNumber`** : valore da impostare nella variabile syncNumber.
  - **`+getVirtualId()`** : int  
Restituisce la variabile virtualId.
  - **`+setVirtualId( virtualId : int )`** : void  
**Argomenti:**
    - \* **`virtualId`** : valore da impostare nella variabile virtualId.
  - **`+equals( o : Object )`** : boolean  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
    - \* **`o`** : oggetto da confrontare con `this`.
  - **`+hashCode()`** : int  
*Override* che genera un codice hash considerando tutti gli attributi della classe.

### 3.48.2.3 shike::web::model::session::track::VirtualTrack

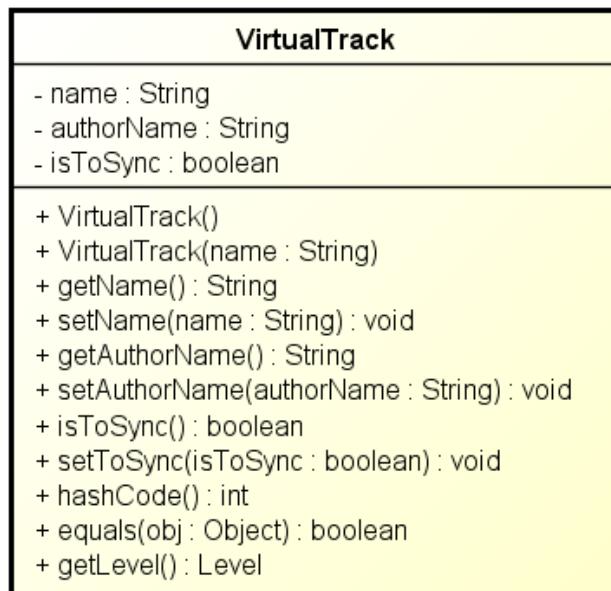


Figura 193: Diagramma di VirtualTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un percorso condiviso nel sito.
- **Superclassi:**
  - shike::web::model::session::track::Track
- **Attributi:**
  - **–authorName : String**  
Nome dell'autore del tracciato.
  - **–name : String**  
Nome del tracciato.
  - **–isToSync : boolean**  
Boleano che se true indica che il tracciato va sincronizzato.
- **Metodi:**
  - **+getAuthorName() : String**  
Restituisce la variabile authorName.
  - **+setAuthorName( authorName : String ) : void**  
**Argomenti:**
    - \* **authorName** : valore da impostare nella variabile authorName.
  - **+getName() : String**  
Restituisce la variabile name.
  - **+setName( name : String ) : void**  
**Argomenti:**
    - \* **name** : nome del tracciato.

- \* `name` : valore da impostare nella variabile `name`.
- `+isToSync() : boolean`  
Restituisce la variabile `isToSync`.
- `+setIsToSync( isToSync : boolean ) : void`  
**Argomenti:**
  - \* `isToSync` : valore da impostare nella variabile `isToSync`.
- `+VirtualTrack( name : String ) :`  
Costruttore ad un parametro.  
**Argomenti:**
  - \* `name` : nome del tracciato.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+hashcode( ) : int`  
*Override* che genera un codice `hash` considerando tutti gli attributi della classe.
- `+getLevel( ) : Level`  
Calcola il livello del percorso.
- `+VirtualTrack( name : String ) :`  
Costruttore della classe.  
**Argomenti:**
  - \* `name` : nome del percorso.

### 3.48.2.4 shike::web::model::session::track::Poi

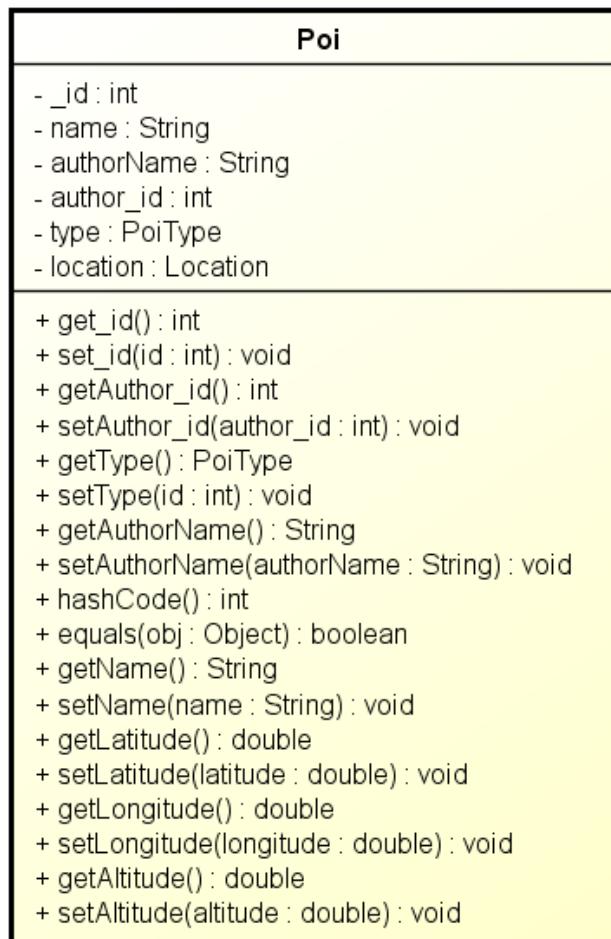


Figura 194: Diagramma di Poi

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un punto di interesse (POI).
- **Attributi:**
  - `_id` : int  
Id del POI.
  - `_author_id` : int  
Id dell'autore del POI.
  - `_type` : PoiType  
Tipo del POI.
  - `_name` : String  
Nome del POI.
  - `_location` : Location  
Posizione del POI.

- **`-authorName : String`**  
Nome completo del POI.

- **Metodi:**

- **`+get_id() : int`**  
Restituisce la variabile `_id`.

- **`+set_id( _id : int ) : void`**  
**Argomenti:**

\* `_id` : valore da impostare nella variabile `_id`.

- **`+getAuthor_id() : int`**  
Restituisce la variabile `author_id`.

- **`+setAuthor_id( author_id : int ) : void`**  
**Argomenti:**

\* `author_id` : valore da impostare nella variabile `author_id`.

- **`+getType() : PoiType`**  
Restituisce la variabile `type`.

- **`+setType( type : PoiType ) : void`**  
**Argomenti:**

\* `type` : valore da impostare nella variabile `type`.

- **`+getName() : String`**  
Restituisce la variabile `name`.

- **`+setName( name : String ) : void`**  
**Argomenti:**

\* `name` : valore da impostare nella variabile `name`.

- **`+getLocation() : Location`**  
Restituisce la variabile `location`.

- **`+ setLocation( location : Location ) : void`**  
**Argomenti:**

\* `location` : valore da impostare nella variabile `location`.

- **`+getAuthorName() : String`**  
Restituisce la variabile `authorName`.

- **`+setAuthorName( authorName : String ) : void`**  
**Argomenti:**

\* `authorName` : valore da impostare nella variabile `authorName`.

- **`+equals( o : Object ) : boolean`**  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.

- Argomenti:**

\* `o` : oggetto da confrontare con `this`.

- **`+hashcode( ) : int`**  
*Override* che genera un codice `hash` considerando tutti gli attributi della classe.

### 3.48.2.5 shike::web::model::session::track::Location

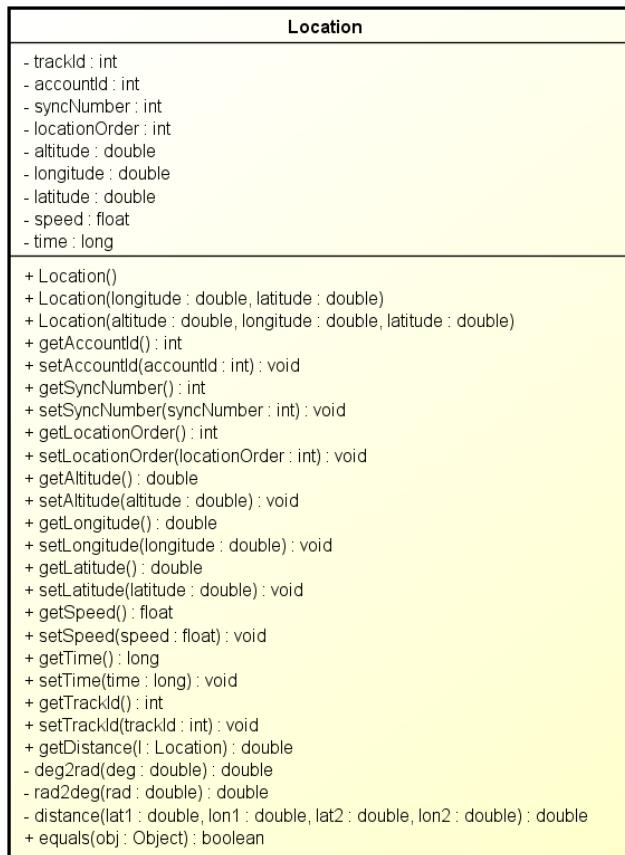


Figura 195: Diagramma di Location

- **Tipo:** concreta
- **Descrizione:** classe che determina un punto nello spazio, corrispondente alla classe *Location* di Android.
- **Attributi:**
  - **`_trackId : int`**  
Id del tracciato corrispondente.
  - **`_accountId : int`**  
Id dell' *account* a cui è associato il punto.
  - **`_syncNumber : int`**  
Numero della sincronizzazione a cui è associato il punto.
  - **`_locationOrder : int`**  
Ordine di percorrenza dei punti.
  - **`_altitude : double`**  
Altitudine del punto.
  - **`_longitude : double`**  
Longitudine del punto.

- `-latitude : double`  
Latitudine del punto.
- `-speed : float`  
Velocità in quel punto.
- `-time : long`  
Tempo in quel punto.

- **Metodi:**

- `+getTrackId() : int`  
Restituisce la variabile `trackId`.

- `+setTrackId( trackId : int ) : void`  
**Argomenti:**

\* `trackId` : valore da impostare nella variabile `trackId`.

- `+getAccountId() : int`  
Restituisce la variabile `accountId`.

- `+setAccountId( accountId : int ) : void`  
**Argomenti:**

\* `accountId` : valore da impostare nella variabile `accountId`.

- `+getSyncNumber() : int`  
Restituisce la variabile `syncNumber`.

- `+setSyncNumber( syncNumber : int ) : void`  
**Argomenti:**

\* `syncNumber` : valore da impostare nella variabile `syncNumber`.

- `+getLocationOrder() : int`  
Restituisce la variabile `locationOrder`.

- `+setLocationOrder( locationOrder : int ) : void`  
**Argomenti:**

\* `locationOrder` : valore da impostare nella variabile `locationOrder`.

- `+getAltitude() : double`  
Restituisce la variabile `altitude`.

- `+setAltitude( altitude : double ) : void`  
**Argomenti:**

\* `altitude` : valore da impostare nella variabile `altitude`.

- `+getLongitude() : double`  
Restituisce la variabile `longitude`.

- `+setLongitude( longitude : double ) : void`  
**Argomenti:**

\* `longitude` : valore da impostare nella variabile `longitude`.

- `+getLatitude() : double`  
Restituisce la variabile `latitude`.

- `+setLatitude( latitude : double ) : void`  
**Argomenti:**

\* `latitude` : valore da impostare nella variabile `latitude`.

- `+getSpeed() : float`  
Restituisce la variabile `speed`.

- `+setSpeed( speed : float ) : void`  
**Argomenti:**
  - \* `speed` : valore da impostare nella variabile `speed`.
- `+getTime() : long`  
 Restituisce la variabile `time`.
- `+setTime( time : long ) : void`  
**Argomenti:**
  - \* `time` : valore da impostare nella variabile `time`.
- `+equals( o : Object ) : boolean`  
*Overriding* che controlla l'uguaglianza per ogni singolo attributo della classe.  
**Argomenti:**
  - \* `o` : oggetto da confrontare con `this`.
- `+Location( ) :`  
 Costruttore della classe.
- `+Location( altitude : double, longitude : double, latitude : double ) :`  
 Costruttore della classe.  
**Argomenti:**
  - \* `altitude` : altitudine della *Location*.
  - \* `longitude` : longitudine della *Location*.
  - \* `latitude` : latitudine della *Location*.
- `+Location( longitude : double, latitude : double ) :`  
 Costruttore della classe.  
**Argomenti:**
  - \* `longitude` : longitudine della *Location*.
  - \* `latitude` : latitudine della *Location*.

### 3.48.2.6 shike::web::model::session::track::Poi::PoiType

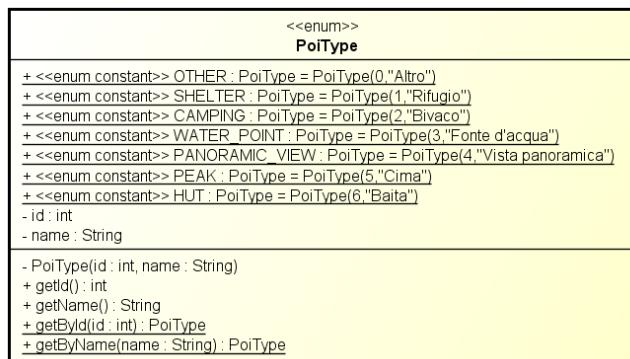


Figura 196: Diagramma di Poi::PoiType

- **Tipo:** enum
- **Descrizione:** enumeratore interno alla classe shike::web::model::session::track::Poi contenente le varie tipologie di POI.
- **Attributi:**

- **`-name : String`**  
Nome della tipologia di POI.
- **`-id : int`**  
Identificatore univoco della tipologia di POI.

- **Metodi:**

- **`+getName() : String`**  
Restituisce la variabile `name`.
  - **`+getId() : int`**  
Restituisce la variabile `id`.
  - **`+getById( id : int ) : PoiType`**  
Ritorna il tipo di POI con l'id inserito, o `null` se tale tipologia non esiste.
- Argomenti:**
- \* `id` : id del tipo di POI desiderato.
- **`+getByName( name : String ) : PoiType`**  
Ritorna il tipo di POI con il nome inserito, o `null` se tale tipologia non esiste. Se ci sono più tipologie con lo stesso nome, viene ritornata la prima dichiarata.
- Argomenti:**
- \* `name` : nome della tipologia di POI desiderata.

### 3.48.2.7 shike::web::model::session::track::VirtualTrack::Level

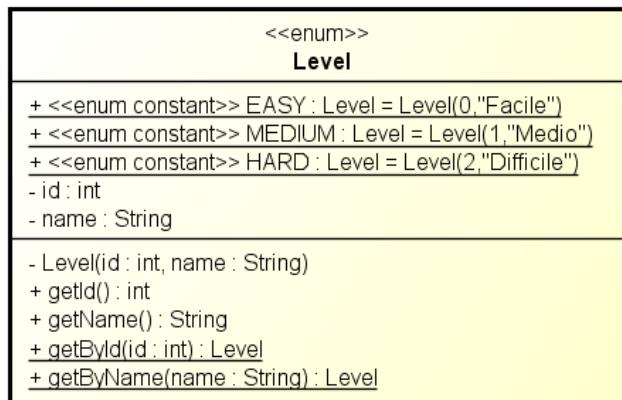


Figura 197: Diagramma di VirtualTrack::Level

- **Tipo:** enum
- **Descrizione:** enumerazione dei livelli disponibili del percorso.
- **Attributi:**
  - **`-id : int`**  
Identificatore univoco della difficoltà.
  - **`-name : String`**  
Stringa identificativa della difficoltà.
- **Metodi:**

- `+getId() : int`  
Restituisce la variabile `id`.
- `+getName() : String`  
Restituisce la variabile `name`.
- `+Level( name : String, id : int ) :`  
Costruttore dell'enumerazione.

**Argomenti:**

- \* `name` : stringa identificativa della difficoltà.
- \* `id` : identificatore univoco della difficoltà.

- `+getById( id : int ) : Level`  
Ritorna il tipo di `level` con l'id inserito, o `null` se tale tipologia non esiste. Se ci sono più tipologie con lo stesso nome, viene ritornata la prima dichiarata.

**Argomenti:**

- \* `id` : identificatore univoco della difficoltà.

- `+getByName( name : String ) : Level`  
Ritorna il tipo di `level` con il nome inserito, o `null` se tale tipologia non esiste. Se ci sono più tipologie con lo stesso nome, viene ritornata la prima dichiarata.

**Argomenti:**

- \* `name` : stringa identificativa della difficoltà.

### 3.49 shike::web::model::session::performance

#### 3.49.1 Informazioni sul package

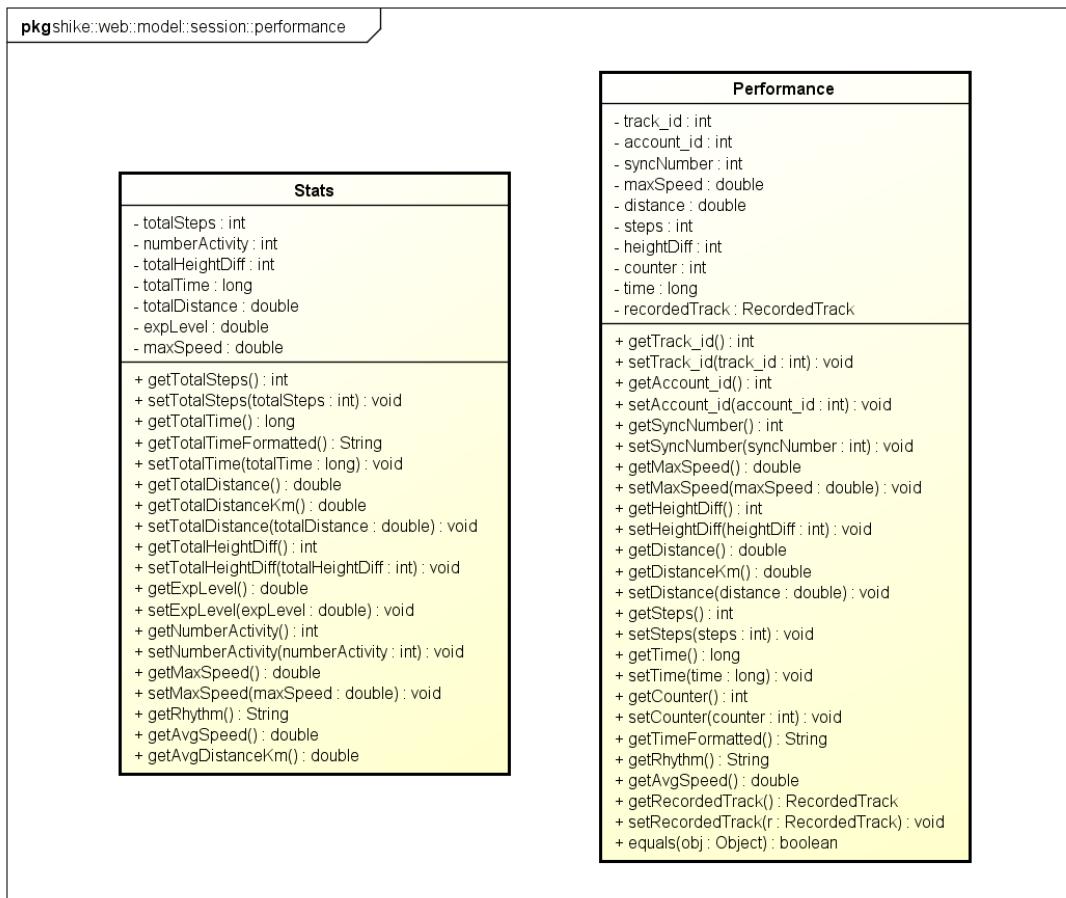


Figura 198: Diagramma di shike::web::model::session::performance

- **Descrizione:** componente del *Model* che raccoglie le informazioni statistiche relative alle sessioni effettuate dagli utenti.
- **Componente padre:** shike::web::model::session
- **Interazioni con altri componenti**
  - shike::web::model::user

### 3.49.2 Classi

#### 3.49.2.1 shike::web::model::session::performance::Performance

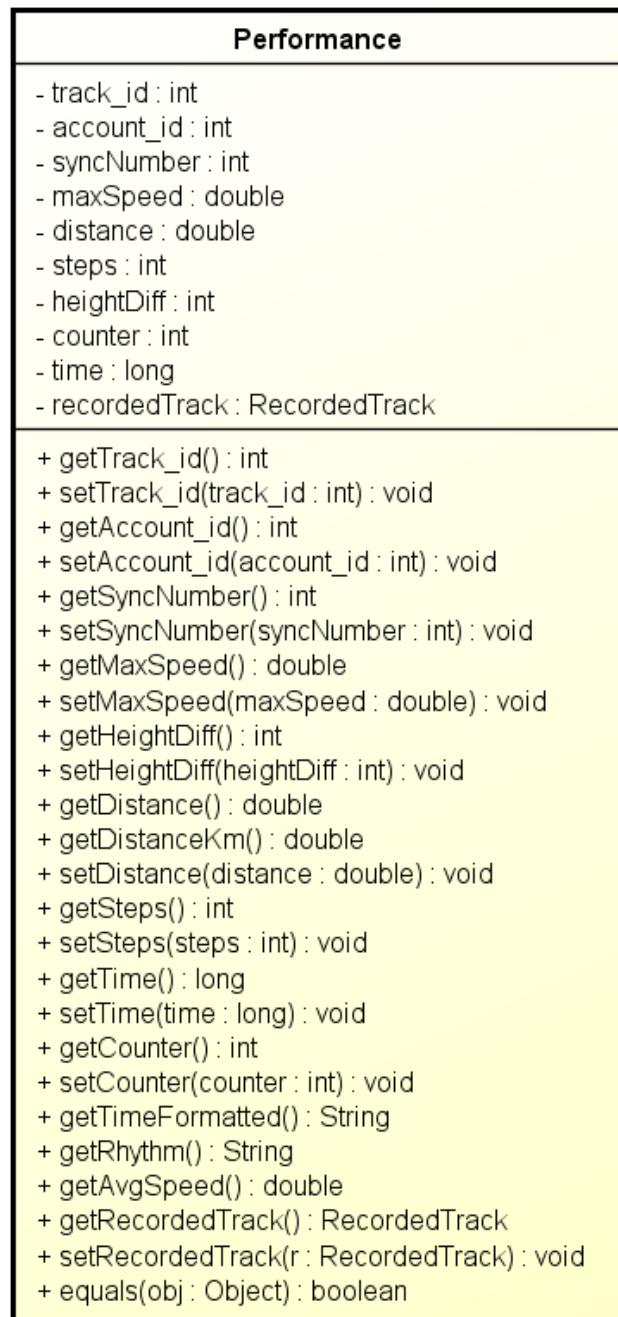


Figura 199: Diagramma di Performance

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di una performance svolta da un utente.

- **Attributi:**

- `_recordedTrack_id : int`  
Id del *RecordTrack* al quale la *performance* si riferisce.
- `_id : int`  
Id della *performance*.
- `_account_id : int`  
Id dell' *account* al quale la *performance* appartiene.
- `_syncNumber : int`  
Numero di sincronizzazione al quale la *performance* appartiene.
- `_date : int`  
Data della *performance* in ms da 01/01/1970.
- `_maxSpeed : double`  
Velocità massima della *performance*.
- `_heightDiff : double`  
Altitudine della *performance*.
- `_distance : double`  
Distanza totale della *performance*.
- `_maxHeartRate : int`  
Battito cardiaco massimo della *performance*.
- `_steps : int`  
Passi effettuati nella *performance*.
- `_kcals : int`  
Calorie consumate nella *performance*.
- `_time : long`  
Durata della *performance*.

- **Metodi:**

- `+getRecordedTrack_id() : int`  
Restituisce la variabile `recordedTrack_id`.  
**Argomenti:**  
\* `recordedTrack_id` : valore da impostare nella variabile `recordedTrack_id`.
- `+setRecordedTrack_id( recordedTrack_id : int ) : void`  
**Argomenti:**  
\* `recordedTrack_id` : valore da impostare nella variabile `recordedTrack_id`.
- `+get_id() : int`  
Restituisce la variabile `_id`.  
**Argomenti:**  
\* `_id` : valore da impostare nella variabile `_id`.
- `+set_id( _id : int ) : void`  
**Argomenti:**  
\* `_id` : valore da impostare nella variabile `_id`.
- `+getAccount_id() : int`  
Restituisce la variabile `account_id`.  
**Argomenti:**  
\* `account_id` : valore da impostare nella variabile `account_id`.
- `+setAccount_id( account_id : int ) : void`  
**Argomenti:**  
\* `account_id` : valore da impostare nella variabile `account_id`.
- `+getSyncNumber() : int`  
Restituisce la variabile `syncNumber`.

- `+setSyncNumber( syncNumber : int ) : void`  
**Argomenti:**
  - \* `syncNumber` : valore da impostare nella variabile syncNumber.
- `+getDate() : int`  
Restituisce la variabile date.
- `+ setDate( date : int ) : void`  
**Argomenti:**
  - \* `date` : valore da impostare nella variabile date.
- `+getMaxSpeed() : double`  
Restituisce la variabile maxSpeed.
- `+setMaxSpeed( maxSpeed : double ) : void`  
**Argomenti:**
  - \* `maxSpeed` : valore da impostare nella variabile maxSpeed.
- `+getHeightDiff() : double`  
Restituisce la variabile heightDiff.
- `+setHeightDiff( heightDiff : double ) : void`  
**Argomenti:**
  - \* `heightDiff` : valore da impostare nella variabile heightDiff.
- `+getDistance() : double`  
Restituisce la variabile distance.
- `+setDistance( distance : double ) : void`  
**Argomenti:**
  - \* `distance` : valore da impostare nella variabile distance.
- `+getMaxHeartRate() : int`  
Restituisce la variabile maxHeartRate.
- `+setMaxHeartRate( maxHeartRate : int ) : void`  
**Argomenti:**
  - \* `maxHeartRate` : valore da impostare nella variabile maxHeartRate.
- `+getSteps() : int`  
Restituisce la variabile steps.
- `+setSteps( steps : int ) : void`  
**Argomenti:**
  - \* `steps` : valore da impostare nella variabile steps.
- `+getKcals() : int`  
Restituisce la variabile kcals.
- `+setKcals( kcals : int ) : void`  
**Argomenti:**
  - \* `kcals` : valore da impostare nella variabile kcals.
- `+getTime() : long`  
Restituisce la variabile time.
- `+ setTime( time : long ) : void`  
**Argomenti:**
  - \* `time` : valore da impostare nella variabile time.

### 3.49.2.2 shike::web::model::session::performance::Stats

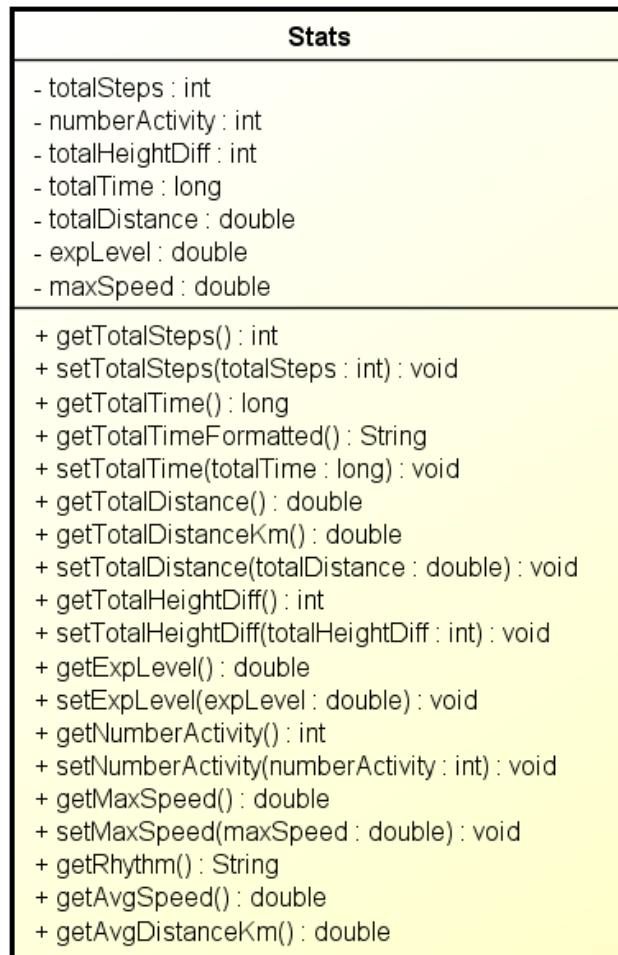


Figura 200: Diagramma di Stats

- **Tipo:** concreta
- **Descrizione:** classe che modella le statistiche generali dell’utente.
- **Attributi:**

- `totalDistance` : float  
Distanza totale percorsa.
- `totalTime` : int  
Tempo totale trascorso.
- `totalSteps` : int  
Passi totali effettuati.
- `totalHeighDiff` : int  
Metri totali di dislivello percorsi.
- `totalKcals` : int  
Calorie totali bruciate.

- `-expLevel : int`  
Livello di esperienza raggiunto.
- `-maxSpeed : double`  
Velocità massima.

- **Metodi:**

- `+getTotalDistance() : float`  
Restituisce la variabile totalDistance.
  - Argomenti:**
    - \* `totalDistance` : valore da impostare nella variabile totalDistance.
- `+setTotalDistance( totalDistance : float ) : void`  
**Argomenti:**
  - \* `totalDistance` : valore da impostare nella variabile totalDistance.
- `+getTotalTime() : int`  
Restituisce la variabile totalTime.
  - Argomenti:**
    - \* `totalTime` : valore da impostare nella variabile totalTime.
- `+setTotalTime( totalTime : int ) : void`  
**Argomenti:**
  - \* `totalTime` : valore da impostare nella variabile totalTime.
- `+getTotalSteps() : int`  
Restituisce la variabile totalSteps.
  - Argomenti:**
    - \* `totalSteps` : valore da impostare nella variabile totalSteps.
- `+setTotalSteps( totalSteps : int ) : void`  
**Argomenti:**
  - \* `totalSteps` : valore da impostare nella variabile totalSteps.
- `+getTotalHeighDiff() : int`  
Restituisce la variabile totalHeighDiff.
  - Argomenti:**
    - \* `totalHeighDiff` : valore da impostare nella variabile totalHeighDiff.
- `+setTotalHeighDiff( totalHeighDiff : int ) : void`  
**Argomenti:**
  - \* `totalHeighDiff` : valore da impostare nella variabile totalHeighDiff.
- `+getTotalKcals() : int`  
Restituisce la variabile totalKcals.
  - Argomenti:**
    - \* `totalKcals` : valore da impostare nella variabile totalKcals.
- `+setTotalKcals( totalKcals : int ) : void`  
**Argomenti:**
  - \* `totalKcals` : valore da impostare nella variabile totalKcals.
- `+getExpLevel() : int`  
Restituisce la variabile expLevel.
  - Argomenti:**
    - \* `expLevel` : valore da impostare nella variabile expLevel.
- `+setExpLevel( expLevel : int ) : void`  
**Argomenti:**
  - \* `expLevel` : valore da impostare nella variabile expLevel.
- `+getMaxSpeed() : double`  
Restituisce la variabile maxSpeed.
  - Argomenti:**
    - \* `maxSpeed` : valore da impostare nella variabile maxSpeed.
- `+setMaxSpeed( maxSpeed : double ) : void`  
**Argomenti:**
  - \* `maxSpeed` : valore da impostare nella variabile maxSpeed.

### 3.50 shike::web::model::dao

#### 3.50.1 Informazioni sul package

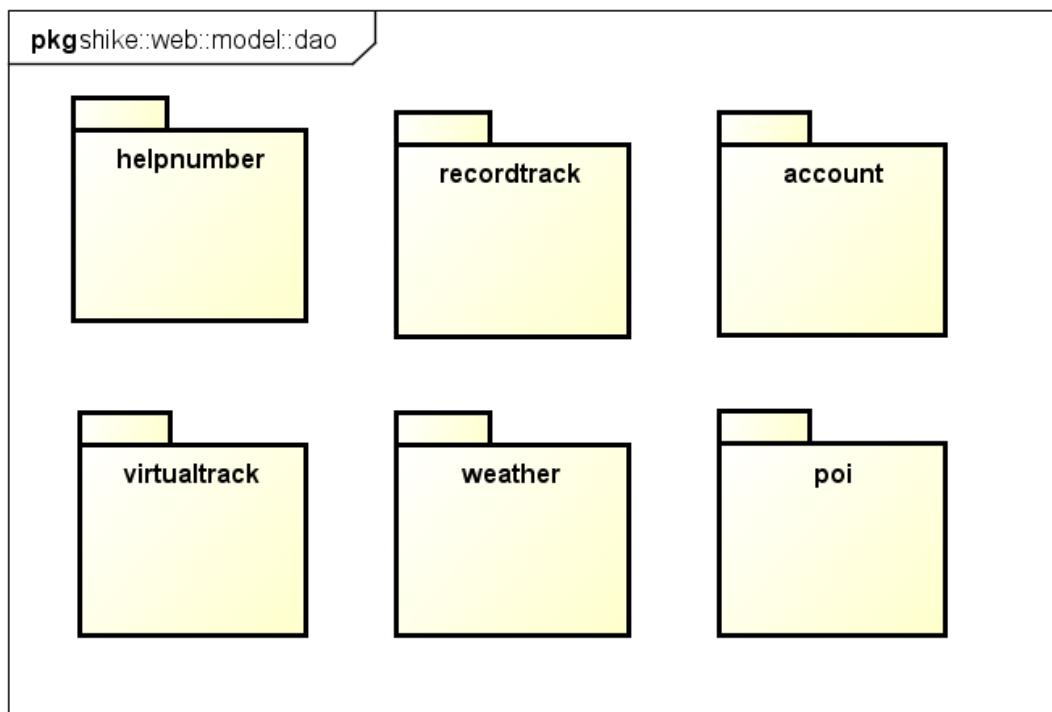


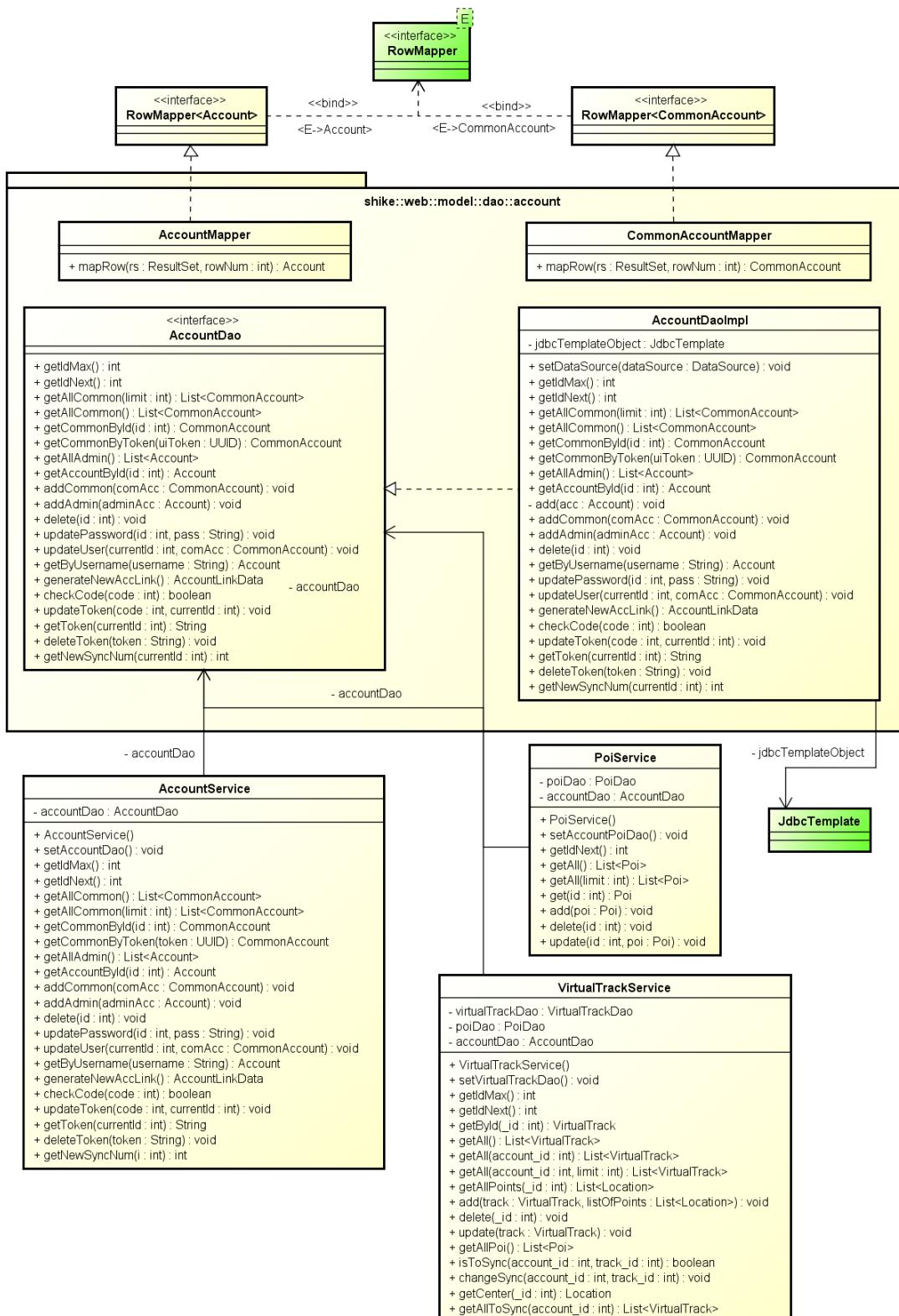
Figura 201: Diagramma di shike::web::model::dao

- **Descrizione:** componente del *Model* che contiene l'implementazione per l'accesso alla base dati degli oggetti implementati nel *model*.
- **Componenti contenute**
  - shike::web::model::dao::account
  - shike::web::model::dao::helpnumber
  - shike::web::model::dao::performance
  - shike::web::model::dao::poi
  - shike::web::model::dao::recordtrack
  - shike::web::model::dao::weather
  - shike::web::model::dao::virtualtrack
- **Componente padre:** shike::web::model
- **Interazioni con altri componenti**
  - shike::web::model::weather
  - shike::web::model::user
  - shike::web::model::session
  - shike::web::model::session::track
  - shike::web::model::session::performance



### 3.51 shike::web::model::dao::account

#### 3.51.1 Informazioni sul package



- **Descrizione:** componente che modella il *design pattern* DAO relativo agli *account*, sia per gli utenti sia per gli amministratori della piattaforma.
- **Componente padre:** shike::web::model::dao
- **Interazioni con altri componenti**
  - shike::web::model::user

### 3.51.2 Classi

#### 3.51.2.1 shike::web::model::dao::account::CommonAccountMapper

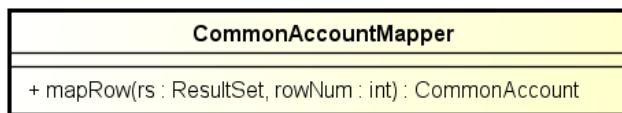


Figura 203: Diagramma di CommonAccountMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *CommonAccount*.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper
- **Metodi:**
  - **+mapRow( rs : ResultSet, rowNum : int ) : CommonAccount**  
Permette il collegamento tra il *Model* e il DAO.  
**Argomenti:**
    - \* **rs** : il ResultSet di una mappa.
    - \* **rowNum** : il numero di righe della mappa.

### 3.51.2.2 shike::web::model::dao::account::AccountDao

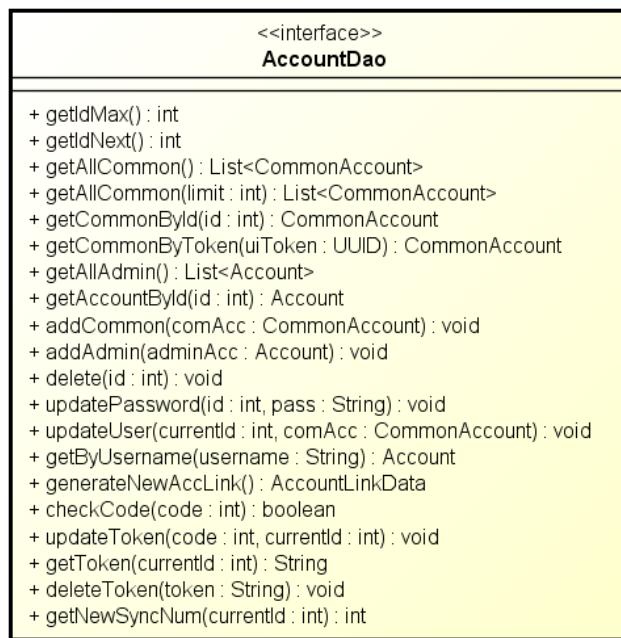


Figura 204: Diagramma di AccountDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un *account*, sia esso utente o amministratore, presente nel *database*.
- **Implementata da:**
  - shike::web::model::dao::account::**AccountDaoImpl**:
- **Metodi:**
  - **+getAllCommon( ) : List<CommonAccount>**  
Ritorna una lista di tutti gli utenti non amministratori presenti nel *database*.
  - **+getAllAdmin( ) : List<Account>**  
Ritorna una lista di tutti gli amministratori presenti nel *database*.
  - **+addCommon( comAcc : CommonAccount ) : void**  
Viene aggiunto al *database* un *CommonAccount*.  
**Argomenti:**
    - \* **comAcc** : commonAccount da aggiungere.
  - **+addAdmin( adminAcc : Account ) : void**  
Viene aggiunto al *database* un *Account*, cioè un amministratore.  
**Argomenti:**
    - \* **adminAcc** : account amministratore da aggiungere.
  - **+delete( id : int ) : void**  
Elimina l'utente con l'id selezionato.  
**Argomenti:**
    - \* **id** : id dell'utente da eliminare.

- \* `id` : id dell'utente da eliminare.
- `+getByUsername( username : String ) : Account`  
Ritorna l'utente con l'*username* corrispondente.
- Argomenti:**
  - \* `username` : username da cercare.
- `+add( acc : Account ) : void`  
Permette l'inserimento di un *Account* facendo l'hash della *password*.
- Argomenti:**
  - \* `acc` : l'account che si vuole aggiungere.
- `+updatePassword( id : int, pass : String ) : void`  
Permette di aggiornare la *password*.
- Argomenti:**
  - \* `id` : id dell'utente che vuole modificare la password.
  - \* `pass` : nuova password dell'utente.
- `+updateUser( currentId : int, comAcc : CommonAccount ) : void`  
Permette di aggiornare i dati di un utente comune.
- Argomenti:**
  - \* `currentId` : id dell'utente da modificare.
  - \* `comAcc` : nuovi dati utente.
- `+getAccountById( id : int ) : Account`  
Restituisce l' *account* associato ad un certo id.
- Argomenti:**
  - \* `id` : id dell'utente da restituire.
- `+getCommonById( id : int ) : CommonAccount`  
Restituisce l' *account* associato ad un certo id.
- Argomenti:**
  - \* `id` : id dell'utente da restituire.
- `+getIdNext() : int`  
Ritorna l'id del prossimo utente che si registrerà. Utile a livello di test.
- `+getIdMax() : int`  
Ritorna l'id dell'ultimo utente registrato.
- `+getAllCommon( limit : int ) : List<CommonAccount>`  
Ritorna una lista, di `limit` elementi, di tutti gli utenti non amministratori presenti nel *database*.
- Argomenti:**
  - \* `limit` : limita la lista ad un certo numero di utenti.
- `+checkCode( code : int ) : boolean`  
Controllo se un codice per l'associazione è valido o meno.
- Argomenti:**
  - \* `code` : codice da controllare.
- `+updateToken( code : int, currentId : int ) : void`  
Permette l'associazione di un dispositivo.
- Argomenti:**
  - \* `code` : codice da aggiornare.
  - \* `currentId` : id dell'utente.

- `+getToken( currentId: int ) : String`  
Se il dispositivo è connesso, restituisce il token di connessione.  
**Argomenti:**
  - \* `currentId` : id dell'utente.
- `+deleteToken( token : String ) : void`  
Permette di disconnettere il dispositivo.  
**Argomenti:**
  - \* `token` : il Token di connessione da cancellare.

### 3.51.2.3 shike::web::model::dao::account::AccountMapper

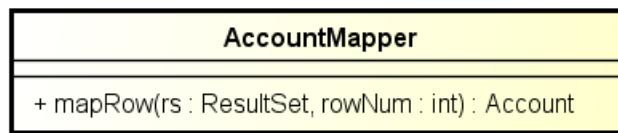


Figura 205: Diagramma di AccountMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *Account*.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper
- **Metodi:**
  - `+mapRow( rs : ResultSet, rowNum : int ) : Account`  
Permette il collegamento tra il *Model* e il DAO.  
**Argomenti:**
    - \* `rs` : il ResultSet di una mappa.
    - \* `rowNum` : il numero di righe della mappa.

### 3.51.2.4 shike::web::model::dao::account::AccountDaoImpl

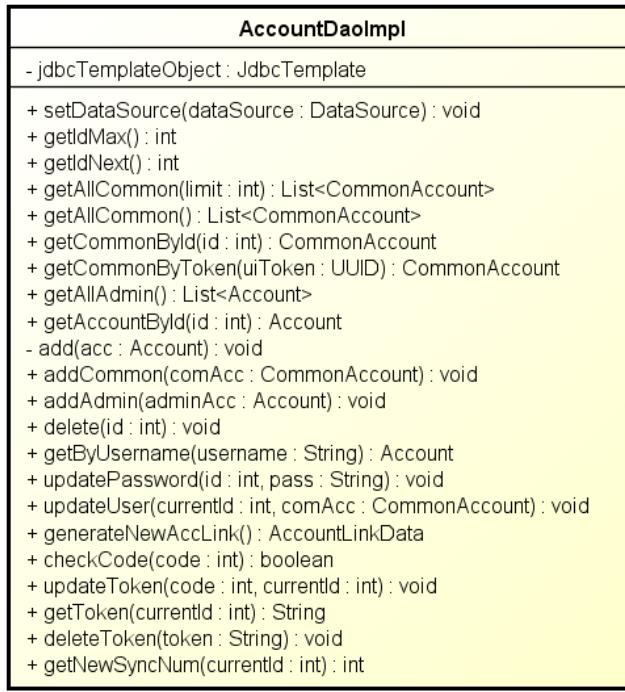


Figura 206: Diagramma di AccountDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *AccountDao*.
- **Implementa:**
  - shike::web::model::dao::account::AccountDao;
- **Attributi:**
  - `–jdbcTemplateObject : JdbcTemplate`  
*Template JDBC utilizzato per l'esecuzione delle query.*
- **Metodi:**
  - `+setDataSource( dataSource : DataSource ) : void`  
Imposta la sorgente dei dati del jdbcTemplateObject.  
**Argomenti:**
    - \* `dataSource` : sorgente dei dati.

### 3.52 shike::web::model::dao::helpnumber

#### 3.52.1 Informazioni sul package

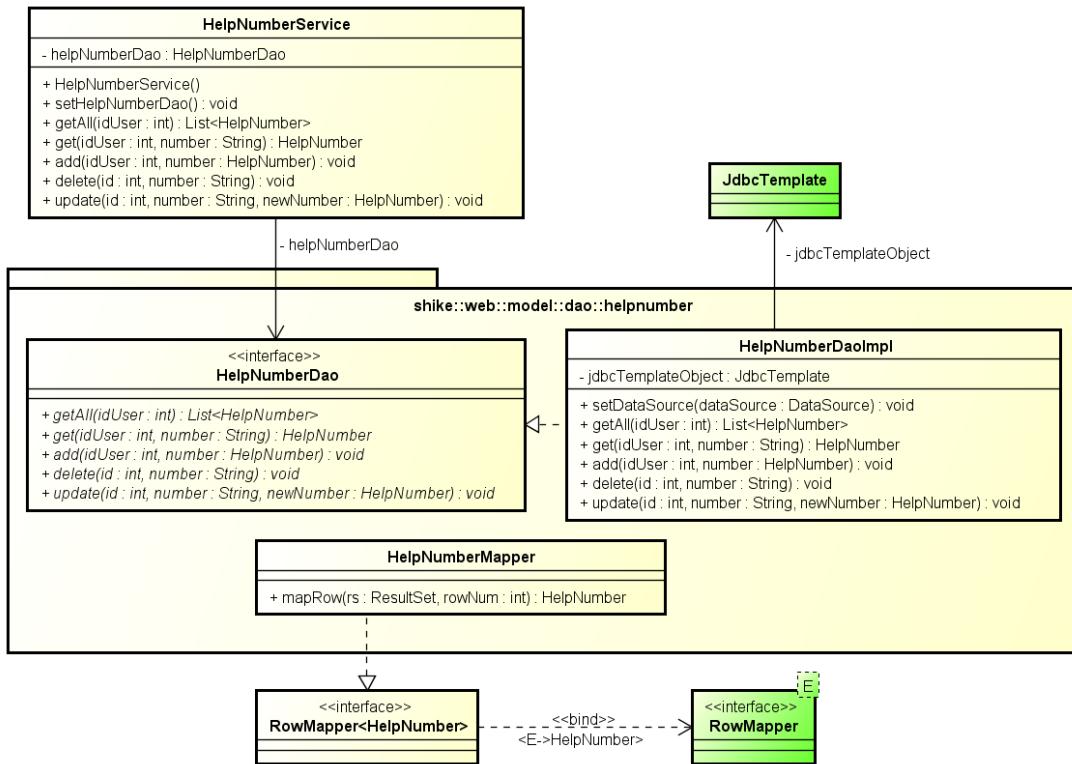


Figura 207: Diagramma di shike::web::model::dao::helpnumber

- Descrizione:** componente che modella il *design pattern* DAO riguardante i numeri utili associati ad un utente della piattaforma.
- Componente padre:** shike::web::model::dao
- Interazioni con altri componenti**
  - shike::web::model::user

#### 3.52.2 Classi

##### 3.52.2.1 shike::web::model::dao::helpnumber::HelpNumberDao

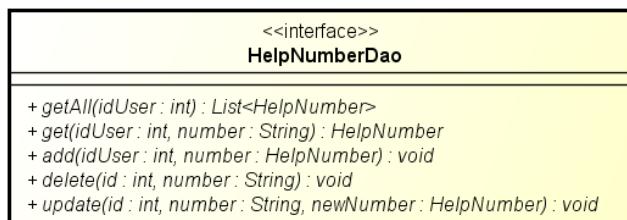


Figura 208: Diagramma di HelpNumberDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un numero di telefono di soccorso presente nel *database*.
- **Implementata da:**
  - shike::web::model::dao::helpnumber::**HelpNumberDaoImpl**:
- **Metodi:**
  - **+getAll( idUser : int ) : List<HelpNumber>**  
Permette di recuperare dal *database* tutti i numeri.  
**Argomenti:**
    - \* **idUser** : id dell'utente che richiede tutti i numeri.
  - **+get( idUser : int, number : String ) : HelpNumber**  
Permette di recuperare dal *database* un numero.  
**Argomenti:**
    - \* **idUser** : id dell'utente che richiede il numero.
    - \* **number** : numero da recuperare.
  - **+add( idUser : int, number : HelpNumber ) : void**  
Permette di aggiungere un numero.  
**Argomenti:**
    - \* **idUser** : id dell'utente che aggiunge il numero.
    - \* **number** : numero da aggiungere.
  - **+delete( id : int, number : String ) : void**  
Permette di eliminare un numero.  
**Argomenti:**
    - \* **id** : id dell'utente che elimina il numero.
    - \* **number** : numero da eliminare.
  - **+update( id : int, number : String, newNumber : HelpNumber ) : void**  
Permette di aggiornare un numero.  
**Argomenti:**
    - \* **id** : id dell'utente che modifica il numero.
    - \* **number** : vecchio numero da sostituire.
    - \* **newNumber** : nuovo numero da aggiungere.

### 3.52.2.2 shike::web::model::dao::helpnumber::HelpNumberMapper

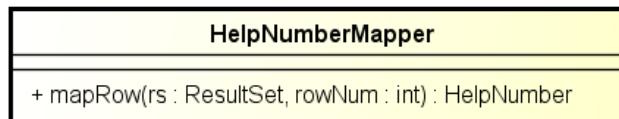


Figura 209: Diagramma di HelpNumberMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *HelpNumber*.

- **Implementa:**
  - org.springframework.jdbc.core.RowMapper
- **Metodi:**
  - `+mapRow( rs : ResultSet, rowNum : int ) : HelpNumber`  
Permette il collegamento tra il *Model* e il DAO.

**Argomenti:**

  - \* `rs` : il ResultSet di una mappa.
  - \* `rowNum` : il numero di righe della mappa.

### 3.52.2.3 shike::web::model::dao::helpnumber::HelpNumberDaoImpl

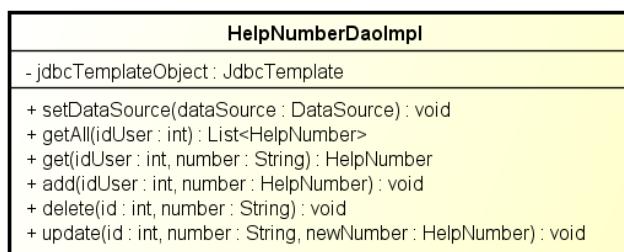


Figura 210: Diagramma di HelpNumberDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l’interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi dell’ *HelpNumberDao*.
- **Implementa:**
  - shike::web::model::dao::helpnumber::HelpNumberDao
- **Attributi:**
  - `–jdbcTemplateObject : JdbcTemplate`  
*Template JDBC utilizzato per l’esecuzione delle query.*
- **Metodi:**
  - `+setDataSource( dataSource : DataSource ) : void`  
Imposta la sorgente dei dati del jdbcTemplateObject.

**Argomenti:**

  - \* `dataSource` : sorgente dei dati.

## 3.53 shike::web::model::dao::performance

### 3.53.1 Informazioni sul package

- **Descrizione:** componente che modella il *design pattern* DAO riguardante le *performance* di un utente.
- **Componente padre:** shike::web::model::dao
- **Interazioni con altri componenti**
  - shike::web::model::session::performance

### 3.54 shike::web::model::dao::poi

#### 3.54.1 Informazioni sul package

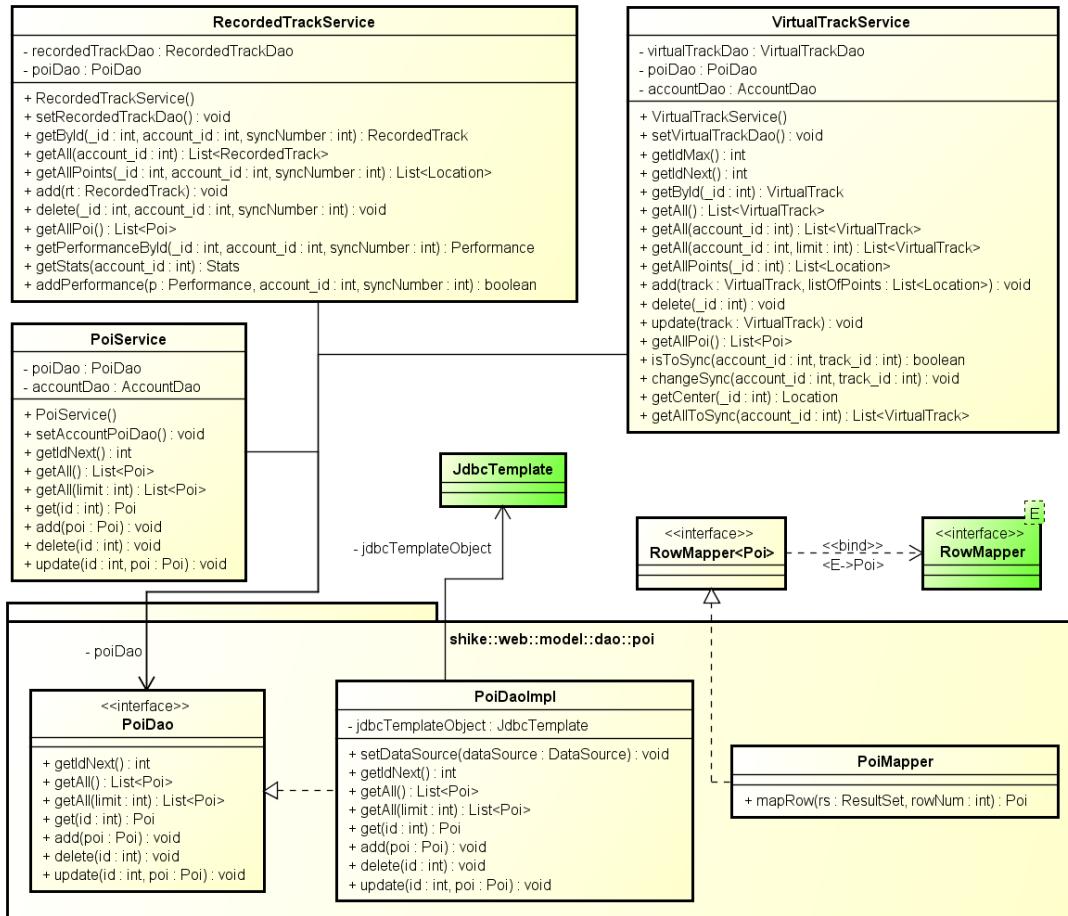


Figura 211: Diagramma di shike::web::model::dao::poi

- **Descrizione:** componente che modella il *design pattern* DAO riguardante i POI presenti all'interno di un percorso.
- **Componente padre:** shike::web::model::dao
- **Interazioni con altri componenti**
  - shike::web::model::session::track

### 3.54.2 Classi

#### 3.54.2.1 shike::web::model::dao::poi::PoiDao

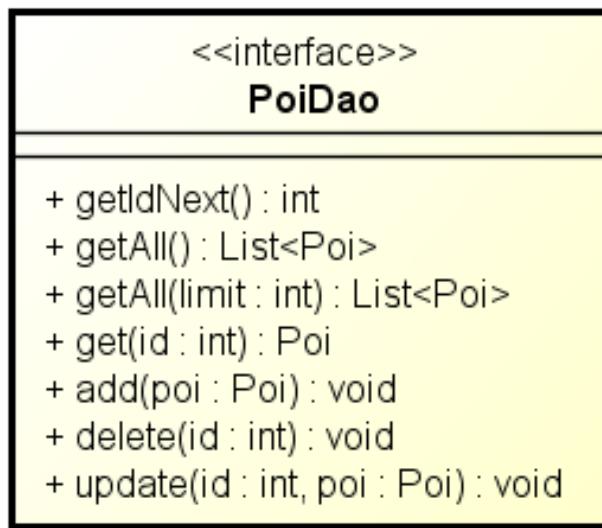


Figura 212: Diagramma di PoiDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un punto di interesse (POI) presente nel *database*.
- **Implementata da:**
  - shike::web::model::dao::poi::**PoiDaoImpl**:
- **Metodi:**
  - **+getAll()** : List<Poi>  
Permette di recuperare dal *database* tutti i POI.
  - **+get( id : int )** : Poi  
Permette di recuperare dal *database* un POI.
   
**Argomenti:**
    - \* id : id del poi richiesto.
  - **+add( poi : Poi )** : void  
Permette di aggiungere un POI.
   
**Argomenti:**
    - \* poi : poi da aggiungere.
  - **+delete( id : int )** : void  
Permette di eliminare un POI.
   
**Argomenti:**
    - \* id : id del poi da eliminare.
  - **+update( id : int, poi : Poi )** : void  
Permette di aggiornare un POI.
   
**Argomenti:**
    - \* id : id del poi da aggiornare.
    - \* poi : poi da aggiornare.

- \* `id` : id del poi da eliminare.
- \* `poi` : nuovo poi che aggiorna il precedente.
- `+getAll( limit : int ) : List<Poi>`  
Permette di recuperare dal *database* `limit` elementi POI.
- Argomenti:**
  - \* `limit` : numero di elementi da recuperare dal database.
- `+getIdNext() : int`  
Ritorna l'id del prossimo POI che verrà inserito. Utile a livello di test.

### 3.54.2.2 shike::web::model::dao::poi::PoiMapper

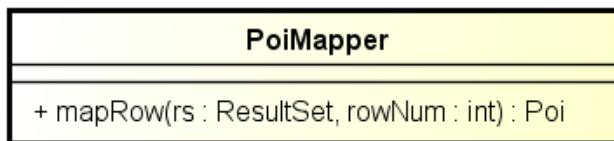


Figura 213: Diagramma di PoiMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un POI.
- **Implementa:**
  - `org.springframework.jdbc.core.RowMapper`
- **Metodi:**
  - `+mapRow( rs : ResultSet, rowNum : int ) : Poi`  
Permette il collegamento tra il *Model* e il DAO.
  - Argomenti:**
    - \* `rs` : il ResultSet di una mappa.
    - \* `rowNum` : il numero di righe della mappa.

### 3.54.2.3 shike::web::model::dao::poi::PoiDaoImpl

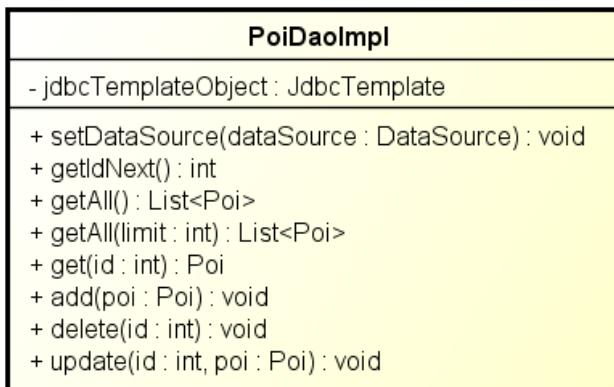


Figura 214: Diagramma di PoiDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un POI. Vengono definiti tutti i metodi del PoiDao.
- **Implementa:**
  - shike::web::model::dao::poi::**PoiDao**:
- **Attributi:**
  - **\_jdbcTemplateObject** : **JdbcTemplate**  
Template JDBC utilizzato per l'esecuzione delle *query*.
- **Metodi:**
  - **+setDataSource( dataSource : DataSource ) : void**  
Imposta la sorgente dei dati del jdbcTemplateObject.  
**Argomenti:**
    - \* **dataSource** : sorgente dei dati.

### 3.55 shike::web::model::dao::recordtrack

#### 3.55.1 Informazioni sul package

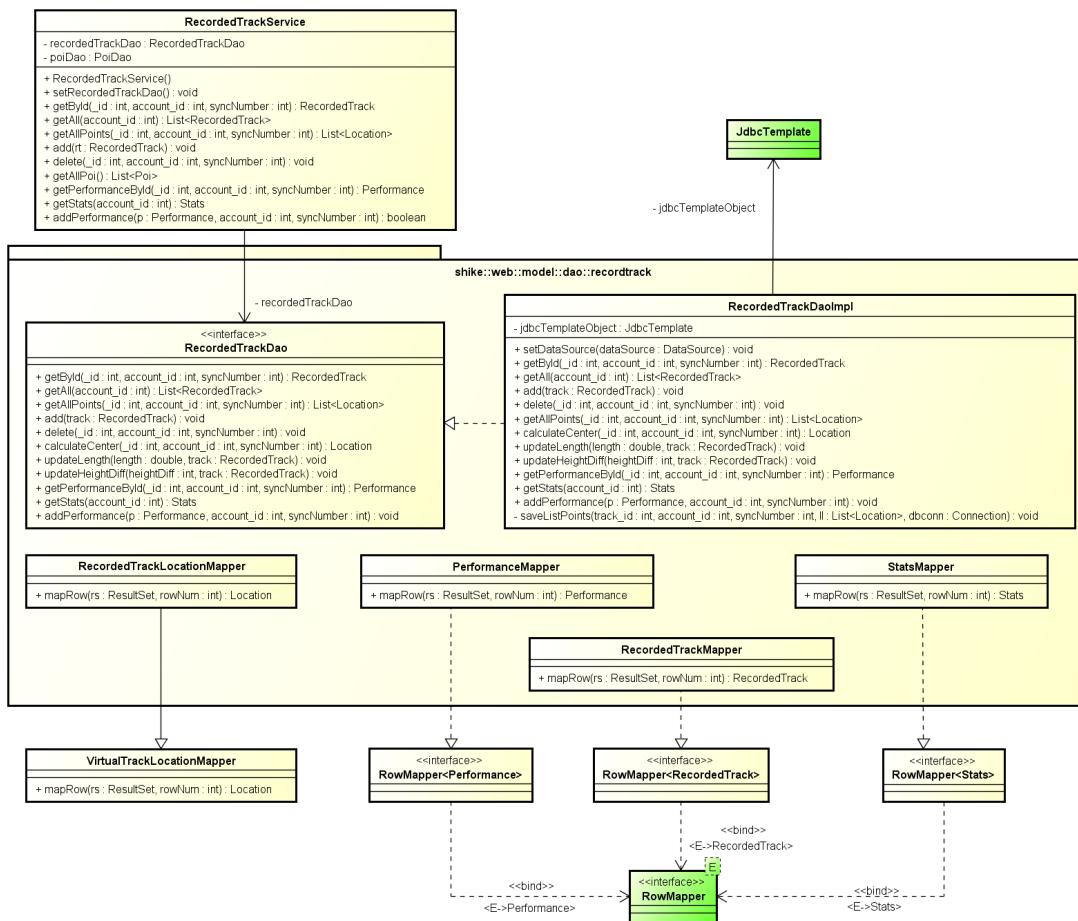


Figura 215: Diagramma di shike::web::model::dao::recordtrack

- **Descrizione:** componente che modella il *design pattern* DAO riguardante un tracciato.
- **Componente padre:** shike::web::model::dao
- **Interazioni con altri componenti**
  - shike::web::model::session::track

### 3.55.2 Classi

#### 3.55.2.1 shike::web::model::dao::recordtrack::RecordedTrackDao

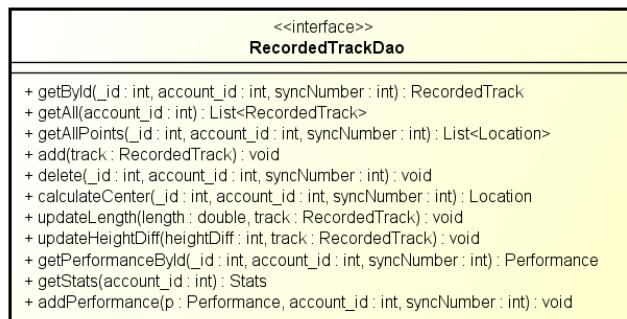


Figura 216: Diagramma di RecordedTrackDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un percorso presente nel *database*.
- **Implementata da:**
  - shike::web::model::dao::recordtrack::RecordedTrackDaoImpl
- **Metodi:**
  - **+getAll( account\_id : int ) : List<RecordedTrack>**  
Restituisce tutti i tracciati relativi ad un utente.  
**Argomenti:**
    - \* **account\_id** : id dell'utente che vuole la lista dei tracciati.
  - **+getAllPoints( \_id : int, account\_id : int, syncNumber : int ) : List<Location>**  
Restituisce tutti i punti di un tracciato selezionato.  
**Argomenti:**
    - \* **\_id** : id del tracciato.
    - \* **account\_id** : id dell'utente che vuole il tracciato.
    - \* **syncNumber** : numero di sync.
  - **+add( track : RecordedTrack ) : void**  
Inserisce nel db un nuovo RecordedTrack, passato come parametro. Viene utilizzato in fase di test, quindi i valori inseriti per i campi del RecordedTrack sono valori per i test.  
**Argomenti:**
    - \* **track** : nuovo percorso effettuato dall'utente da aggiungere.
  - **+delete( \_id : int, account\_id : int, syncNumber : int ) : void**  
Elimina dal database il percorso effettuato dall'utente identificato dai parametri passati.  
**Argomenti:**
    - \* **\_id** : id del tracciato da cancellare.
    - \* **account\_id** : id dell'utente che vuole cancellare il tracciato.

- \* `syncNumber` : numero della sincronizzazione effettuata dall'utente per caricare nella piattaforma il percorso da cancellare.
- `+getById( _id : int, int : account_id, int : syncNumber ) : RecordedTrack`  
Restituisce il tracciato con l'id selezionato.
- Argomenti:**
  - \* `_id` : id del tracciato da cancellare.
  - \* `int` : id dell'utente che vuole cancellare il tracciato.
  - \* `int` : numero di sync.
- `+updateLength( length : double, track : RecordedTrack ) : void`  
Aggiunge la distanza ad un percorso.
- Argomenti:**
  - \* `length` : distanza del percorso.
  - \* `track` : tracciato al quale si va ad aggiungere la distanza.
- `+updateHeightDiff( heightDiff : double, track : RecordedTrack ) : void`  
Aggiunge il dislivello ad un percorso.
- Argomenti:**
  - \* `heightDiff` : distanza del percorso.
  - \* `track` : tracciato al quale si va ad aggiungere il dislivello.
- `+getPerformanceById( _id : int, account_id : int, syncNumber : int ) : Performance`  
Recupera le performance riguardanti un certo percorso.
- Argomenti:**
  - \* `_id` : id del tracciato del quale si vuole recuperare le performance.
  - \* `account_id` : id dell'utente.
  - \* `syncNumber` : numero di sync.
- `+getStats( _id : int ) : Stats`  
Statistiche globali dell'utente selezionato.
- Argomenti:**
  - \* `_id` : id dell'utente.
- `+calculateCenter( syncNumber : int, account_id : int, id : int ) : Location`  
Calcola il punto centrale del percorso effettuato dall'utente identificato dai parametri passati.
- Argomenti:**
  - \* `syncNumber` : numero della sincronizzazione effettuata dall'utente per caricare nella piattaforma il percorso di cui calcolare il centro.
  - \* `account_id` : id del account che ha effettuato il percorso di cui calcolare il centro.
  - \* `id` : id del percorso di cui calcolare il centro.

### 3.55.2.2 shike::web::model::dao::recordtrack::PerformanceMapper

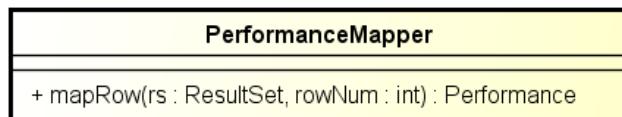


Figura 217: Diagramma di PerformanceMapper

- **Tipo:** concreta
- **Descrizione:** mapper che permette il collegamento al DAO per prelevare i dati di una Performance.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper

### 3.55.2.3 shike::web::model::dao::recordtrack::RecordedTrackDaoImpl

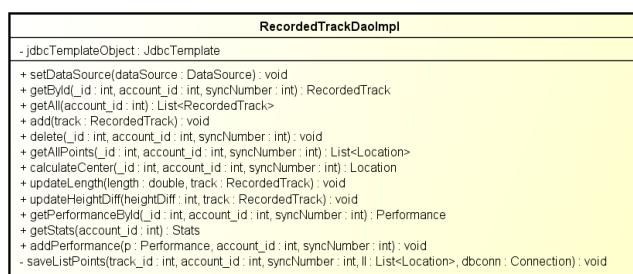


Figura 218: Diagramma di RecordedTrackDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.
- **Implementa:**
  - shike::web::model::dao::recordtrack::RecordedTrackDao
- **Attributi:**
  - **\_jdbcTemplateObject : JdbcTemplate**  
Template JDBC utilizzato per l'esecuzione delle *query*.
- **Metodi:**
  - **+setDataSource( dataSource : DataSource ) : void**  
Imposta la sorgente dei dati del jdbcTemplateObject.  
**Argomenti:**  
\* **dataSource** : sorgente dei dati.

### 3.55.2.4 shike::web::model::dao::recordtrack::RecordedTrackMapper



Figura 219: Diagramma di RecordedTrackMapper

- **Tipo:** concreta

- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *RecordedTrack*.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper
- **Metodi:**
  - `+mapRow(rs : ResultSet, rowNum : int) : RecordedTrack`  
Permette il collegamento tra il *Model* e il DAO.
  - Argomenti:**
    - \* `rs` : il ResultSet di una mappa.
    - \* `rowNum` : il numero di righe della mappa.

### 3.55.2.5 shike::web::model::dao::recordtrack::RecordedTrackLocationMapper



Figura 220: Diagramma di RecordedTrackLocationMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *RecordedTrackLocation*.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper
- **Metodi:**
  - `+mapRow(rs : ResultSet, rowNum : int) : RecordedTrackLocation`  
Permette il collegamento tra il *Model* e il DAO.
  - Argomenti:**
    - \* `rs` : il ResultSet di una mappa.
    - \* `rowNum` : il numero di righe della mappa.

### 3.55.2.6 shike::web::model::dao::recordtrack::StatsMapper

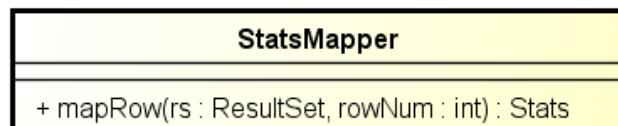


Figura 221: Diagramma di StatsMapper

- **Tipo:** concreta

- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *Stats*.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper

### 3.56 shike::web::model::dao::weather

#### 3.56.1 Informazioni sul package

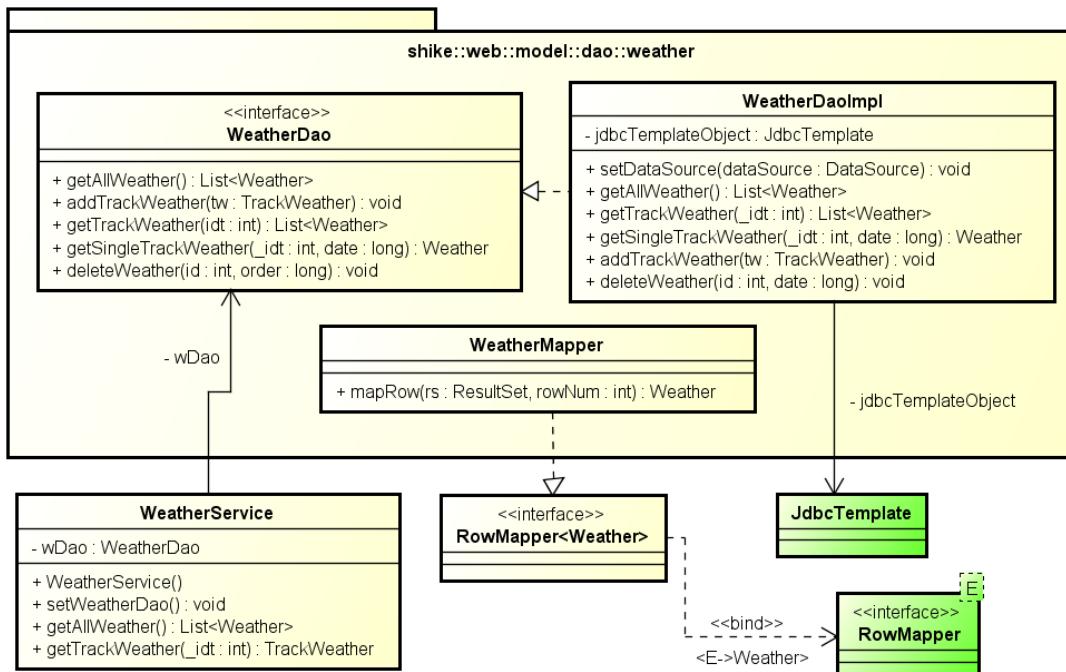


Figura 222: Diagramma di shike::web::model::dao::weather

- **Descrizione:** componente che modella il *design pattern* DAO riguardante il meteo di un certo percorso.
- **Componente padre:** shike::web::model::dao
- **Interazioni con altri componenti**
  - shike::web::model::weather

### 3.56.2 Classi

#### 3.56.2.1 shike::web::model::dao::weather::WeatherDao



Figura 223: Diagramma di WeatherDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad una previsione meteo presente nel *database*.
- **Implementata da:**
  - shike::web::model::dao::weather::WeatherDaoImpl
- **Metodi:**
  - **+listWeather()** : `List<Weather>`  
Metodo che viene usato per elencare tutti i record appartenenti alla tabella Weather.
  - **+add( tw : TrackWeather ) : void**  
Permette di aggiungere informazioni meteo alla tabella *weather*.  
**Argomenti:**
    - \* `tw` : traccia meteo da aggiungere.
  - **+delete( id : int, order : long ) : void**  
Permette la cancellazione di una traccia meteo.  
**Argomenti:**
    - \* `id` : id della previsione meteo.
    - \* `order` : `forecastOrder` del meteo da cancellare.

#### 3.56.2.2 shike::web::model::dao::weather::WeatherMapper

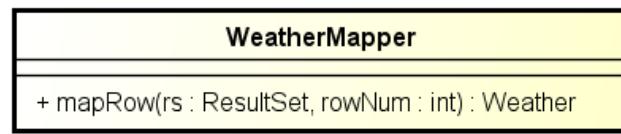


Figura 224: Diagramma di WeatherMapper

- **Tipo:** concreta
- **Descrizione:** mapper che permette il collegamento al DAO per prelevare i dati di un Weather.

- **Implementa:**
  - org.springframework.jdbc.core.RowMapper

- **Metodi:**

- `+mapRow( rs : ResultSet, rowNum : int ) : Weather`  
Permette il collegamento tra il *Model* e il DAO.

**Argomenti:**

- \* `rs` : il ResultSet di una mappa.
- \* `rowNum` : il numero di righe della mappa.

### 3.56.2.3 shike::web::model::dao::weather::WeatherDaoImpl

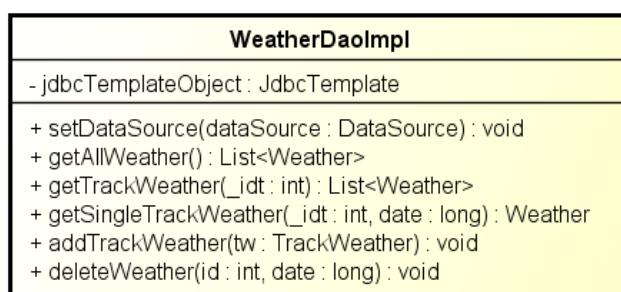


Figura 225: Diagramma di WeatherDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *WeatherDao*.
- **Implementa:**
  - shike::web::model::dao::weather::WeatherDao
- **Attributi:**
  - `_jdbcTemplateObject : JdbcTemplate`  
*Template JDBC* utilizzato per l'esecuzione delle *query*.
- **Metodi:**
  - `+setDataSource( dataSource : DataSource ) : void`  
Imposta la sorgente dei dati del `jdbcTemplateObject`.

**Argomenti:**

  - \* `dataSource` : sorgente dei dati.

### 3.57 shike::web::model::dao::virtualtrack

#### 3.57.1 Informazioni sul package

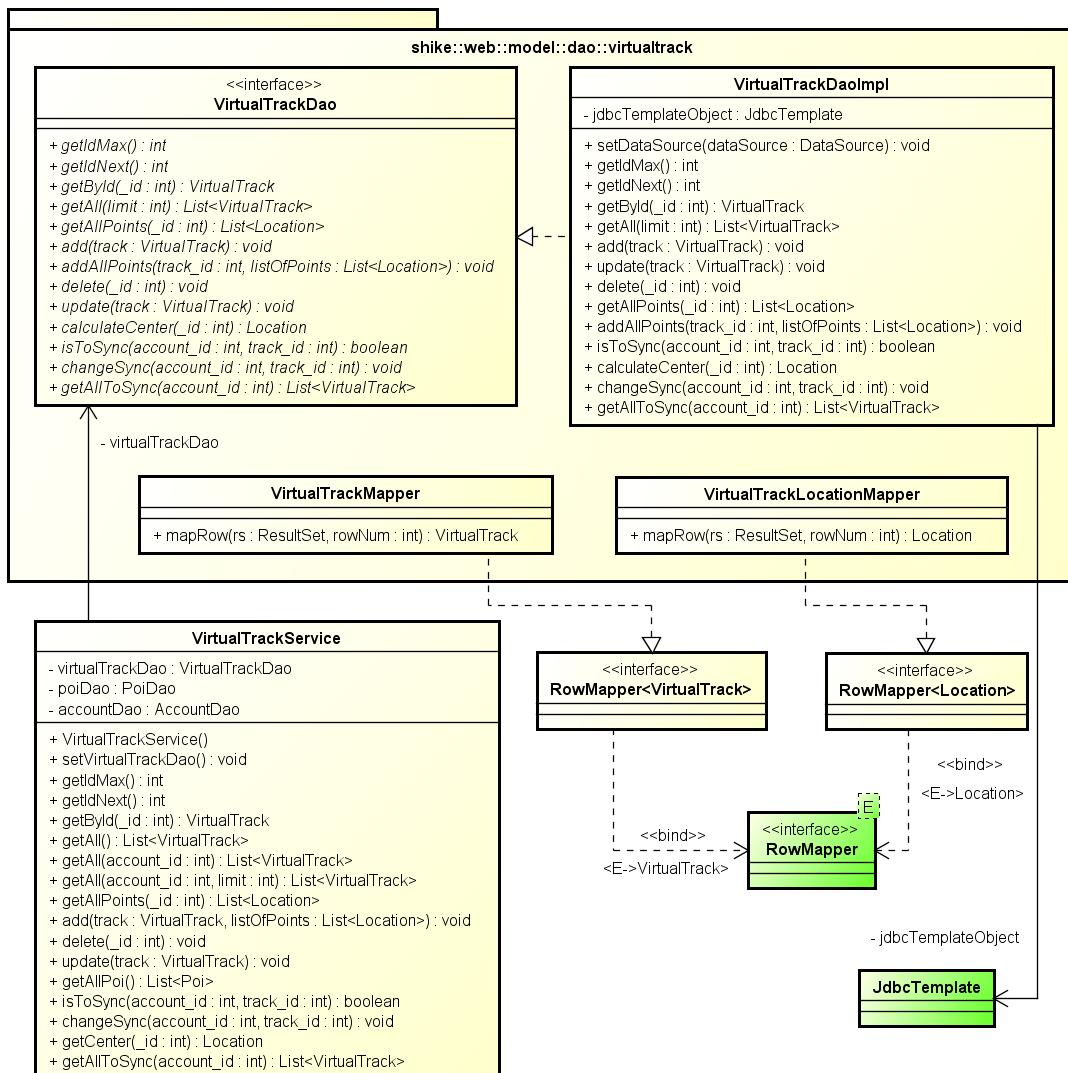


Figura 226: Diagramma di shike::web::model::dao::virtualtrack

- **Descrizione:** componente che modella il *design pattern* DAO riguardante un *virtual track*, cioè un percorso condiviso da un utente con gli altri.
- **Componente padre:** shike::web::model::dao
- **Interazioni con altri componenti**
  - shike::web::model::session::track

### 3.57.2 Classi

#### 3.57.2.1 shike::web::model::dao::virtualtrack::VirtualTrackDao

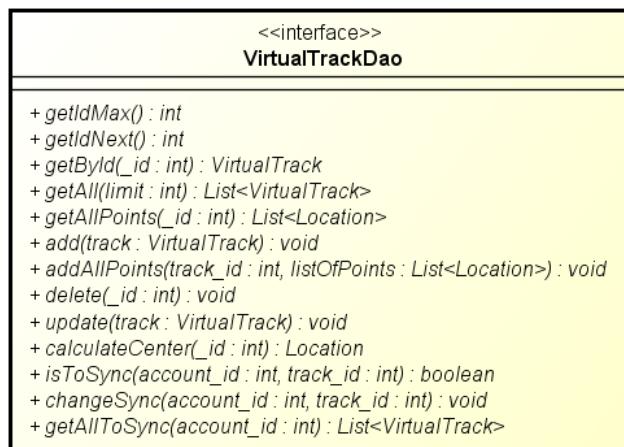


Figura 227: Diagramma di VirtualTrackDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un percorso che è stato condiviso dagli utenti.
- **Implementata da:**
  - shike::web::model::dao::virtualtrack::VirtualTrackDaoImpl
- **Metodi:**
  - **+getAll( ) : List<VirtualTrack>**  
Restituisce tutti i tracciati condivisi degli utenti.
  - **+getAllPoints( \_id : int ) : List<Location>**  
Restituisce tutti i punti *Location* di un tracciato.  
**Argomenti:**  
\* *\_id* : id del tracciato virtuale.
  - **+add( track : VirtualTrack ) : void**  
Aggiunge un virtual *track* al *database*.  
**Argomenti:**  
\* *track* : virtualTrack da aggiungere.
  - **+delete( \_id : int ) : void**  
Cancella un virtual *track* dal *database*.  
**Argomenti:**  
\* *\_id* : id del tracciato virtuale da cancellare.
  - **+getById( \_id : int ) : VirtualTrack**  
Seleziona un virtual *track* singolo.  
**Argomenti:**  
\* *\_id* : id del tracciato virtuale.

- `+update( track : VirtualTrack ) : void`  
Aggiorna il percorso con i campi del nuovo percorso passato per parametro, selezionando il percorso da modificare tramite l'id del percorso passato come parametro.  
**Argomenti:**  
\* `track` : percorso aggiornato.
- `+calculateCenter( _id : int ) : Location`  
Calcola il punto centrale del percorso identificato dall'id passato come parametro.  
**Argomenti:**  
\* `_id` : id del percorso di cui calcolare il punto centrale.
- `+addAllPoints( listOfPoints : List<Location>, track_id : int ) : void`  
Aggiunge al percorso i punti che lo compongono, passati come parametro.  
**Argomenti:**  
\* `listOfPoints` : elenco dei punti da aggiungere.  
\* `track_id` : id del percorso a cui aggiungere i punti che lo compongono.

### 3.57.2.2 shike::web::model::dao::virtualtrack::VirtualTrackMapper

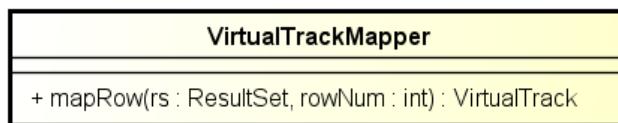


Figura 228: Diagramma di VirtualTrackMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *VirtualTrack*.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper
- **Metodi:**
  - `+mapRow( rs : ResultSet, rowNum : int ) : VirtualTrack`  
Permette il collegamento tra il *Model* e il DAO.  
**Argomenti:**  
\* `rs` : il ResultSet di una mappa.  
\* `rowNum` : il numero di righe della mappa.

### 3.57.2.3 shike::web::model::dao::virtualtrack::VirtualTrackLocationMapper

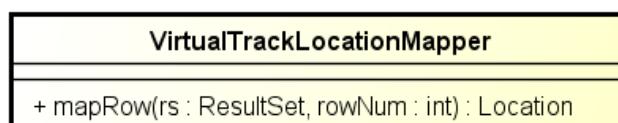


Figura 229: Diagramma di VirtualTrackLocationMapper

- **Tipo:** concreta

- **Descrizione:** Mapper che permette il collegamento al DAO per prelevare i dati di un *VirtualTrackLocation*.
- **Implementa:**
  - org.springframework.jdbc.core.RowMapper
- **Metodi:**
  - `+mapRow(rs : ResultSet, rowNum : int) : VirtualTrackLocation`  
Permette il collegamento tra il *Model* e il DAO.
  - Argomenti:**
    - \* `rs` : il ResultSet di una mappa.
    - \* `rowNum` : il numero di righe della mappa.

### 3.57.2.4 shike::web::model::dao::virtualtrack::VirtualTrackDaoImpl

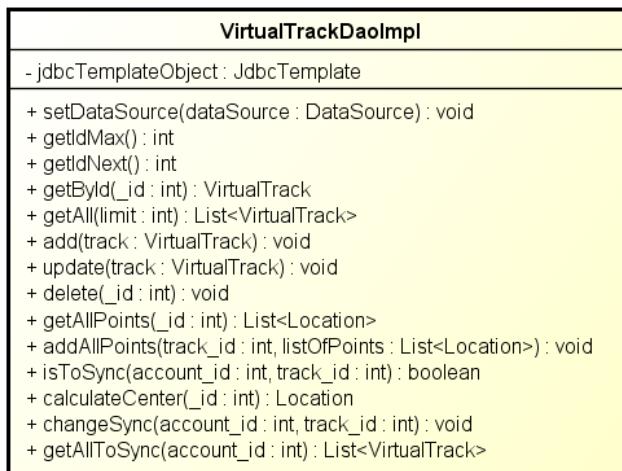


Figura 230: Diagramma di VirtualTrackDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *VirtualTrackDao*.
- **Implementa:**
  - shike::web::model::dao::virtualtrack::VirtualTrackDao
- **Attributi:**
  - `–jdbcTemplateObject : JdbcTemplate`  
*Template JDBC* utilizzato per l'esecuzione delle *query*.
- **Metodi:**
  - `+setDataSource(dataSource : DataSource) : void`  
Imposta la sorgente dei dati del jdbcTemplateObject.
  - Argomenti:**
    - \* `dataSource` : sorgente dei dati.

### 3.58 shike::web::model::service

#### 3.58.1 Informazioni sul package

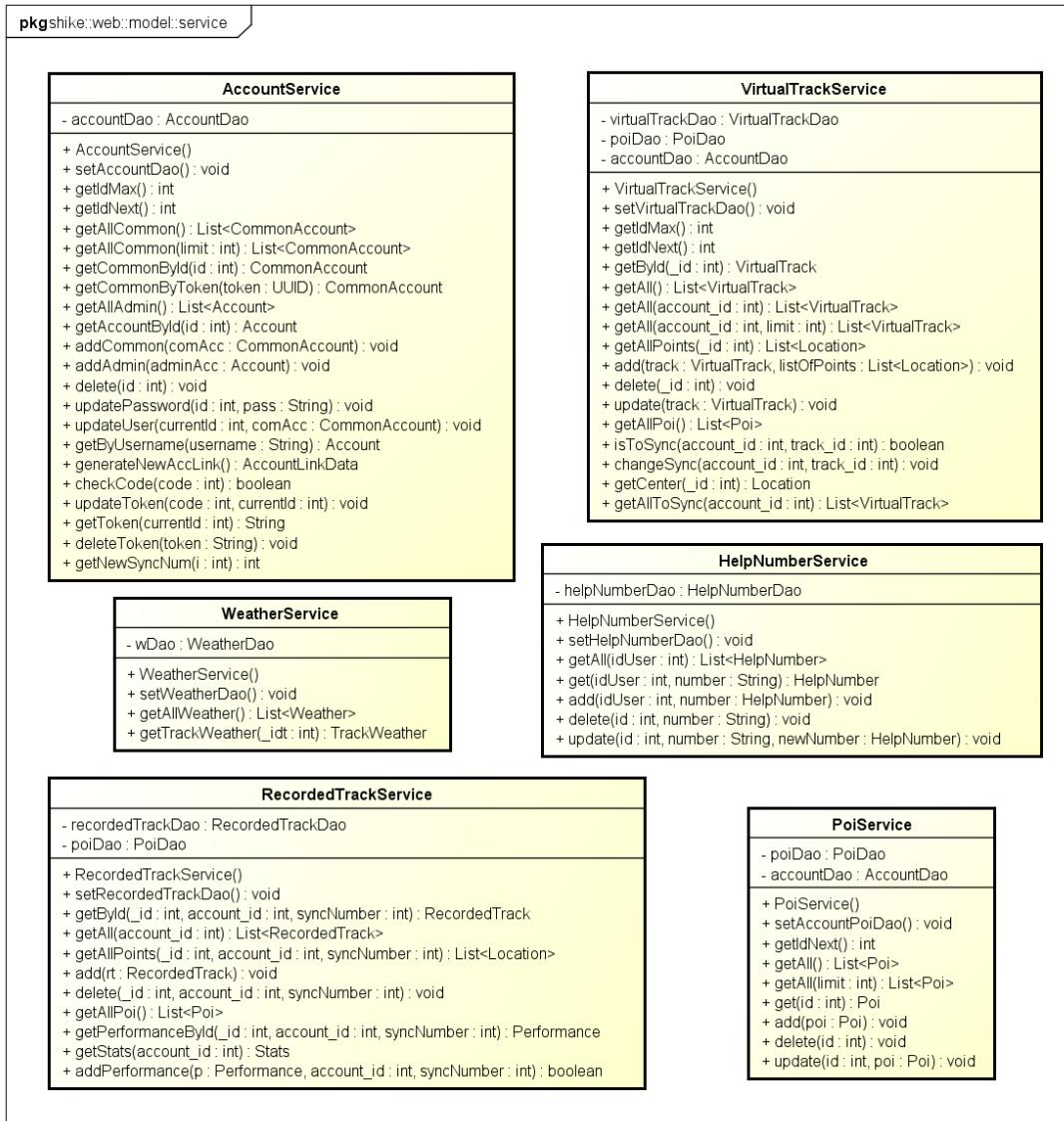


Figura 231: Diagramma di shike::web::model::service

- **Descrizione:** componente contenente i servizi che permettono ai controller di accedere ai DAO restituendo l'oggetto desiderato.
- **Componente padre:** shike::web::model

### 3.58.2 Classi

#### 3.58.2.1 shike::web::model::service::VirtualTrackService

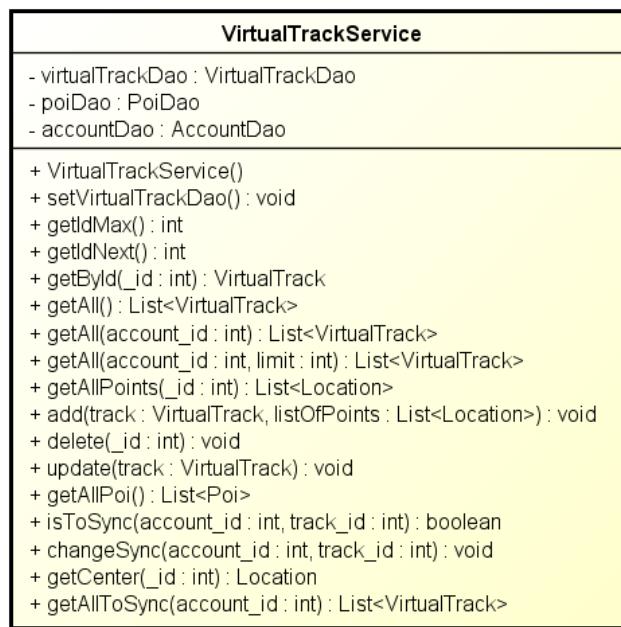


Figura 232: Diagramma di VirtualTrackService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *VirtualTrack* tramite la classe DAO corrispondente.
- **Attributi:**
  - **`-virtualTrackDao : VirtualTrackDao`**  
DAO del *VirtualTrack*.
  - **`-poiDao : PoiDao`**  
DAO dei *Poi*.
  - **`-accountDao : AccountDao`**  
DAO degli *Account*.
- **Metodi:**
  - **`+getAll() : List<VirtualTrack>`**  
Restituisce tutti i tracciati condivisi degli utenti.
  - **`+getAllPoints( _id : int ) : List<Location>`**  
Restituisce tutti i punti *Location* di un tracciato.  
**Argomenti:**  
\* `_id` : id del tracciato virtuale.
  - **`+add( track : VirtualTrack ) : void`**  
Aggiunge un virtual *track* al *database*.  
**Argomenti:**

- \* `track` : virtualTrack da aggiungere.
- `+delete( _id : int ) : void`  
Cancella un virtual *track* dal database.  
**Argomenti:**  
\* `_id` : id del tracciato virtuale da cancellare.
- `+getById( _id : int ) : VirtualTrack`  
Seleziona un virtual *track* singolo.  
**Argomenti:**  
\* `_id` : id del tracciato virtuale.
- `+update( track : VirtualTrack ) : void`  
Aggiorna il percorso con i campi del nuovo percorso passato per parametro, selezionando il percorso da modificare tramite l'id del percorso passato come parametro.  
**Argomenti:**  
\* `track` : percorso aggiornato.
- `+calculateCenter( _id : int ) : Location`  
Calcola il punto centrale del percorso identificato dall'id passato come parametro.  
**Argomenti:**  
\* `_id` : id del percorso di cui calcolare il punto centrale.
- `+addAllPoints( track_id : int, listOfPoints : List<Location> ) : void`  
Aggiunge al percorso i punti che lo compongono, passati come parametro.  
**Argomenti:**  
\* `track_id` : id del percorso a cui aggiungere i punti che lo compongono.  
\* `listOfPoints` : elenco dei punti da aggiungere.
- `+addAllPoints( ) : void`  
Aggiunge al percorso i punti che lo compongono, passati come parametro.
- `+addAllPoints( ) : void`  
Aggiunge al percorso i punti che lo compongono, passati come parametro.
- `+addAllPoints( ) : void`  
Aggiunge al percorso i punti che lo compongono, passati come parametro.
- `+addAllPoints( ) : void`  
Aggiunge al percorso i punti che lo compongono, passati come parametro.

### 3.58.2.2 shike::web::model::service::WeatherService

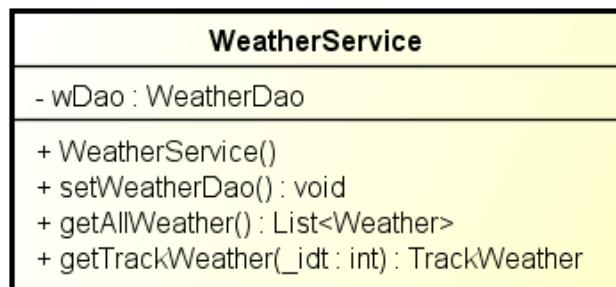


Figura 233: Diagramma di WeatherService

- **Tipo:** concreta

- **Descrizione:** classe che permette di recuperare un riferimento ad un *Weather* tramite la classe DAO corrispondente.

- **Attributi:**

- `_wDao : WeatherDao`  
DAO del Weather.

- **Metodi:**

- `+WeatherService() :`  
Costruttore della classe.
- `+setWeatherDao() :`  
Imposta il DAO del Weather.
- `+getAllWeather() : List<Weather>`  
Ritorna la lista delle previsioni meteo.
- `+getTrackWeather( _idt : int ) : TrackWeather`  
Ritorna il meteo del tracciato.

**Argomenti:**

\* `_idt : .`

### 3.58.2.3 shike::web::model::service::AccountService

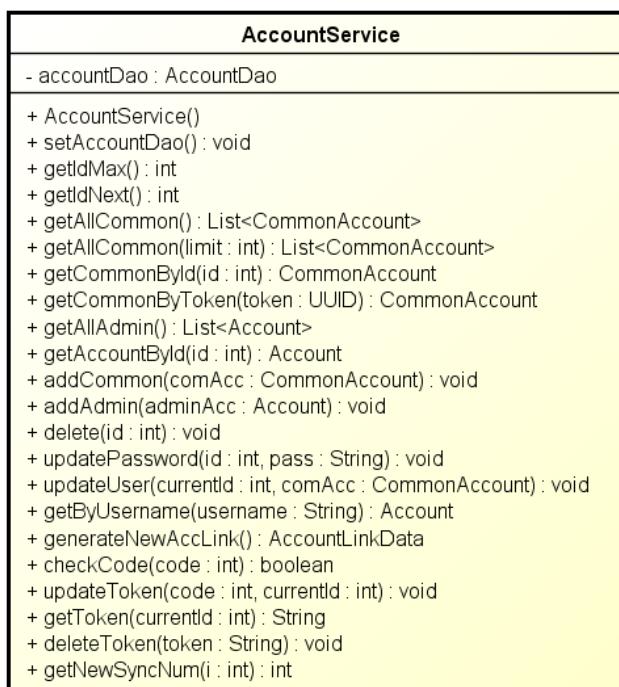


Figura 234: Diagramma di AccountService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.

- **Attributi:**

- `–accountDao : AccountDao`  
DAO degli account.

- **Metodi:**

- `+setAccountDao( accountDao : AccountDao ) : void`  
**Argomenti:**

\* `accountDao` : valore da impostare nella variabile `accountDao`.

- `+getAccountById( id : int ) : Account`  
Restituisce l'*account* associato ad un certo id.

**Argomenti:**

\* `id` : id dell'utente da restituire.

- `+AccountService( ) :`  
Costruttore che imposta automaticamente `accountDao`.
- `+getCommonById( id : int ) : CommonAccount`  
Restituisce l'*account* associato ad un certo id.

**Argomenti:**

\* `id` : id dell'utente da restituire.

- `+getIdNext( ) : int`  
Ritorna l'id del prossimo utente che si registrerà. Utile a livello di test.
- `+getIdMax( ) : int`  
Ritorna l'id dell'ultimo utente registrato.
- `+updateUser( comAcc : CommonAccount, currentId : int ) : void`  
Permette di aggiornare i dati di un utente comune.

**Argomenti:**

\* `comAcc` : nuovi dati utente.

\* `currentId` : id dell'utente da modificare.

- `+updatePassword( id : int, pass : String ) : void`  
Permette di aggiornare la *password*.

**Argomenti:**

\* `id` : id dell'utente che vuole modificare la password.

\* `pass` : nuova password dell'utente.

- `+updatePassword( ) : void`  
Permette di aggiornare la *password*.
- `+add( acc : Account ) : void`  
Permette l'inserimento di un *Account* facendo l'hash della *password*.

**Argomenti:**

\* `acc` : l'account che si vuole aggiungere.

- `+getByUsername( username : String ) : Account`  
Ritorna l'utente con l'*username* corrispondente.

**Argomenti:**

\* `username` : username da cercare.

- `+delete( id : int ) : void`  
Elimina l'utente con l'id selezionato.

**Argomenti:**

- \* `id` : id dell'utente da eliminare.
- `+addAdmin( adminAcc : Account ) : void`  
Viene aggiunto al *database* un *Account*, cioè un amministratore.  
**Argomenti:**
  - \* `adminAcc` : account amministratore da aggiungere.
- `+addCommon( comAcc : CommonAccount ) : void`  
Viene aggiunto al *database* un *CommonAccount*.  
**Argomenti:**
  - \* `comAcc` : commonAccount da aggiungere.
- `+getAllAdmin() : List<Account>`  
Ritorna una lista di tutti gli amministratori presenti nel database.
- `+getAllCommon( limit : int ) : List<CommonAccount>`  
Ritorna una lista, di `limit` elementi, di tutti gli utenti non amministratori presenti nel *database*.  
**Argomenti:**
  - \* `limit` : limita la lista ad un certo numero di utenti.
- `+getAllCommon() : List<CommonAccount>`  
Ritorna una lista di tutti gli utenti non amministratori presenti nel *database*.
- `+getCommonByToken( token : UUID ) : CommonAccount`  
Ritorna il *CommonAccount* che ha come token il token passato come parametro.  
**Argomenti:**
  - \* `token` : token per il collegamento Account tra parte App e parte Web.

### 3.58.2.4 shike::web::model::service::HelpNumberService

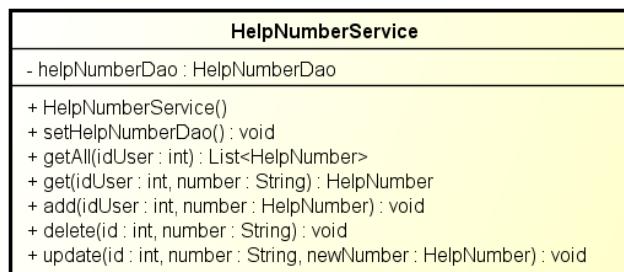


Figura 235: Diagramma di HelpNumberService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *HelpNumber* tramite la classe DAO corrispondente.
- **Attributi:**
  - `-helpNumberDao : HelpNumberDao`  
DAO degli *HelpNumber*.
- **Metodi:**

- `+setHelpNumberDao( helpNumberDao : HelpNumberDao ) : void`  
**Argomenti:**
  - \* `helpNumberDao` : valore da impostare nella variabile `helpNumberDao`.
- `+HelpNumberService( ) :`  
Costruttore che imposta automaticamente `helpNumberDao`.
- `+getAll(idUser : int) : List<HelpNumber>`  
Permette di recuperare dal *database* tutti i numeri.  
**Argomenti:**
  - \* `idUser` : id dell'utente che richiede tutti i numeri.
- `+get(idUser : int, number : String) : HelpNumber`  
Permette di recuperare dal *database* un numero.  
**Argomenti:**
  - \* `idUser` : id dell'utente che richiede il numero.
  - \* `number` : numero da recuperare.
- `+add(idUser : int, number : String) : void`  
Permette di aggiungere un numero.  
**Argomenti:**
  - \* `idUser` : id dell'utente che aggiunge il numero.
  - \* `number` : numero da aggiungere.
- `+delete(idUser : int, number : String) : void`  
Permette di eliminare un numero.  
**Argomenti:**
  - \* `idUser` : id dell'utente che elimina il numero.
  - \* `number` : numero da eliminare.
- `+update( id : int, number : String, newNumber : HelpNumber ) : void`  
Permette di aggiornare un numero.  
**Argomenti:**
  - \* `id` : id dell'utente che modifica il numero.
  - \* `number` : vecchio numero da sostituire.
  - \* `newNumber` : nuovo numero da aggiungere.

### 3.58.2.5 shike::web::model::service::PoiService

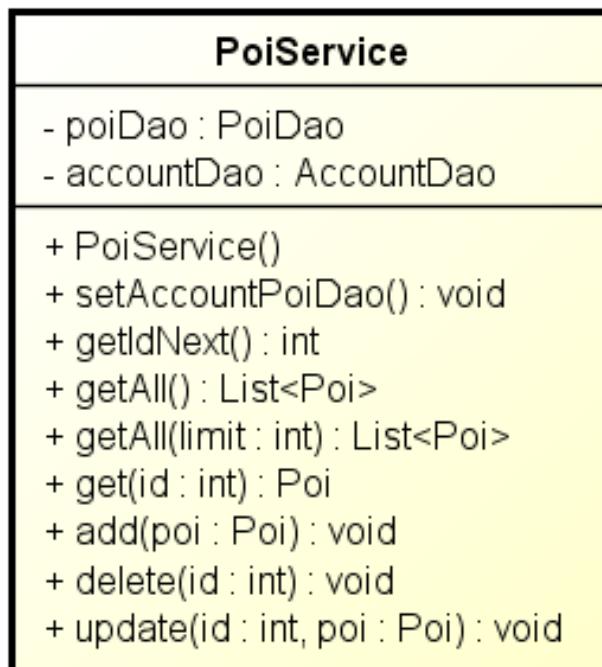


Figura 236: Diagramma di PoiService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *poi* tramite la classe DAO corrispondente.
- **Attributi:**
  - `-accountDao : AccountDao`  
DAO degli account.
  - `-poiDao : PoiDao`  
DAO dei POI.
- **Metodi:**
  - `+setAccountPoiDao( ) : void`  
Imposta il contesto dei DAO per poi inizializzarli con il Bean.
  - `+getAll( ) : List<Poi>`  
Permette di recuperare dal *database* tutti i POI.
  - `+getAll( limit : int ) : List<Poi>`  
Permette di recuperare dal *database* limit elementi POI.

**Argomenti:**

  - \* `limit` : numero di elementi da recuperare dal database.
  - `+getIdNext( ) : int`  
Ritorna l'id del prossimo POI che verrà inserito. Utile a livello di test.

- **+update( Poi : int, poi : Poi ) : void**  
Permette di aggiornare un POI.
 **Argomenti:**
  - \* **Poi** : id del poi da eliminare.
  - \* **poi** : nuovo poi che aggiorna il precedente.
- **+delete( id : int ) : void**  
Permette di eliminare un POI.
 **Argomenti:**
  - \* **id** : id del poi da eliminare.
- **+add( poi : Poi ) : void**  
Permette di aggiungere un POI.
 **Argomenti:**
  - \* **poi** : poi da aggiungere.
- **+get( id : int ) : Poi**  
Permette di recuperare dal *database* un POI. Tramite il DAO degli account permette di recuperare anche il nome di chi ha creato il POI.
 **Argomenti:**
  - \* **id** : id del poi richiesto.

### 3.58.2.6 shike::web::model::service::RecordedTrackService

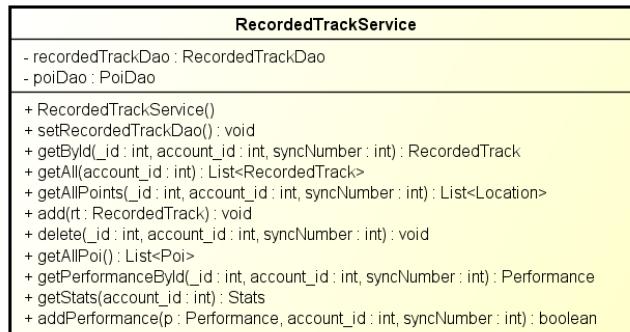


Figura 237: Diagramma di RecordedTrackService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *RecordedTrack* tramite la classe DAO corrispondente.
- **Attributi:**
  - **-recordedTrackDao : RecordedTrackDao**  
DAO dei RecordedTrack.
  - **-poiDao : PoiDao**  
DAO dei POI.
- **Metodi:**
  - **+setRecordedTrackDao( ) : void**  
Imposta il contesto dei DAO per poi inizializzarli con il Bean.

- `+getAllPoints( _id : int, account_id : int, syncNumber : int ) : List<Location>`

Restituisce tutti i punti di un tracciato selezionato.

**Argomenti:**

\* `_id` : id del tracciato.

\* `account_id` : id dell'utente che vuole il tracciato.

\* `syncNumber` : numero di sync.

- `+add( rt : RecordedTrack ) : void`

Aggiunge un tracciato.

**Argomenti:**

\* `rt` : tracciato da aggiungere.

- `+delete( _id : int, account_id : int, syncNumber : int ) : void`

Cancella il tracciato selezionato.

**Argomenti:**

\* `_id` : id del tracciato da cancellare.

\* `account_id` : id dell'utente che vuole cancellare il tracciato.

\* `syncNumber` : numero di sync.

- `+getById( _id : int, int : account_id, int : syncNumber ) : RecordedTrack`

Restituisce il tracciato con l'id selezionato.

**Argomenti:**

\* `_id` : id del tracciato da cancellare.

\* `int` : id dell'utente che vuole cancellare il tracciato.

\* `int` : numero di sync.

- `+getPerformanceById( _id : int, account_id : int, syncNumber : int ) : Performance`

Recupera le performance riguardanti un certo percorso.

**Argomenti:**

\* `_id` : id del tracciato del quale si vuole recuperare le performance.

\* `account_id` : id dell'utente.

\* `syncNumber` : numero di sync.

- `+getStats( _id : int ) : Stats`

Statistiche globali dell'utente selezionato.

**Argomenti:**

\* `_id` : id dell'utente.

### 3.59 shike::web::model::sync

#### 3.59.1 Informazioni sul package

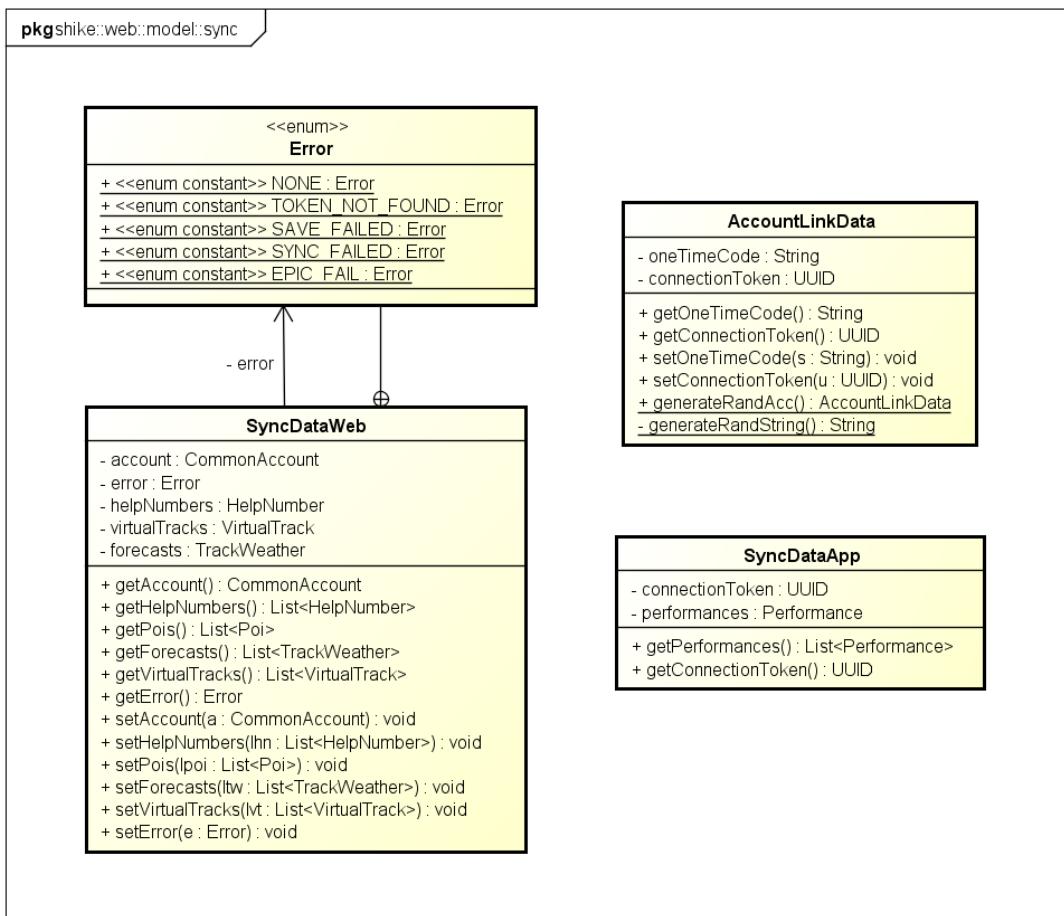


Figura 238: Diagramma di shike::web::model::sync

- **Descrizione:** componente che contiene le classi che modellano i dati che vengono scambiati durante la sincronizzazione e il collegamento account con il device.
- **Componente padre:** shike::web::model

### 3.59.2 Classi

#### 3.59.2.1 shike::web::model::sync::AccountLinkData

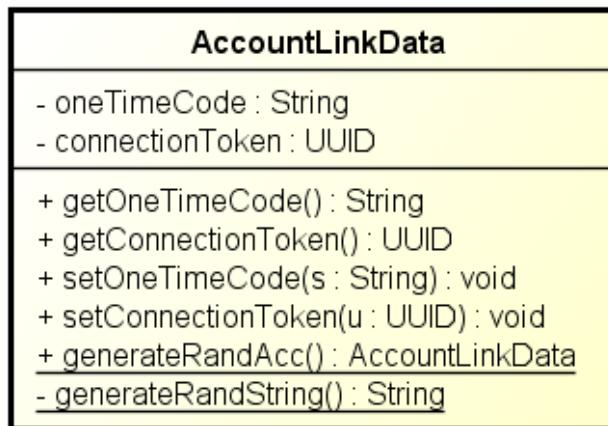


Figura 239: Diagramma di AccountLinkData

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati che sono inviati dalla piattaforma web per il collegamento del dispositivo all' *account*.
- **Attributi:**
  - **oneTimeCode** : String  
Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per effettuare il collegamento.
  - **Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento.** : UUID  
*Token* di connessione utilizzato dalla piattaforma web per identificare l'utente durante la procedura di sincronizzazione dati del device.
- **Metodi:**
  - **+getOneTimeCode() : String**  
Restituisce la variabile oneTimeCode.
  - **+setOneTimeCode( oneTimeCode : String ) : void**  
**Argomenti:**
    - \* **oneTimeCode** : valore da impostare nella variabile oneTimeCode.
  - **+getCodice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento.() : UUID**  
Restituisce la variabile Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento..
  - **+setCodice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento.( Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento. : UUID ) : void**  
**Argomenti:**
    - \* **Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento.** : UUID

- \* Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento. : valore da impostare nella variabile Codice monouso di 5 cifre che dev'essere immesso nella piattaforma web per completare il collegamento..
- [+generateRandAcc\(\)](#) : AccountLinkData  
Metodo che genera un nuovo AccountLinkData con parametri casuali.
- [-generateRandString\(\)](#) : String  
Ritorna una stringa di cinque numeri casuali che rappresentano il codice per collegare il device all'utente.

### 3.59.2.2 shike::web::model::sync::SyncDataApp

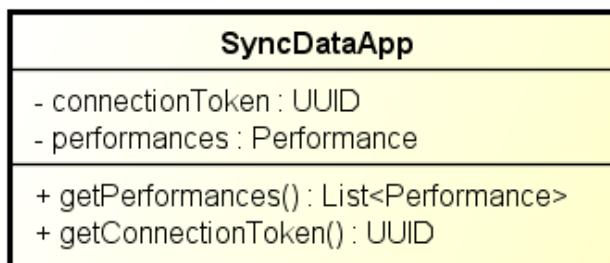


Figura 240: Diagramma di SyncDataApp

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati inviati dal device alla piattaforma web durante una sincronizzazione.
- **Attributi:**
  - [-performances](#) : List<Performance>  
Lista delle performance effettuate dall'ultima sincronizzazione.
  - [-connectionToken](#) : UUID  
Token di connessione univoco per l'utente.
- **Metodi:**
  - [+getPerformances\(\)](#) : List<Performance>  
Restituisce la variabile performances.
  - [+getConnectionToken\(\)](#) : UUID  
Restituisce la variabile connectionToken.

### 3.59.2.3 shike::web::model::sync::SyncDataWeb

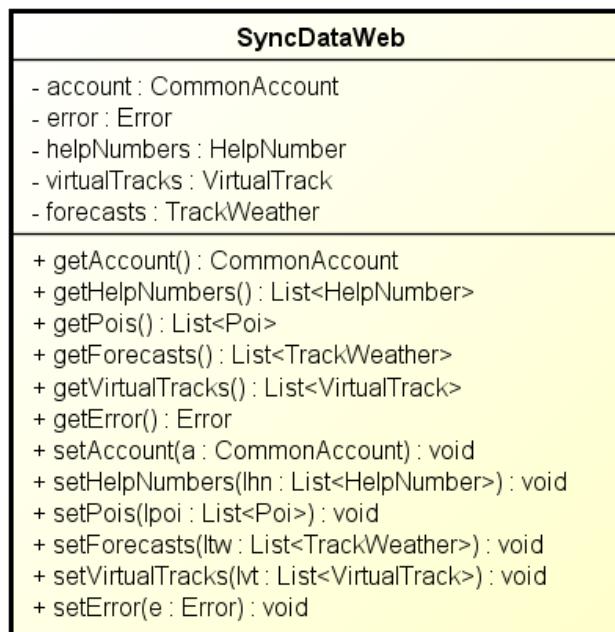


Figura 241: Diagramma di SyncDataWeb

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
- **Attributi:**
  - **error** : SyncDataWeb.Error  
Valore che indica se si sono verificati errori nella piattaforma web durante la sincronizzazione. Vale NONE se invece non si sono verificati errori.
  - **pois** : List<Poi>  
Lista dei POI aggiornati.
  - **account** : CommonAccount  
Informazioni aggiornate sull'account dell'utente.
  - **forecasts** : List<TrackWeather>  
Lista delle previsioni associate ai percorsi.
  - **virtualTracks** : List<VirtualTrack>  
Lista dei percorsi selezionati dall'utente e inviati al device.
  - **helpNumbers** : List<HelpNumber>  
Lista aggiornata dei numeri di soccorso dell'utente.
- **Metodi:**
  - **+getError()** : SyncDataWeb.Error  
Restituisce la variabile **error**.
  - **+setError( error : SyncDataWeb.Error ) : void**  
**Argomenti:**

\* `error` : valore da impostare nella variabile `error`.

- `+getPois() : List<Poi>`  
Restituisce la variabile `pois`.
- `+setPois( pois : List<Poi> ) : void`  
**Argomenti:**  
\* `pois` : valore da impostare nella variabile `pois`.
- `+getAccount() : CommonAccount`  
Restituisce la variabile `account`.
- `+setAccount( account : CommonAccount ) : void`  
**Argomenti:**  
\* `account` : valore da impostare nella variabile `account`.
- `+getForecasts() : List<TrackWeather>`  
Restituisce la variabile `forecasts`.
- `+setForecasts( forecasts : List<TrackWeather> ) : void`  
**Argomenti:**  
\* `forecasts` : valore da impostare nella variabile `forecasts`.
- `+getVirtualTracks() : List<VirtualTrack>`  
Restituisce la variabile `virtualTracks`.
- `+setVirtualTracks( virtualTracks : List<VirtualTrack> ) : void`  
**Argomenti:**  
\* `virtualTracks` : valore da impostare nella variabile `virtualTracks`.
- `+getHelpNumbers() : List<HelpNumber>`  
Restituisce la variabile `helpNumbers`.
- `+setHelpNumbers( helpNumbers : List<HelpNumber> ) : void`  
**Argomenti:**  
\* `helpNumbers` : valore da impostare nella variabile `helpNumbers`.

### 3.59.2.4 shike::web::model::sync::SyncDataWeb::Error

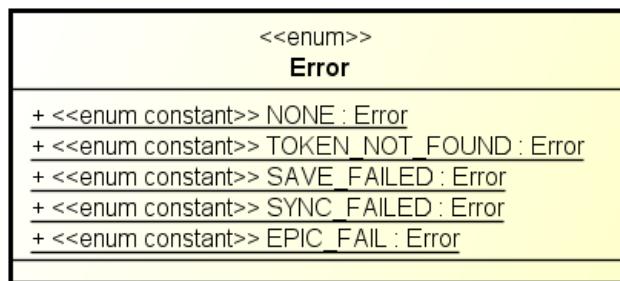


Figura 242: Diagramma di SyncDataWeb::Error

- **Tipo:** concreta
- **Descrizione:** enum contenente i possibili errori che la piattaforma web può dare di risposta all'applicazione.

## 4 Schema base di dati

Avendo scelto di utilizzare DBMS relazionali in entrambe le parti del prodotto, le basi di dati sono state implementate secondo i seguenti schemi. Tutti gli attributi che esprimono una data o un momento specifico nel tempo sono indicati in entrambe le basi in *Unix Epoch Time*, con precisione al millisecondo. Questa rappresentazione particolare è stata scelta in quanto congeniale all'ambiente di programmazione Java (nello specifico, la classe `java.util.Date` richiede nel suo costruttore a un parametro la data nello stesso formato). I diagrammi seguono il modello ER e utilizzano la notazione IDEF1X.

### 4.1 Parte applicazione

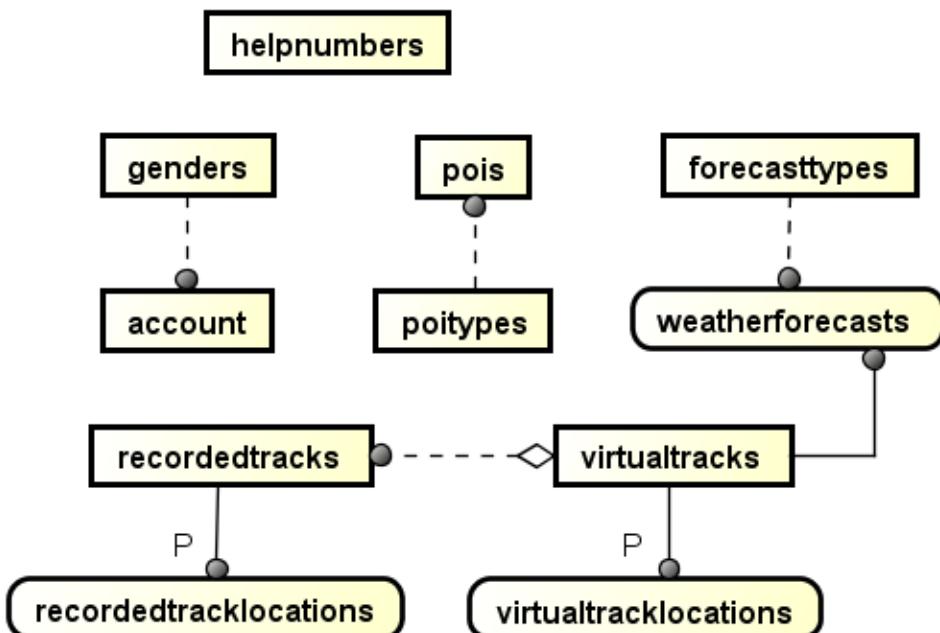


Figura 243: Schema logico della base di dati - Parte applicazione

#### 4.1.1 Lista delle tabelle

##### 4.1.1.1 account

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INTEGER	NO	PK	
connectionToken	TEXT	NO		
firstName	TEXT	NO		
lastName	TEXT	NO		
height	INTEGER	NO		Altezza in cm
weight	INTEGER	NO		Peso in g
gender_id	INTEGER	NO	FK(genders)	
birthDate	INTEGER	NO		

Tabella 2: Attributi tabella account - parte applicazione

Tabella *singleton* contenente l'unico *account* associato all'applicazione. *connectionToken* è il codice di connessione che l'applicazione utilizzerà per effettuare la sincronizzazione con il sito.

#### 4.1.1.2 genders

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INTEGER	NO	PK	
name	TEXT	NO		

Tabella 3: Attributi tabella genders - parte applicazione

Tabella statica contenente gli id di riferimento dei possibili sessi dell'utente. Gli id utilizzati seguono lo standard ISO 5218.

#### 4.1.1.3 helpnumbers

Attributo	Tipo	Nullable	Vincoli	Commento
number	TEXT	NO	PK	
name	TEXT	NO		

Tabella 4: Attributi tabella helpnumbers - parte applicazione

Tabella contenente i numeri di soccorso dell'utente.

#### 4.1.1.4 recordedtracks

Attributo	Tipo	Nullable	Vincoli	Commento
_id	TEXT	NO	PK	
virtual_id	TEXT	SI	FK(virtualtracks)	
date	INTEGER	NO		
distance	REAL	NO		Distanza in metri
time	INTEGER	NO		Tempo in ms
maxSpeed	REAL	NO		Velocità in m/s
steps	INTEGER	NO		

Tabella 5: Attributi tabella recordedtracks - parte applicazione

Tabella contenente le informazioni principali di un percorso creato nel dispositivo. Il percorso creato può avere un *VirtualTrack* associato (nel caso in cui l'utente decida di avviare una nuova sessione seguendo uno dei percorsi che ha scaricato) o no (nel caso in cui l'utente abbia avviato una sessione libera). Contiene inoltre le informazioni riguardanti le *performance* dell'utente durante la creazione del percorso.

#### 4.1.1.5 recordedtracklocations

Attributo	Tipo	Nullable	Vincoli	Commento
recordedTrack_id	INTEGER	NO	FK(recordedtracks), PK	
locationOrder	INTEGER	NO	PK	
latitude	REAL	NO		
longitude	REAL	NO		
altitude	REAL	NO		Altitudine in m

Tabella 6: Attributi tabella recordedtracklocations - parte applicazione

Tabella contenente tutti i punti che compongono i percorsi contenuti in *recordedtracks*. *locationOrder* indica l'ordine numerico del punto nel percorso di appartenenza.

#### 4.1.1.6 virtualtracks

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INTEGER	NO	PK	
name	TEXT	NO		
length	REAL	NO		Lunghezza in m
creationDate	INTEGER	NO		
barycenterLatitude	REAL	NO		
barycenterLongitude	REAL	NO		

Tabella 7: Attributi tabella virtualtracks - parte applicazione

Tabella contenente le informazioni sui percorsi che l'utente ha scaricato dalla piattaforma web sul dispositivo. *barycenterLongitude* e *barycenterLatitude* esprimono le coordinate del baricentro del percorso, che viene utilizzato per lo scaricamento delle condizioni meteo da internet.

#### 4.1.1.7 virtualtracklocations

Attributo	Tipo	Nullable	Vincoli	Commento
virtualTrack_id	INTEGER	NO	FK(virtualtracks), PK	
locationOrder	INTEGER	NO	PK	
latitude	REAL	NO		
longitude	REAL	NO		
altitude	REAL	NO		

Tabella 8: Attributi tabella virtualtracklocations - parte applicazione

Analogo di *recordedtracklocations* applicato ai percorsi in *virtualtracks*.

#### 4.1.1.8 weatherforecasts

Attributo	Tipo	Nullable	Vincoli	Commento
virtualTrack_id	INTEGER	NO	FK(virtualtracks), PK	
temperature	REAL	NO		Temperatura in K
pressure	REAL	NO		Pressione in hPa
date	INTEGER	NO	PK	
forecast_id	INTEGER	NO	FK(forecasttypes)	
humidity	REAL	NO		Umidità in %
windDirection	INTEGER	NO		Angolazione in gradi
windAvgSpeed	REAL	NO		Velocità in m/s
windMaxSpeed	REAL	NO		Velocità in m/s

Tabella 9: Attributi tabella weatherforecasts - parte applicazione

Tabella contenente le previsioni meteo riguardanti i percorsi scaricati dalla piattaforma web. La validità della previsione comincia dalla data indicata in **date** e finisce nella data della previsione immediatamente successiva.

#### 4.1.1.9 forecasttypes

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INTEGER	NO	PK	
name	TEXT	NO		

Tabella 10: Attributi tabella forecasttypes - parte applicazione

Tabella statica contenente tutte le tipologie di condizioni meteo previste, in accordanza con le specifiche delle API di *OpenWeatherMap*, reperibili nei riferimenti informativi di questo documento.

#### 4.1.1.10 pois

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INTEGER	NO	PK	
name	TEXT	NO		
type_id	INTEGER	NO	FK(poitypes)	
latitude	REAL	NO		
longitude	REAL	NO		
altitude	REAL	NO		Altitudine in m

Tabella 11: Attributi tabella pois - parte applicazione

Tabella che contiene le informazioni riguardanti i POI nelle vicinanze dei percorsi scaricati.

#### 4.1.1.11 poitypes

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INTEGER	NO	PK	
name	TEXT	NO		

Tabella 12: Attributi tabella poitypes - parte applicazione

Tabella statica che contiene le informazioni riguardanti le tipologie di POI. I valori scelti sono i seguenti:

ID	Nome
0	other
1	shelter
2	camping
3	water_point
4	panoramic_view
5	peak
6	hut

Tabella 13: Valori tabella poitypes

## 4.2 Parte web

NB: Le entità raffigurate in azzurro rappresentano delle viste.

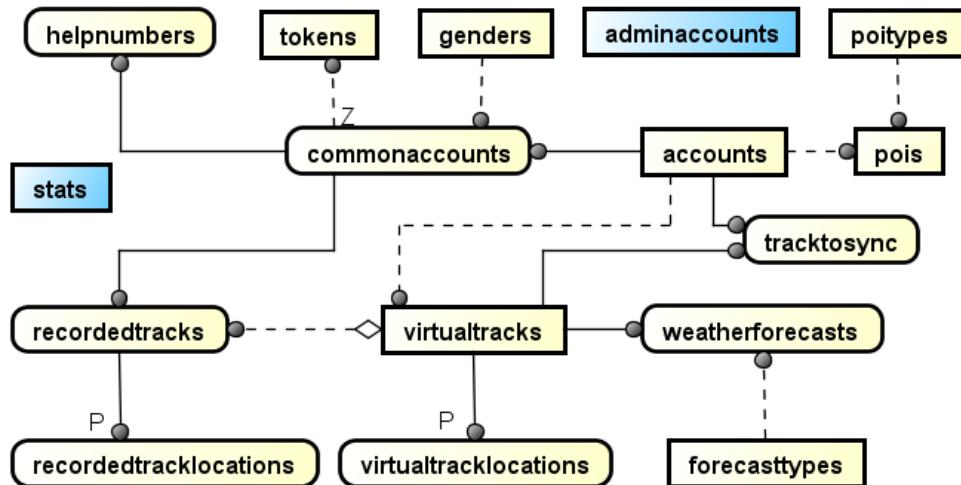


Figura 244: Schema logico della base di dati - Parte web

#### 4.2.1 Lista delle tabelle

##### 4.2.1.1 accounts

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	PK	
emailAddress	VARCHAR(254)	NO	UNIQUE	
passwordHash	BINARY(32)	NO		Hash SHA-256
role	VARCHAR(45)	NO		
enabled	TINYINT(1)	NO		

Tabella 14: Attributi tabella accounts - parte web

Tabella che contiene le credenziali di accesso di tutti gli *account* della piattaforma. `role` indica la classe di utenza di appartenenza dell'utente, necessaria per l'integrazione con l'autenticazione di Spring, mentre `enabled` indica se l'utente è attivo (ovvero può autenticarsi) o meno.

##### 4.2.1.2 commonaccounts

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	FK(accounts), PK	
syncCount	INT	NO		
firstName	VARCHAR(255)	NO		
lastName	VARCHAR(255)	NO		
height	SMALLINT(5)	NO		
weight	FLOAT	NO		
gender	TINYINT(1)	NO	FK(genders)	
birthDate	INT	NO		

Tabella 15: Attributi tabella commonaccounts - parte web

Tabella che contiene le informazioni degli account degli utenti comuni della piattaforma.

##### 4.2.1.3 adminaccounts

Vista che contiene le credenziali di tutti gli account degli amministratori del sito.

Query di creazione vista:

```
SELECT _id, emailAddress, passwordHash
FROM accounts NATURAL LEFT JOIN commonaccounts
WHERE firstName IS NULL
```

##### 4.2.1.4 genders

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	PK	
name	VARCHAR(50)	NO		

Tabella 16: Attributi tabella genders - parte web

Analogia alla tabella 3

#### 4.2.1.5 helpnumbers

Attributo	Tipo	Nullable	Vincoli	Commento
account_id	INT	NO	FK(commonaccounts), PK	
number	VARCHAR(50)	NO	PK	
description	VARCHAR(50)	NO		

Tabella 17: Attributi tabella helpnumbers - parte web

Tabella contenente i numeri di soccorso di tutti gli utenti comuni.

#### 4.2.1.6 recordedtracks

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	PK	
account_id	INT	NO	FK(commonaccounts), PK	
syncNumber	INT	NO	PK	
virtual_id	INT	SI	FK(virtualtracks)	
date	INT	NO		
distance	DOUBLE	NO		
time	BIGINT	NO		
maxSpeed	DOUBLE	NO		
heightDiff	DOUBLE	NO		
steps	INT	NO		

Tabella 18: Attributi tabella recordedtracks - parte web

Tabella analoga a Attributi tabella recordedtracks - parte applicazione, con l'aggiunta di syncNumber, che indica il numero di sincronizzazioni totali effettuate dall'utente al momento del caricamento del percorso sulla piattaforma, e di account\_id, chiave esterna che indica l'utente creatore della performance.

#### 4.2.1.7 recordedtracklocations

Attributo	Tipo	Nullable	Vincoli	Commento
track_id	INT	NO		
account_id	INT	NO	FK(recordedtracks), PK	
syncNumber	INT	NO		
locationOrder	INT	NO	PK	
latitude	DOUBLE	NO		
longitude	DOUBLE	NO		
altitude	DOUBLE	NO		

Tabella 19: Attributi tabella recordedtracklocations - parte web

Tabella contenente tutti i punti che compongono i percorsi contenuti in recordedtracks. recordedTrack\_id, account\_id e syncNumber appartengono ad un'unica chiave esterna riferita a recordedtracks.

#### 4.2.1.8 virtualtracks

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	PK	
author_id	INT	NO	FK(accounts)	
name	VARCHAR(255)	NO		
length	REAL	NO		
creationDate	BIGINT	NO		
heightDiff	INT	NO		

Tabella 20: Attributi tabella virtualtracks - parte web

Tabella contenente le informazioni sui percorsi pubblicati dagli utenti della piattaforma.

#### 4.2.1.9 virtualtracklocations

Attributo	Tipo	Nullable	Vincoli	Commento
track_id	INT	NO	FK(virtualtracks), PK	
locationOrder	INT	NO	PK	
latitude	DOUBLE	NO		
longitude	DOUBLE	NO		
altitude	DOUBLE	NO		

Tabella 21: Attributi tabella virtualtracklocations - parte web

Analogia alla tabella 8.

#### 4.2.1.10 weatherforecasts

Attributo	Tipo	Nullable	Vincoli	Commento
virtualTrack_id	INT	NO	FK(virtualtracks), PK	
temperature	DOUBLE	NO		
pressure	DOUBLE	NO		
date	BIGINT	NO	PK	
forecast_id	INT	NO	FK(forecasttypes)	
humidity	DOUBLE	NO		
windDirection	DOUBLE	NO		
windAvgSpeed	DOUBLE	NO		
windMaxSpeed	DOUBLE	NO		

Tabella 22: Attributi tabella weatherforecasts - parte web

Analogia alla tabella 9

#### 4.2.1.11 forecasttypes

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	PK	
name	VARCHAR(255)	NO		

Tabella 23: Attributi tabella forecasttypes - parte web

Analogia alla tabella 10

#### 4.2.1.12 stats

Vista che contiene le statistiche dell'attività fisica di ogni utente comune.

Query di creazione:

```
SELECT account_id,
       SUM(distance) AS totalDistance,
       SUM(time) AS totalTime,
       SUM(steps) AS totalSteps,
       SUM(heightDiff) AS totalHeightDiff,
       LOG10(1 + (2000 * SUM(distance) * GREATEST(1, SUM(heightDiff))
                  / GREATEST(1, SUM(time)))) AS expLevel
  FROM performance
 GROUP BY account_id
```

Il livello di esperienza dell'utente è calcolato secondo la seguente formula:

$$\text{expLevel} = \log_{10} \left( \frac{2000 * \text{totalDistance} * \max(1, \text{totalHeightDiff})}{\max(1, \text{totalTime})} + 1 \right)$$

Se il livello di esperienza è superiore a 3, allora l'utente è considerato esperto.

#### 4.2.1.13 pois

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	PK	
author_id	INT	NO	FK(accounts)	
name	VARCHAR(255)	NO		
type_id	INT	NO	FK(poitypes)	
latitude	DOUBLE	NO		
longitude	DOUBLE	NO		
altitude	DOUBLE	NO		

Tabella 24: Attributi tabella pois - parte web

Tabella che contiene le informazioni riguardanti tutti i POI creati.

#### 4.2.1.14 poitypes

Attributo	Tipo	Nullable	Vincoli	Commento
_id	INT	NO	PK	
name	VARCHAR(255)	NO		

Tabella 25: Attributi tabella poitypes - parte web

Analogia alla tabella 12.

#### 4.2.1.15 tokens

Attributo	Tipo	Nullable	Vincoli	Commento
token	VARCHAR(36)	NO	PK	
oneTimeCode	VARCHAR(5)	SI	UNIQUE	
account_id	INT	SI	FK(commonaccounts), UNIQUE	
timestamp	TIMESTAMP	NO		

Tabella 26: Attributi tabella tokens - parte web

Tabella che memorizza tutti i *token* di connessione all'applicazione Android. Un token può essere temporaneo, ovvero non essere collegato ad alcun account (*account\_id* è NULL), quando l'utente non ha ancora inserito il codice a 5 cifre (*oneTimeCode*) corrispondente nella sua pagina personale. Quando il collegamento è completo, il codice a 5 cifre viene eliminato, in modo da poter essere riutilizzato in futuro. *timestamp* indica la data di creazione del token, serve a calcolare quando cancellarlo nel caso non venga mai associato: se è più vecchio di 10 minuti viene eliminato in modo da non occupare inutilmente spazio e per renderlo riutilizzabile in seguito.

#### 4.2.1.16 tracktosync

Attributo	Tipo	Nullable	Vincoli	Commento
account_id	INT	NO	FK(commonaccounts), PK	
track_id	INT	NO	FK(virtualtracks), PK	

Tabella 27: Attributi tabella tracktosync - parte web

Tabella che memorizza tutti i percorsi che gli utenti hanno scelto di scaricare sul proprio dispositivo. Viene utilizzata durante la sincronizzazione per comporre la lista di percorsi da inviare al dispositivo richiedente.

## 5 Diagrammi di sequenza

### 5.1 Gestione di una richiesta di registrazione

Il diagramma in figura 245 illustra la gestione da parte del *server* di una richiesta di registrazione. La sequenza di azioni viene avviata quando un utente tramite un browser web invia una richiesta in “get” per ricevere il modulo di registrazione, operazione effettuata tramite il metodo *addAccount()*.

Una volta compilato il modulo, l’utente farà il *submit* dei dati inviando una richiesta in “post” contenente i campi compilati. La richiesta, passata tramite la funzione *addAccountConfirm()*, viene raccolta dall’*AccountController* il quale provvede a gestire i dati passandoli al validatore *CommonAccountValidator* che conferma che i campi inseriti siano validi. Fatto ciò il *controller* invoca il metodo *addCommonAccount()* il quale aggiunge i dati appena ricevuti come *account*. Se tutto si conclude in modo positivo il *controller* invierà un messaggio di successo alla *View* nella quale l’utente attendeva una risposta.

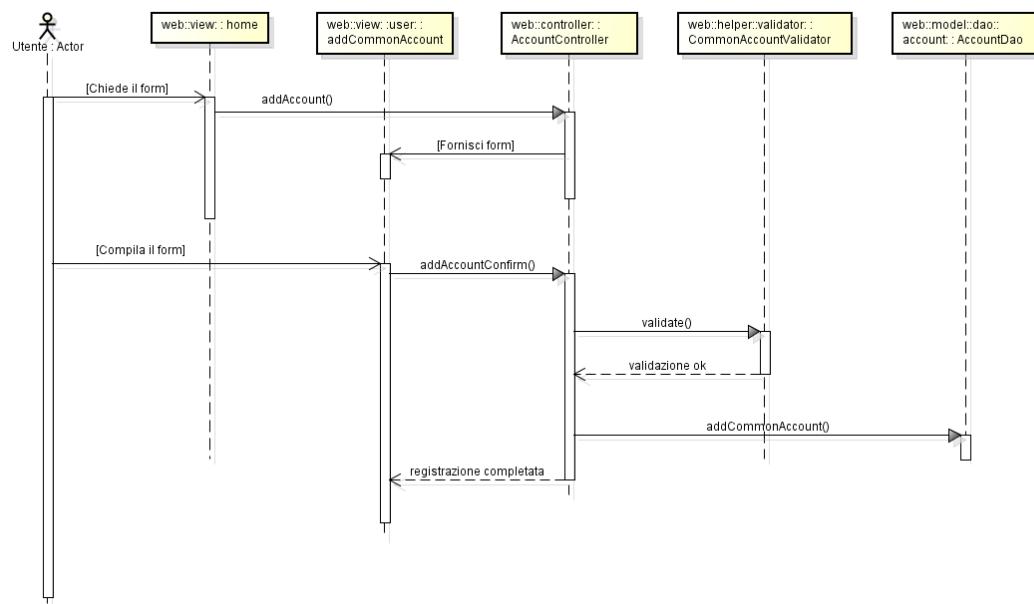


Figura 245: Gestione di una richiesta di registrazione

## 5.2 Gestione dei tracciati comuni a tutti gli utenti

Il diagramma in figura 246 illustra la gestione da parte del *server* della visualizzazione dei percorsi e dei relativi dettagli. La sequenza di azioni viene avviata quando un utente entra nella pagina di visualizzazione dei percorsi. Il *VirtualTrackController* riceve la richiesta e visualizza tramite il metodo *listOfTracks()* la lista dei *VirtualTrack* (ovvero dei percorsi comuni a tutti gli utenti). La *view Tracks* segnala al *controller* che la lista è stata visualizzata e l'utente riceve la notifica dal *controller*. Dalla lista dei percorsi visualizzata, l'utente può accedere ai dettagli dei vari percorsi. Il *controller* gestisce la richiesta e tramite il metodo *showTrack()* visualizza la lista dei dettagli tramite la vista *Track*. Qui verranno visualizzati la mappa e i dettagli. Fatto ciò la vista invia un messaggio al *controller* per confermare che la richiesta è stata eseguita. Il *controller* infine segnalera all'utente che l'operazione è stata eseguita e l'utente potrà visualizzare i dettagli del percorso.

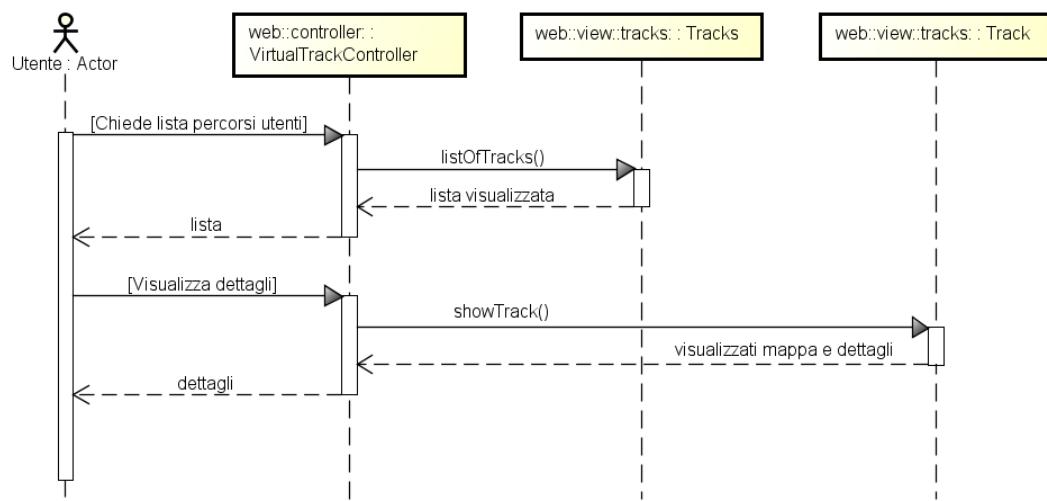


Figura 246: Gestione dei tracciati comuni a tutti gli utenti

### 5.3 Selezione percorso

Il diagramma in figura 247 illustra la selezione lato applicazione di un percorso partendo dalla *homepage* e arrivando alla *dashboard*. La sezione delle azioni descritte nell'immagine viene avviata quando l'utente è sulla *homepage* quindi si da già per fatta la parte di creazione di quest'ultima; l'utente a quel punto preme sul pulsante *Selezione percorso* tale pulsante associato alla funzione *onClick()* effettua un cambio di *Fragment* andando a creare tramite il metodo *onCreateView()* la vista in cui sono presenti tutti i tracciati salvati nel dispositivo, successivamente tramite i metodi *onActivityCreated()* e *setAdapter()* si adatta il *layout* della vista. L'utente a questo punto seleziona uno dei tracciati e tramite il metodo *onItemClick()* viene cambiato il *Fragment* creando tramite la funzione *onCreateView()* la vista che mostra le informazioni del singolo tracciato. L'utente per concludere clicca sul pulsante *Start* che associato al metodo *onClick()* va a creare l' *Activity* che gestisce la *dashboard*, tale *Activity* viene creata con il metodo *onCreate()*, successivamente adatta la *view* tramite il metodo *setAdapter()* e crea la *dashboard* tramite il metodo *onCreateView()*.

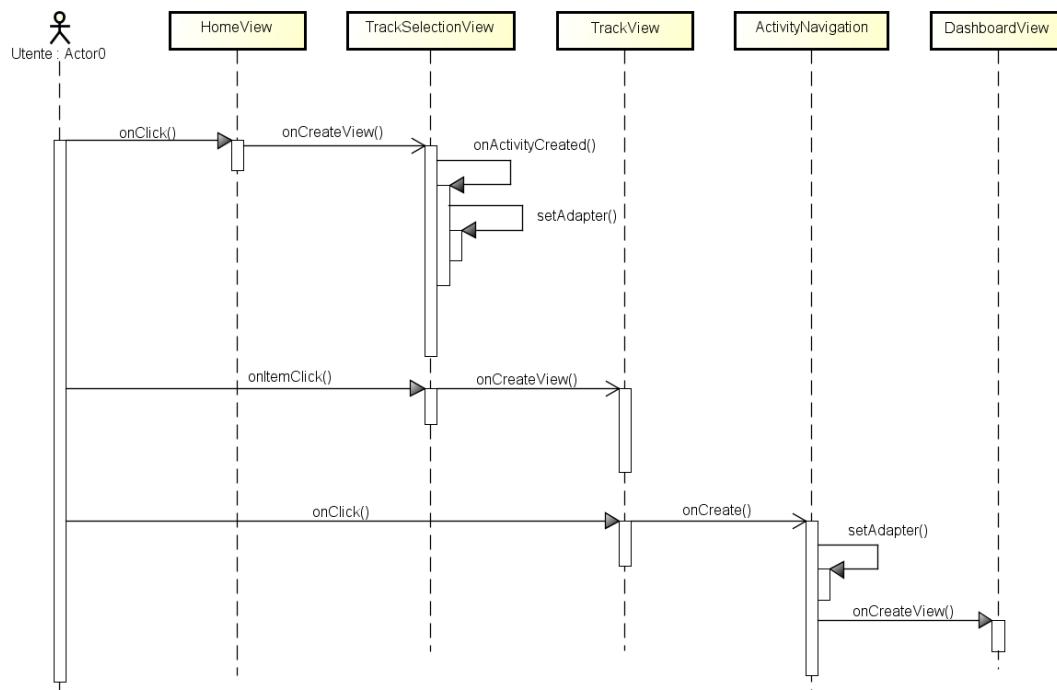


Figura 247: Gestisce la selezione di un percorso

## 6 Tracciamento