



Specifica Tecnica

Informazioni sul documento

Nome file:	specifica_tecnica_v3.0.pdf
Versione:	3.0
Data creazione:	2015-03-15
Data ultima modifica:	2015-06-16
Lista di distribuzione:	Prof. Tullio Vardanega Prof. Riccardo Cardin Si14 S.p.a.
Redattori:	Andrea Boscolo Nata Andrea Costa Tiziano Longo
Approvato da:	Mario Garavello
Verificatori:	Davide Canal
Stato:	Formale
Uso:	Esterno

Storia delle modifiche

Versione	Descrizione intervento	Autore	Ruolo	Data
3.0	Approvato	Mario Garavello	Responsabile	2015-06-16
2.4	Verifica	Davide Canal	Verificatore	2015-06-13
2.3	Ultimo importa dal sito Zeitrack	Andrea Boscolo Nata	Progettista	2015-06-08
2.2	Aggiornati Design Pattern	Tiziano Longo	Progettista	2015-06-05
2.1	Aggiunte tecnologie utilizzate	Andrea Costa	Progettista	2015-06-04
2.0	Approvato	Andrea Costa	Responsabile	2015-05-26
1.9	Verifica	Mario Garavello	Verificatore	2015-05-25
1.8	Ultimo importa dal sito Zeitrack	Tiziano Longo	Progettista	2015-05-25
1.7	Aggiornata Descrizione architettura	Tiziano Longo	Progettista	2015-05-22
1.6	Aggiornati Design Pattern - Parte Android	Luca Tiozzo Brasiola	Progettista	2015-05-19
1.5	Aggiornati Design Pattern - Parte Web	Tiziano Longo	Progettista	2015-05-18
1.4	Aggiornata tabella tracciamento	Luca Tiozzo Brasiola	Progettista	2015-05-15
1.3	Rivisti Design Pattern generali in appendice	Luca Tiozzo Brasiola	Progettista	2015-05-15
1.2	Corretti diagrammi UML	Luca Tiozzo Brasiola	Progettista	2015-05-14
1.1	Revisione indicazioni RP	Tiziano Longo	Progettista	2015-05-13
1.0	Approvazione	Jacopo Cavallarin	Responsabile	2015-04-14
0.8	Verifica	Andrea Boscolo Nata	Verificatore	2015-04-11
0.7	Aggiornata sezione Componenti e Classi	Luca Tiozzo Brasiola	Progettista	2015-04-09
0.6	Definizione Diagrammi di attività	Luca Tiozzo Brasiola	Progettista	2015-04-08
0.5	Definizione Design Pattern	Tiziano Longo	Progettista	2015-04-07
0.4	Definizione Descrizione generale Design Pattern	Tiziano Longo	Progettista	2015-04-07
0.3	Definizione Descrizione architettura	Andrea Costa	Progettista	2015-04-06
0.2	Definizione Tecnologie Utilizzate	Davide Canal	Progettista	2015-04-01
0.1	Scheletro documento	Davide Canal	Progettista	2015-03-15

Indice

1	Introduzione	1
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
2	Tecnologie Utilizzate	3
2.1	Spring Framework	3
2.2	JSON Schema	3
2.3	Android SDK	3
2.4	CSS3	4
2.5	HTML5	4
2.6	Gson	4
2.7	Jackson	5
3	Descrizione architettura	6
3.1	Metodo e formalismo di specifica	6
3.2	Architettura generale dell'applicazione mobile	6
3.2.1	Model	7
3.2.2	View	8
3.2.3	Presenter	9
3.2.4	Base di Dati	10
3.3	Architettura generale dell'applicazione web	11
3.3.1	Model	11
3.3.2	View	12
3.3.3	Controller	13
3.3.4	Base di Dati	14
4	Componenti e Classi	15
4.1	shike	15
4.1.1	Informazioni sulla componente	15
4.2	shike::app	16
4.2.1	Informazioni sulla componente	16
4.3	shike::app::presenter	17
4.3.1	Informazioni sulla componente	17
4.3.2	Classi	17
4.3.2.1	NavigationActivity	17
4.3.2.2	WeatherPresenter	17
4.3.2.3	DashboardPresenter	18
4.3.2.4	CompassPresenter	18
4.3.2.5	SyncPresenter	18
4.3.2.6	PoiPresenter	18
4.3.2.7	SettingPresenter	18
4.3.2.8	DashboardPresenter.TimerThread	18
4.3.2.9	HomeActivity	18
4.3.2.10	HomePresenter	19
4.3.2.11	NavigationActivity.MyPagerAdapter	19
4.3.2.12	SyncPresenter	19
4.3.2.13	DashboardPresenter.ButtonThread	19

4.3.2.14	HelpNumberPresenter	19
4.4	shike::app::model	20
4.4.1	Informazioni sulla componente	20
4.5	shike::app::model::weather	21
4.5.1	Informazioni sulla componente	21
4.5.2	Classi	22
4.5.2.1	Weather	22
4.5.2.2	TrackWeather	22
4.5.2.3	Weather.ForecastType	22
4.5.2.4	Weather.Wind	22
4.6	shike::app::model::session	23
4.6.1	Informazioni sulla componente	23
4.7	shike::app::model::session::track	24
4.7.1	Informazioni sulla componente	24
4.7.2	Classi	24
4.7.2.1	Poi	24
4.7.2.2	Track	25
4.7.2.3	VirtualTrack	25
4.7.2.4	RecordedTrack	25
4.7.2.5	Poi.PoiType	25
4.8	shike::app::model::session::performance	26
4.8.1	Informazioni sulla componente	26
4.8.2	Classi	26
4.8.2.1	Performance	26
4.9	shike::app::model::user	27
4.9.1	Informazioni sulla componente	27
4.9.2	Classi	27
4.9.2.1	Account	27
4.9.2.2	HelpNumber	28
4.9.2.3	Account.Gender	28
4.10	shike::app::model::dao	29
4.10.1	Informazioni sulla componente	29
4.10.2	Classi	30
4.10.2.1	GeneralDao	30
4.10.2.2	GeneralDaoImpl	30
4.11	shike::app::model::dao::account	31
4.11.1	Informazioni sulla componente	31
4.11.2	Classi	31
4.11.2.1	AccountDao	31
4.11.2.2	AccountDaoImpl	31
4.12	shike::app::model::dao::helpnumber	32
4.12.1	Informazioni sulla componente	32
4.12.2	Classi	32
4.12.2.1	HelpNumberDao	32
4.12.2.2	HelpNumberDaoImpl	33
4.13	shike::app::model::dao::performance	34
4.13.1	Informazioni sulla componente	34
4.13.2	Classi	34
4.13.2.1	PerformanceDaoImpl	34
4.13.2.2	PerformanceDao	35
4.13.2.3	PerformanceDaoImpl	35
4.13.2.4	PerformanceDao	35

4.14 shike::app::model::dao::poi	36
4.14.1 Informazioni sulla componente	36
4.14.2 Classi	36
4.14.2.1 PoiDaoImpl	36
4.14.2.2 PoiDao	37
4.15 shike::app::model::dao::track	38
4.15.1 Informazioni sulla componente	38
4.15.2 Classi	38
4.15.2.1 VirtualTrackDaoImpl	38
4.15.2.2 VirtualTrackDao	39
4.16 shike::app::model::dao::weather	39
4.16.1 Informazioni sulla componente	39
4.16.2 Classi	40
4.16.2.1 TrackWeatherDao	40
4.16.2.2 TrackWeatherDaoImpl	40
4.17 shike::app::model::dao::db	41
4.17.1 Informazioni sulla componente	41
4.17.2 Classi	41
4.17.2.1 AccountTable	41
4.17.2.2 GendersTable	42
4.17.2.3 DbContract	42
4.17.2.4 DbUtil	43
4.17.2.5 HelpNumbersTable	43
4.17.2.6 RecordedTracksTable	44
4.17.2.7 RecordedTrackLocationsTable	44
4.17.2.8 VirtualTracksTable	44
4.17.2.9 VirtualTrackLocationsTable	44
4.17.2.10 WeatherForecastsTable	45
4.17.2.11 ForecastTypesTable	45
4.17.2.12 PoisTable	45
4.17.2.13 PoiTypesTable	45
4.18 shike::app::model::service	46
4.18.1 Informazioni sulla componente	46
4.18.2 Classi	46
4.18.2.1 AccountService	46
4.18.2.2 GeneralService	47
4.18.2.3 HelpNumberService	47
4.18.2.4 PerformanceService	47
4.18.2.5 PoiService	48
4.18.2.6 TrackWeatherService	48
4.18.2.7 VirtualTrackService	48
4.19 shike::app::model::sync	49
4.19.1 Informazioni sulla componente	49
4.19.2 Classi	49
4.19.2.1 SyncDataApp	49
4.19.2.2 AccountLinkData	50
4.19.2.3 SyncDataWeb	50
4.19.2.4 SyncDataWeb.Error	50
4.20 shike::app::view	51
4.20.1 Informazioni sulla componente	51
4.21 shike::app::view::config	52
4.21.1 Informazioni sulla componente	52

4.21.2 Classi	52
4.21.2.1 ConfigView	52
4.21.2.2 SyncNumberDialog	52
4.22 shike::app::view::helpnumber	53
4.22.1 Informazioni sulla componente	53
4.22.2 Classi	53
4.22.2.1 HelpNumberSelectionView	53
4.22.2.2 HelpNumberViewAdapter	54
4.23 shike::app::view::home	54
4.23.1 Informazioni sulla componente	54
4.23.2 Classi	55
4.23.2.1 TrackView	55
4.23.2.2 HomeView	55
4.23.2.3 ListViewAdapter	55
4.23.2.4 TrackSelectionView	56
4.24 shike::app::view::poi	56
4.24.1 Informazioni sulla componente	56
4.24.2 Classi	56
4.24.2.1 PoiSelectionView	56
4.24.2.2 PoiView	57
4.24.2.3 PoiViewAdapter	57
4.25 shike::app::view::session	58
4.25.1 Informazioni sulla componente	58
4.25.2 Classi	58
4.25.2.1 DashboardView	58
4.25.2.2 CompassView	58
4.25.2.3 WidgetSelectionView	58
4.25.2.4 DialogConfirmSaveTrack	59
4.26 shike::app::view::weather	59
4.26.1 Informazioni sulla componente	59
4.26.2 Classi	59
4.26.2.1 WeatherView	59
4.26.2.2 WeatherSelectionView	60
4.26.2.3 WeatherViewAdapter	60
4.27 shike::app::helper	60
4.27.1 Informazioni sulla componente	60
4.27.2 Classi	61
4.27.2.1 WebConnectionManager	61
4.27.2.2 WebConnectionManager.Result	62
4.28 shike::app::helper::json	62
4.28.1 Informazioni sulla componente	62
4.28.2 Classi	62
4.28.2.1 JsonDateSerializer	62
4.28.2.2 JsonDateDeserializer	63
4.28.2.3 JsonConverter	63
4.28.2.4 JsonLocationSerializer	63
4.28.2.5 JsonLocationDeserializer	64
4.29 shike::web	64
4.29.1 Informazioni sulla componente	64
4.30 shike::web::controller	65
4.30.1 Informazioni sulla componente	65
4.30.2 Classi	65

4.30.2.1	MainController	65
4.30.2.2	RecordedTrackController	66
4.30.2.3	PoiController	66
4.30.2.4	AuthController	66
4.30.2.5	Account Controller	67
4.30.2.6	SyncController	67
4.30.2.7	HelpNumberController	68
4.30.2.8	VirtualTrackController	68
4.31	shike::web::view	69
4.31.1	Informazioni sulla componente	69
4.31.2	Classi	69
4.31.2.1	Home	69
4.32	shike::web::view::account	70
4.32.1	Informazioni sulla componente	70
4.32.2	Classi	71
4.32.2.1	ResetPassword	71
4.32.2.2	Login	71
4.32.2.3	AddCommonAccount	71
4.32.2.4	EditPassword	72
4.33	shike::web::view::account::user	72
4.33.1	Informazioni sulla componente	72
4.33.2	Classi	72
4.33.2.1	Dashboard	72
4.33.2.2	AddHelpNumber	73
4.33.2.3	EditHelpNumber	73
4.33.2.4	EditUser	73
4.33.2.5	ListHelpNumbers	74
4.34	shike::web::view::account::admin	74
4.34.1	Informazioni sulla componente	74
4.34.2	Classi	75
4.34.2.1	Dashboard	75
4.34.2.2	ListCommonAccount	75
4.35	shike::web::view::track	76
4.35.1	Informazioni sulla componente	76
4.35.2	Classi	76
4.35.2.1	Stats	76
4.36	shike::web::view::track::poi	77
4.36.1	Informazioni sulla componente	77
4.36.2	Classi	77
4.36.2.1	DetailsPoi	77
4.36.2.2	AddPoi	78
4.36.2.3	ListPoi	78
4.37	shike::web::view::track::virtual	79
4.37.1	Informazioni sulla componente	79
4.37.2	Classi	79
4.37.2.1	ListTrack	79
4.37.2.2	EditTrack	79
4.37.2.3	DetailsTrack	80
4.38	shike::web::view::track::recorded	80
4.38.1	Informazioni sulla componente	80
4.38.2	Classi	81
4.38.2.1	DetailsTrack	81

4.38.2.2	ShareTrack	81
4.38.2.3	ListTrack	81
4.39	shike::web::view::inc	82
4.39.1	Informazioni sulla componente	82
4.39.2	Classi	82
4.39.2.1	Footer	82
4.39.2.2	NotFound	83
4.39.2.3	Header	83
4.39.2.4	Menu	84
4.39.2.5	Message	85
4.40	shike::web::helper	86
4.40.1	Informazioni sulla componente	86
4.40.2	Classi	86
4.40.2.1	MailManager	86
4.41	shike::web::helper::sharing	87
4.41.1	Informazioni sulla componente	87
4.41.2	Classi	87
4.41.2.1	Share	87
4.41.2.2	ShareFacebook	88
4.41.2.3	ShareTwitter	88
4.42	shike::web::helper::validator	88
4.42.1	Informazioni sulla componente	88
4.42.2	Classi	89
4.42.2.1	HelpNumberValidator	89
4.42.2.2	CommonAccountValidator	89
4.43	shike::web::model	89
4.43.1	Informazioni sulla componente	89
4.43.2	Classi	90
4.43.2.1	Message.Alert	90
4.43.2.2	Message	90
4.44	shike::web::model::weather	91
4.44.1	Informazioni sulla componente	91
4.44.2	Classi	91
4.44.2.1	TrackWeather	91
4.44.2.2	Weather	92
4.44.2.3	Weather.Wind	92
4.44.2.4	Weather.ForecastType	92
4.45	shike::web::model::user	93
4.45.1	Informazioni sulla componente	93
4.45.2	Classi	93
4.45.2.1	Account	93
4.45.2.2	CommonAccount	94
4.45.2.3	HelpNumber	94
4.45.2.4	CommonAccount.Gender	95
4.46	shike::web::model::session	95
4.46.1	Informazioni sulla componente	95
4.47	shike::web::model::session::track	96
4.47.1	Informazioni sulla componente	96
4.47.2	Classi	96
4.47.2.1	Track	96
4.47.2.2	RecordedTrack	97
4.47.2.3	VirtualTrack	97

4.47.2.4	Poi	98
4.47.2.5	Location	98
4.47.2.6	Poi.PoiType	98
4.47.2.7	VirtualTrack.Level	99
4.48	shike::web::model::session::performance	99
4.48.1	Informazioni sulla componente	99
4.48.2	Classi	99
4.48.2.1	Performance	99
4.48.2.2	Stats	100
4.49	shike::web::model::dao	100
4.49.1	Informazioni sulla componente	100
4.50	shike::web::model::dao::account	102
4.50.1	Informazioni sulla componente	102
4.50.2	Classi	103
4.50.2.1	CommonAccountMapper	103
4.50.2.2	AccountDao	103
4.50.2.3	AccountMapper	103
4.50.2.4	AccountDaoImpl	104
4.51	shike::web::model::dao::helpnumber	104
4.51.1	Informazioni sulla componente	104
4.51.2	Classi	105
4.51.2.1	HelpNumberDao	105
4.51.2.2	HelpNumberMapper	105
4.51.2.3	HelpNumberDaoImpl	105
4.52	shike::web::model::dao::performance	106
4.52.1	Informazioni sulla componente	106
4.53	shike::web::model::dao::poi	106
4.53.1	Informazioni sulla componente	106
4.53.2	Classi	107
4.53.2.1	PoiDao	107
4.53.2.2	PoiMapper	107
4.53.2.3	PoiDaoImpl	107
4.54	shike::web::model::dao::recordtrack	108
4.54.1	Informazioni sulla componente	108
4.54.2	Classi	108
4.54.2.1	RecordedTrackDao	108
4.54.2.2	PerformanceMapper	109
4.54.2.3	RecordedTrackDaoImpl	109
4.54.2.4	RecordedTrackMapper	109
4.54.2.5	RecordedTrackLocationMapper	110
4.54.2.6	StatsMapper	110
4.55	shike::web::model::dao::weather	111
4.55.1	Informazioni sulla componente	111
4.55.2	Classi	111
4.55.2.1	WeatherDao	111
4.55.2.2	WeatherMapper	111
4.55.2.3	WeatherDaoImpl	112
4.56	shike::web::model::dao::virtualtrack	112
4.56.1	Informazioni sulla componente	112
4.56.2	Classi	113
4.56.2.1	VirtualTrackDao	113
4.56.2.2	VirtualTrackMapper	113

4.56.2.3	VirtualTrackLocationMapper	113
4.56.2.4	VirtualTrackDaoImpl	114
4.57	shike::web::model::service	115
4.57.1	Informazioni sulla componente	115
4.57.2	Classi	115
4.57.2.1	VirtualTrackService	115
4.57.2.2	WeatherService	115
4.57.2.3	AccountService	116
4.57.2.4	HelpNumberService	116
4.57.2.5	PoiService	116
4.57.2.6	RecordedTrackService	117
4.58	shike::web::model::sync	117
4.58.1	Informazioni sulla componente	117
4.58.2	Classi	118
4.58.2.1	AccountLinkData	118
4.58.2.2	SyncDataApp	118
4.58.2.3	SyncDataWeb	118
4.58.2.4	SyncDataWeb.Error	119
5	Diagrammi di attività	120
5.1	Applicazione mobile	120
5.1.1	Attività principali dell’utente	120
5.1.2	Sessione utente con percorso	121
5.1.3	Sessione utente libera	122
5.1.4	Visualizza dashboard	123
5.2	Applicazione web	124
5.2.1	Registrazione utente	124
5.2.2	Attività principali dell’utente	125
5.2.3	Attività principali dell’amministratore della piattaforma	126
5.2.4	Gestione dei Percorsi	127
5.2.5	Gestione dei POI	128
6	Design Pattern	129
6.1	Design Pattern architettoniali	129
6.1.1	MVC, Model-View-Controller	129
6.1.2	MVP, Model-View-Presenter	129
6.1.3	DAO, Data Access Object	130
6.2	Design Pattern creazionali	131
6.2.1	Singleton	131
6.3	Design Pattern comportamentali	131
6.3.1	Strategy	131
7	Stime di fattibilità e di bisogno di risorse	133
8	Tracciamento	134
8.1	Tabella delle associazioni Requisiti-Componenti	134
8.2	Tabella delle associazioni Componenti-Requisiti	140
A	Descrizione generale Design Pattern	144
A.1	Design Pattern architettoniali	144
A.1.1	MVC, Model-View-Controller	144
A.1.2	DAO, Data Access Object	145
A.2	Design Pattern creazionali	145

A.2.1	Singleton	145
A.3	Design Pattern comportamentali	146
A.3.1	Strategy	146

Elenco delle figure

1	Diagrammi delle classi parte applicazione - Model	7
2	Diagrammi delle classi parte applicazione - View	8
3	Diagrammi delle classi parte applicazione - Presenter	9
4	Schema concettuale base di dati - parte applicazione	10
5	Diagrammi delle classi parte web - Model	11
6	Diagrammi delle classi parte web - View	12
7	Diagrammi delle classi parte web - Controller	13
8	Schema concettuale base di dati - parte web	14
9	Diagramma di shike	15
10	Diagramma di shike::app	16
11	Diagramma di shike::app::presenter	17
12	Diagramma di shike::app::model	20
13	Diagramma di shike::app::model::weather	21
14	Diagramma di shike::app::model::session	23
15	Diagramma di shike::app::model::session::track	24
16	Diagramma di shike::app::model::session::performance	26
17	Diagramma di shike::app::model::user	27
18	Diagramma di shike::app::model::dao	29
19	Diagramma di shike::app::model::dao::account	31
20	Diagramma di shike::app::model::dao::helpnumber	32
21	Diagramma di shike::app::model::dao::performance	34
22	Diagramma di shike::app::model::dao::poi	36
23	Diagramma di shike::app::model::dao::track	38
24	Diagramma di shike::app::model::dao::weather	39
25	Diagramma di shike::app::model::dao::db	41
26	Diagramma di shike::app::model::service	46
27	Diagramma di shike::app::model::sync	49
28	Diagramma di shike::app::view	51
29	Diagramma di shike::app::view::config	52
30	Diagramma di shike::app::view::helpnumber	53
31	Diagramma di shike::app::view::home	54
32	Diagramma di shike::app::view::poi	56
33	Diagramma di shike::app::view::session	58
34	Diagramma di shike::app::view::weather	59
35	Diagramma di shike::app::helper	60
36	Diagramma di shike::app::helper::json	62
37	Diagramma di shike::web	64
38	Diagramma di shike::web::controller	65
39	Diagramma di shike::web::view	69
40	Diagramma di shike::web::view::account	70
41	Diagramma di shike::web::view::account::user	72
42	Diagramma di shike::web::view::account::admin	74
43	Diagramma di shike::web::view::track	76
44	Diagramma di shike::web::view::track::poi	77
45	Diagramma di shike::web::view::track::virtual	79
46	Diagramma di shike::web::view::track::recorded	80
47	Diagramma di shike::web::view::inc	82
48	Diagramma di shike::web::helper	86
49	Diagramma di shike::web::helper::sharing	87
50	Diagramma di shike::web::helper::validator	88

51	Diagramma di shike::web::model	89
52	Diagramma di shike::web::model::weather	91
53	Diagramma di shike::web::model::user	93
54	Diagramma di shike::web::model::session	95
55	Diagramma di shike::web::model::session::track	96
56	Diagramma di shike::web::model::session::performance	99
57	Diagramma di shike::web::model::dao	100
58	Diagramma di shike::web::model::dao::account	102
59	Diagramma di shike::web::model::dao::helpnumber	104
60	Diagramma di shike::web::model::dao::poi	106
61	Diagramma di shike::web::model::dao::recordtrack	108
62	Diagramma di shike::web::model::dao::weather	111
63	Diagramma di shike::web::model::dao::virtualtrack	112
64	Diagramma di shike::web::model::service	115
65	Diagramma di shike::web::model::sync	117
66	Diagramma delle attività principali dell’utente	120
67	Diagramma per l’avvio di una nuova sessione con percorso	121
68	Diagramma per l’avvio di una nuova sessione libera	122
69	Diagramma per la gestione della dashboard	123
70	Diagramma per la registrazione di un nuovo utente	124
71	Diagramma delle attività principali dell’utente	125
72	Diagramma delle attività principali dell’amministratore della piattaforma	126
73	Diagramma per la gestione dei Percorsi	127
74	Diagramma per la gestione dei POI	128
75	Diagramma del Design Pattern Model View Controller	129
76	Diagramma del Design Pattern Model View Presenter	129
77	Diagramma del Design Pattern Data Access Object	130
78	Diagramma del Design Pattern Singleton	131
79	Diagramma del Design Pattern Strategy	131
80	Diagramma generico del Design Pattern MVC	144
81	Diagramma generico del Design Pattern Data Access Object	145
82	Diagramma generico del Design Pattern Singleton	145
83	Diagramma generico del Design Pattern Strategy	146

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello del progetto sHike. Viene presentata l'architettura generale secondo la quale saranno organizzate le varie componenti software e saranno descritti i Design Pattern utilizzati. Inoltre, viene presentato il tracciamento tra le componenti e i requisiti individuati.

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

- **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;

- **Android 4.4 APIs - API Level: 19:** <http://developer.android.com/reference/packages.html>;
- **SQLite Documentation:** <https://www.sqlite.org/docs.html>;
- **Design Patterns: Elementi per il riuso di software a oggetti:** - E. Gamma, R. Helm, R. Johnson, J. Vlissides (Pearson, 2002);
- **Software Engineering - Chapter 6: Architectural design:** - Ian Sommerville - 9th ed. (2010);
- **Librerie Gson:** <https://code.google.com/p/google-gson/>.

2 Tecnologie Utilizzate

2.1 Spring Framework

L'utilizzo del framework Spring è vincolato dal capitolato come descritto dal requisito RWVO 5, esso è un *framework open source* per lo sviluppo di applicazioni su piattaforma Java.

Vantaggi

- Architettura a livelli, che permette di scegliere quale dei suoi componenti usare fornendo allo stesso tempo un *framework* coesivo per lo sviluppo di applicazioni distribuite;
- Non serve imparare niente di nuovo, riutilizza già molte tecnologie
- Inversione di controllo;
- Modularità, facile capire che classi/*package* servono e quali possono essere ignorati;

Svantaggi

- Può essere difficile dialogare con altre componenti di Spring;
- Complesso: contiene oltre 2400 classi;
- Obbligatorio l'utilizzo intensivo di XML;
- I molti meccanismi di parallelizzazione rischiano di aumentare la complessità dello sviluppo.

2.2 JSON Schema

L'utilizzo di JSON Schema è vincolato dal requisito RTVO 1 come richiesto dal proponente nel capitolato. JSON Schema serve a descrivere la struttura di un file JSON che è un formato adatto per lo scambio dei dati in applicazioni *client-server* nato come alternativa a XML.

Vantaggi

- Permette di risparmiare banda rispetto a XML;
- Viene processato molto velocemente da un computer;
- Supportato da moltissimi linguaggi (portabile).

Svantaggi

- La sua semplicità è anche la sua debolezza, in quanto nel caso si debbano trasportare informazioni complesse e strutturate potrebbe essere necessario del lavoro aggiuntivo per la gestione.

2.3 Android SDK

L'utilizzo del SDK di Android è vincolato dal capitolato come descritto dal requisito RAVO 3. Android SDK include molti strumenti utili allo sviluppatore quali librerie e un IDE comprensivo di debugger, oltre ad una documentazione esauriente.

Vantaggi

- Lo sviluppo dell'applicazione risulta più semplice;

- Rispetto al NDK non richiede allo sviluppatore di occuparsi della gestione della memoria;
- Contiene un debugger, le librerie, un emulatore per qualsiasi tipo di dispositivo Android.

Svantaggi

- Pur basandosi su Java non utilizza la sua macchina virtuale (minor portabilità);
- L'ambiente di sviluppo seppur completo risulta molto pesante anche in un computer moderno.

2.4 CSS3

L'utilizzo di CSS3 è vincolato dal requisito RWQO 1.2. Si è deciso di utilizzare CSS come linguaggio per la formattazione dello stile delle pagine web HTML che compongono il sito.

Vantaggi

- Consente di separare la struttura del sito dall'aspetto grafico;
- La manutenzione della grafica risulta molto semplice.

2.5 HTML5

L'utilizzo di HTML5 è vincolato dal requisito RWQO 1.1. Si è deciso di utilizzare HTML5 come linguaggio di *markup* per la strutturazione delle pagine web.

Vantaggi

- Permette di definire semplicemente le pagine web in modo tale che si adattino ai vari tipi di browser;
- Non necessita l'uso di *plugin* esterni.

Svantaggi

- Il suo supporto non è ancora completo nei principali browser.

2.6 Gson

Per la conversione di oggetti Java da/verso JSON nella parte app si è deciso di usare la libreria Gson di *Google*.

Vantaggi

- Conversione immediata con l'invocazione di un unico metodo (`toJson` e `fromJson`).
- La conversione avviene anche se la classe di partenza e quella di destinazione hanno attributi differenti (gli attributi diversi tra le due vengono inizializzati al valore di default).

Svantaggi

- Alcuni tipi di dato necessitano di serializzatori e deserializzatori personalizzati.
- Gli oggetti convertiti da JSON potrebbero trovarsi in uno stato non consistente (alcuni attributi potrebbero avere valori non previsti dall'implementazione della classe).

2.7 Jackson

Per la conversione di oggetti Java da/verso JSON nella parte web si è deciso di usare la libreria Open Source Jackson.

Vantaggi

- La libreria è de facto lo standard per la serializzazione JSON nelle applicazioni Spring.
- Permetta la configurazione degli attributi tramiti semplici *annotation*.

Svantaggi

- L'alta automatizzazione fornita dalla libreria richiede di porre attenzione nella creazione delle classi per evitare di serializzare informazioni sensibili.

3 Descrizione architettura

3.1 Metodo e formalismo di specifica

L'esposizione dell'architettura del progetto viene esposta tramite un approccio *top-down* sia per la parte web sia per la parte dell'applicazione, iniziando a descrivere l'architettura dal generale ed andando verso il particolare.

Si procede descrivendo le componenti, per ognuna di esse si analizzano le classi che la compongono specificando per ognuna il tipo, l'obiettivo, la funzione e le relazioni in ingresso e in uscita. Successivamente si descrivono i design pattern utilizzati per la realizzazione dell'architettura, la spiegazione dei design pattern è riportata nell'appendice Design Pattern.

I diagrammi delle componenti, delle classi, di sequenza e delle attività rispettano il formalismo di 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 il colore **verde**. Le classi e le componenti dell'applicazione sono invece raffigurati con il colore giallo.

L'architettura generale non descrive nel complesso tutte le sottoclassi del sistema, infatti viene spiegato solamente lo scopo di una gerarchia e le sue relazioni con altre componenti, la spiegazione dettagliata delle sottoclassi e di attributi e metodi viene effettuata nella **Progettazione di Dettaglio**.

Successivamente vengono descritte le due parti che compongono il progetto: l'applicazione Android e l'applicazione web.

3.2 Architettura generale dell'applicazione mobile

L'architettura generale dell'applicazione Android segue il *pattern MVP* suddividendosi quindi in tre parti: *Model*, *View* e *Presenter*. Una quarta parte è la Base di Dati, in cui vengono memorizzati in maniera persistente i dati.

3.2.1 Model

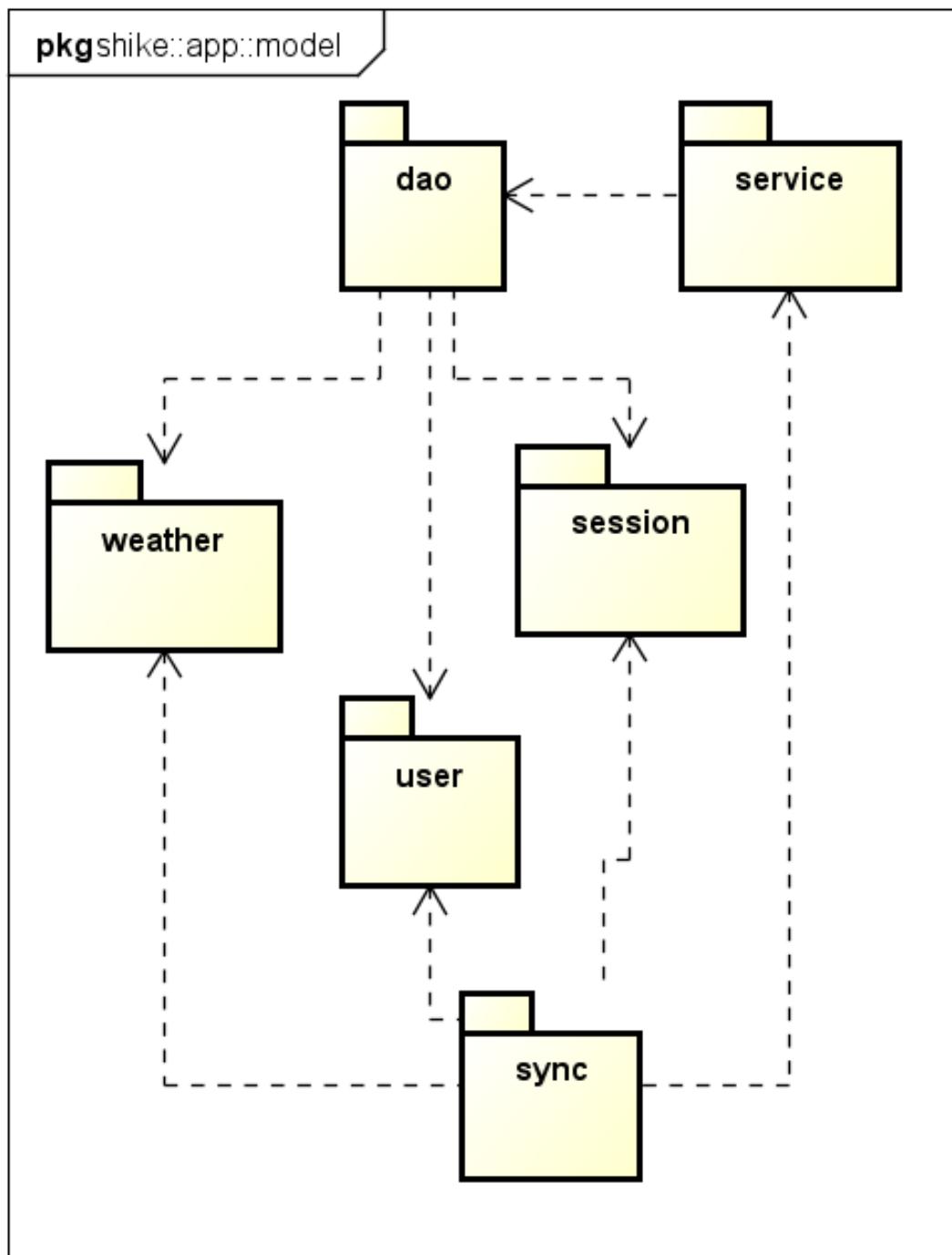


Figura 1: Diagrammi delle classi parte applicazione - Model

Nel diagramma sopra riportato sono presenti tutte le classi di cui è composto il Model, esse descrivono:

- la composizione di un modello;
- la strutturazione di un percorso;
- le informazioni visualizzabili sul display;
- le informazioni sulle statistiche generali;

La componente Model mette inoltre a disposizione tutti i metodi per modificare gli oggetti in esso contenuti; si occupa di inviare segnali alla View quando vi è una modifica nei dati. Se vi è un errore questo viene gestito dal Model e segnalato alla View.

3.2.2 View

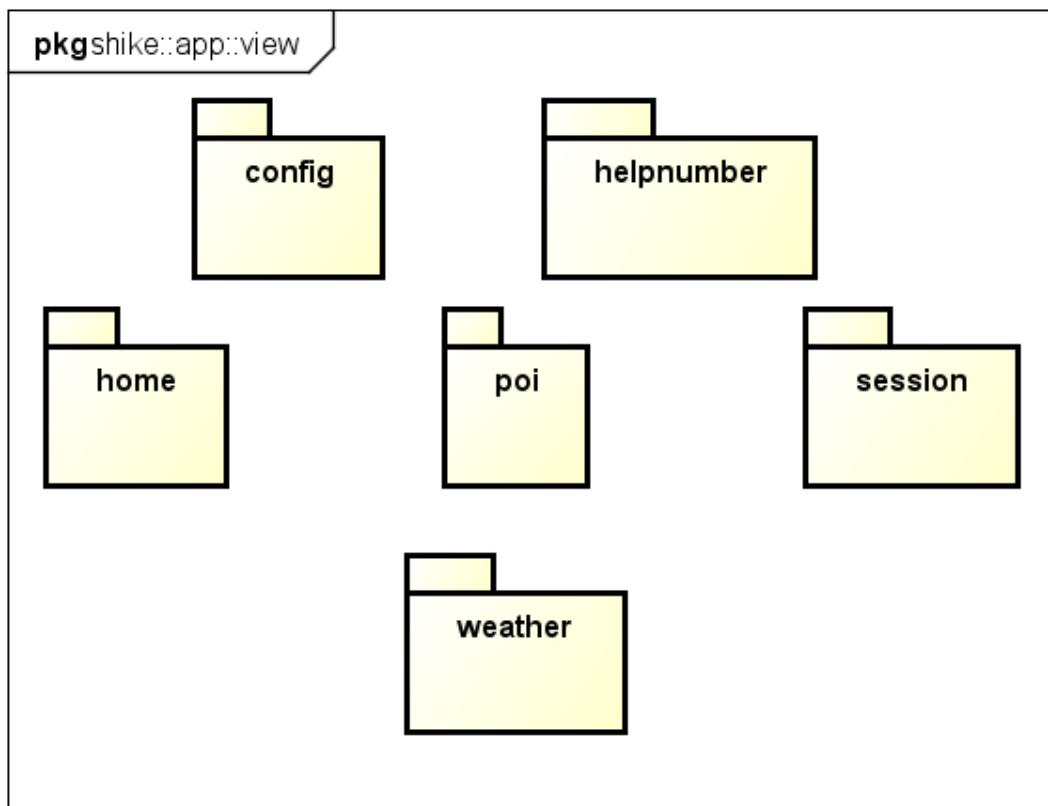


Figura 2: Diagrammi delle classi parte applicazione - View

Nella View sono presenti tutti fragments che vengono mostrati nel display, fornisce inoltre i widget che posso essere visualizzati nella dashboard. Le componenti che permettono di modificare i dati del Model, tramite la View, invoca i metodi del Presenter il quale si occupa a sua volta di invocare i metodi presenti nel Model per effettuare la modifica. Una volta modificati i dati il Model si occupa di inviare un segnale alla View, le cui componenti successivamente si occupano di prendere i dati aggiornati e visualizzarli.

3.2.3 Presenter

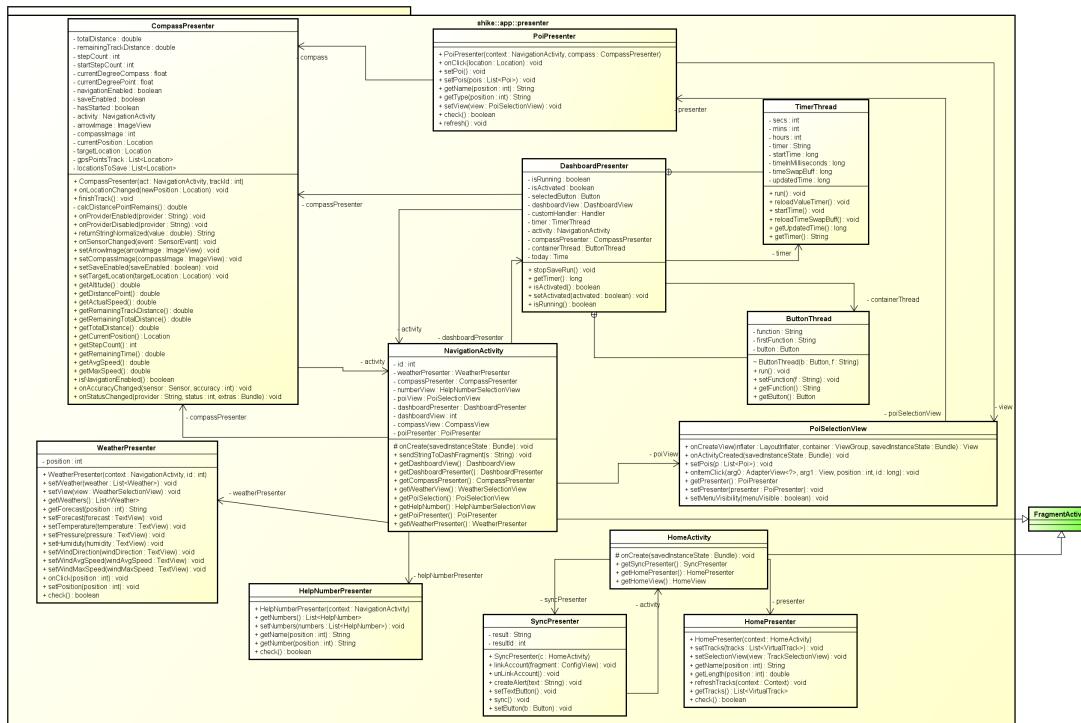


Figura 3: Diagrammi delle classi parte applicazione - Presenter

Nel Presenter sono presenti le classi che permettono di collaborare sia con la View che con il Model:

- **View**: il Presenter si occupa di visualizzare nuovi fragment in base a ciò che viene comunicato dalla View;
- **Model**: il Presenter si occupa di invocare i comandi da inviare al Model;

3.2.4 Base di Dati

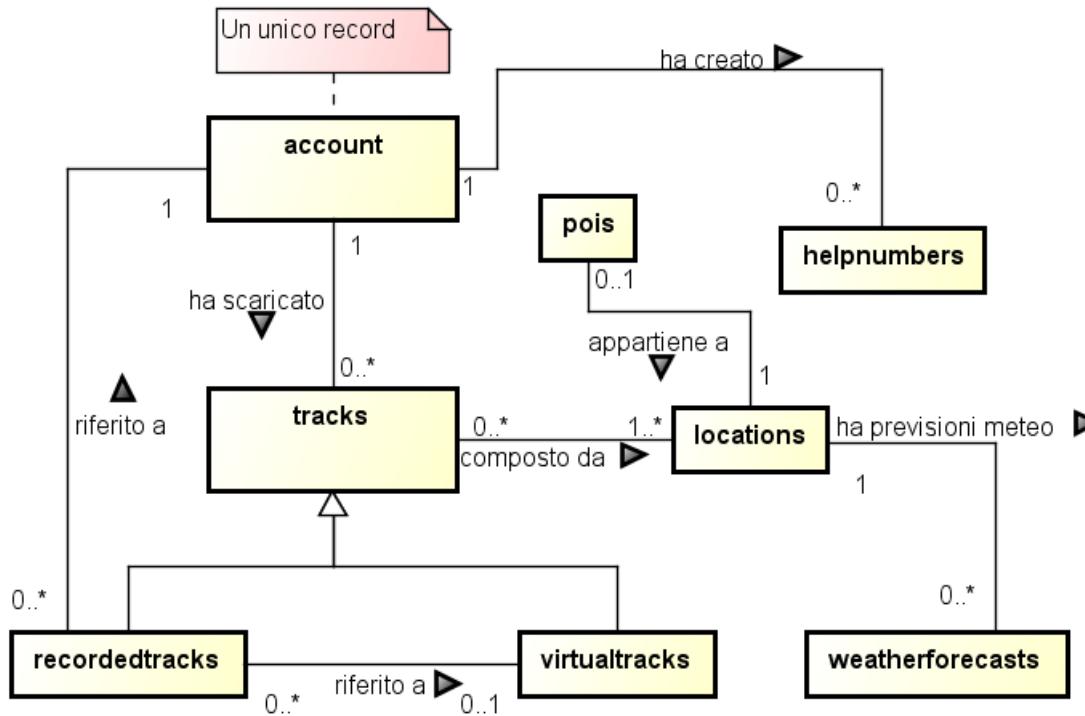


Figura 4: Schema concettuale base di dati - parte applicazione

La base di dati è organizzata nelle seguenti classi:

- **account**: modella l'account dell'utente collegato all'applicazione;
- **tracks**: modella i percorsi che l'utente ha scaricato dalla piattaforma web o che ha creato dall'ultima sincronizzazione;
 - **virtualtracks**: modella un percorso scaricato dalla piattaforma web;
 - **recordedtracks**: modella un percorso effettuato dall'utente (può essere sia nuovo che riferito ad uno scaricato), con tutte le relative statistiche (come tempo totale, velocità media,...);
- **helpnumbers**: modella un numero di soccorso creato dall'utente;
- **locations**: modella un punto geografico;
- **pois**: modella un POI;
- **weatherforecasts**: modella una previsione meteo per un punto geografico;

3.3 Architettura generale dell'applicazione web

L'architettura generale dell'applicazione web segue il pattern MVC suddividendosi quindi in tre parti: Model, View e Controller. Anche in questo caso è presente la Base di Dati, per la memorizzazione persistente dei dati.

3.3.1 Model

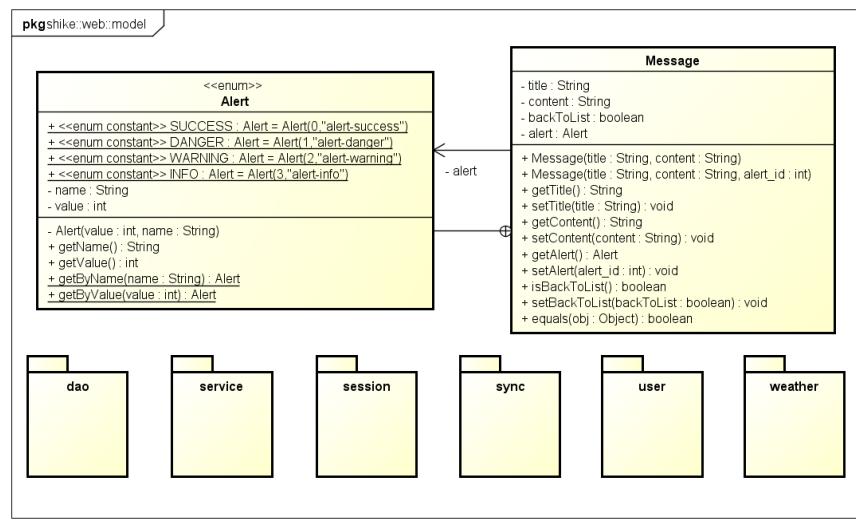


Figura 5: Diagrammi delle classi parte web - Model

Nel diagramma sopra riportato sono presenti tutte le classi di cui è composto il Model della parte web, esse descrivono:

- la composizione di un main(contiene tutti i dati di un utente);
- la raccolta di tutti i dati all'interno della classe data(utenti, percorsi, POI)
- la strutturazione di un percorso;
- le informazioni sulle statistiche generali;
- la possibilità di condividere una performance;

La componente Model mette inoltre a disposizione tutti i metodi per modificare gli oggetti in esso contenuti; si occupa di inviare segnali alla View quando vi è una modifica nei dati. Se vi è un errore questo viene gestito dal Model e segnalato alla View.

3.3.2 View

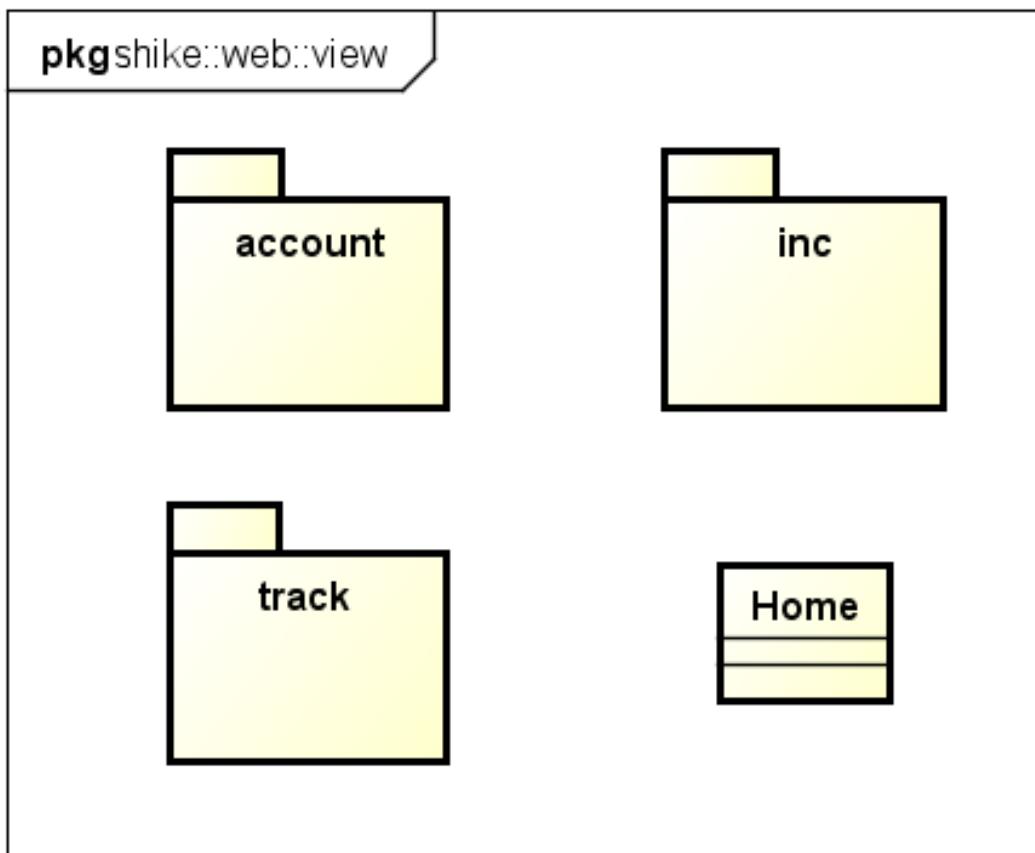


Figura 6: Diagrammi delle classi parte web - View

Nella View sono presenti tutte le classi che serviranno per la visualizzazione dei dati. Le componenti che permettono di modificare i dati del Model, tramite la View, mandano segnali al Controller il quale si occupa di invocare i metodi presenti nel Model per effettuare la modifica. Una volta modificati i dati il Model si occupa di inviare un segnale alla View, le cui componenti successivamente si occupano di prendere i dati aggiornati e visualizzarli.

3.3.3 Controller

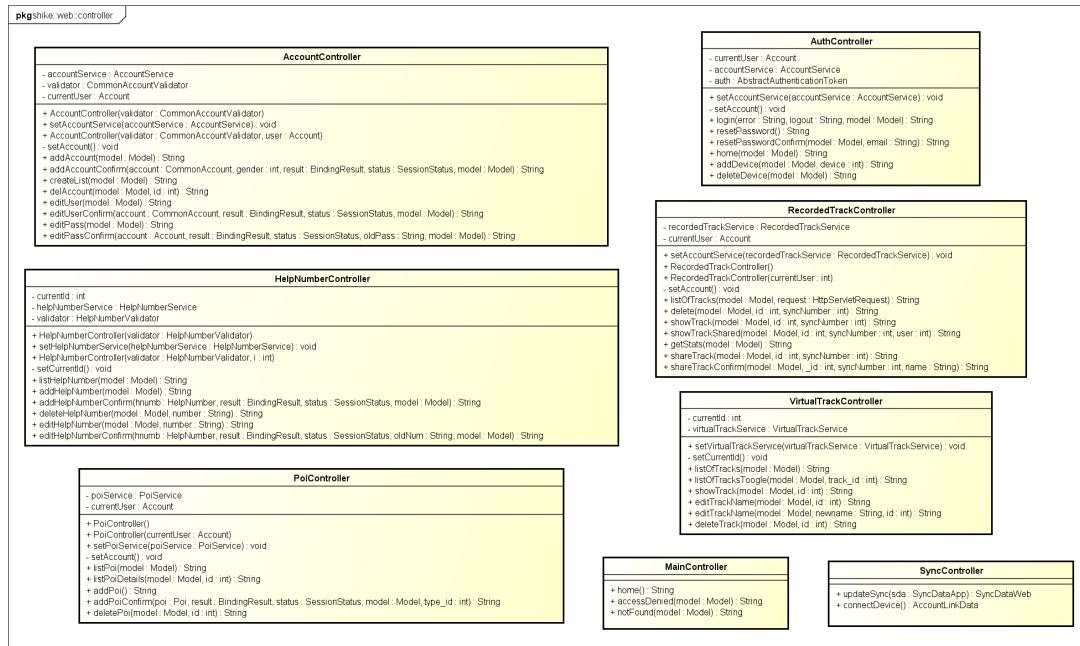


Figura 7: Diagrammi delle classi parte web - Controller

Nel Controller è presente la classe che riceve tutti i segnali inviati dalla View e successivamente si occupa di decidere che operazioni effettuare:

- **View:** il Controller può agire sulla View e visualizzare nuove finestre;
- **Model:** il Controller si occupa di invocare i comandi da inviare al Model;

3.3.4 Base di Dati

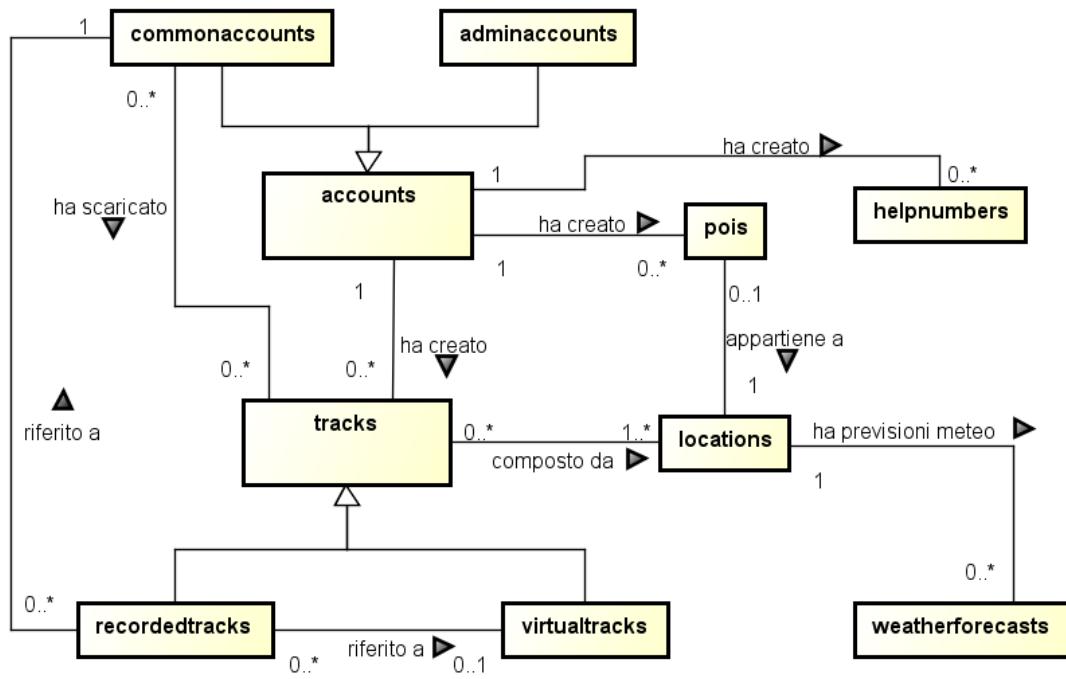


Figura 8: Schema concettuale base di dati - parte web

La base di dati è organizzata nelle seguenti classi:

- **accounts**: modella un account utente registrato alla piattaforma;
 - **adminaccounts**: modella un account amministratore del sito;
 - **commonaccounts**: modella un account utente comune;
- **tracks**: modella un percorso qualsiasi;
 - **virtualtracks**: modella un percorso pubblicato da un utente;
 - **recordedtracks**: modella un percorso effettuato da un utente, con relative statistiche. Può essere sia nuovo, quindi non ancora pubblicato, sia riferito ad uno già esistente.
- **helpnumbers**, **locations**, **pois**, **weatherforecasts**: identiche alla parte applicazione.

4 Componenti e Classi

Per indicare le relazioni tra classi si utilizza la seguente notazione: \leftarrow per indicare una relazione entrante nella classe corrente, \rightarrow per una relazione uscente.

4.1 shike

4.1.1 Informazioni sulla componente

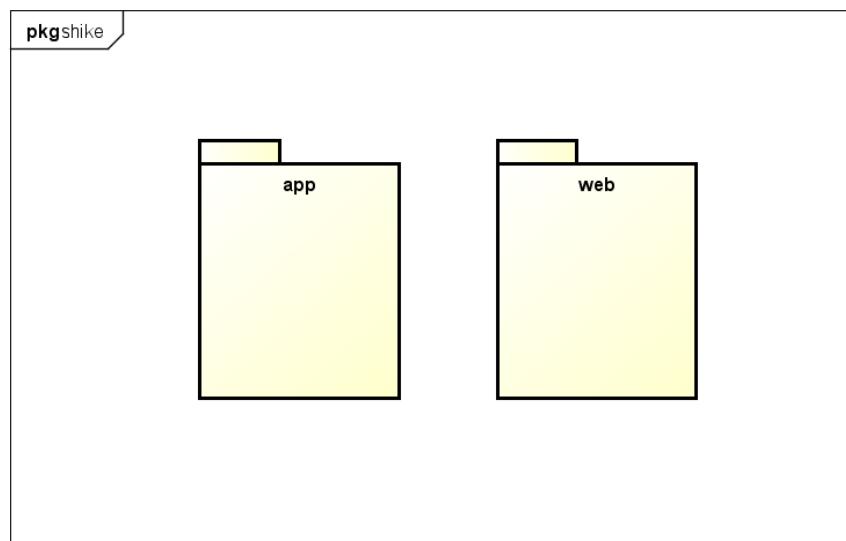


Figura 9: Diagramma di shike

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

4.2 shike::app

4.2.1 Informazioni sulla componente

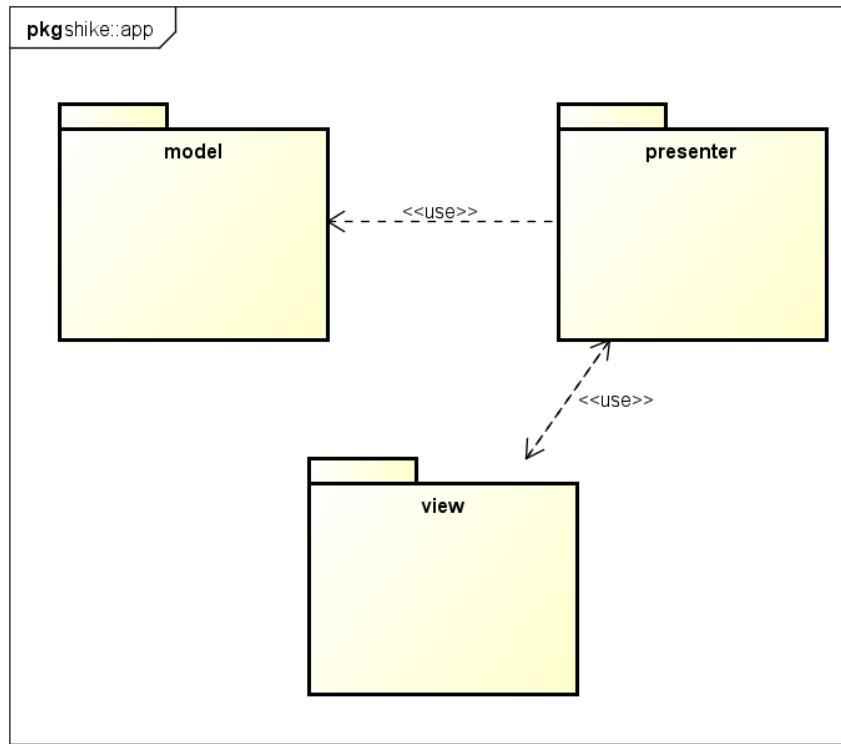


Figura 10: Diagramma di shike::app

- **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

4.3 shike::app::presenter

4.3.1 Informazioni sulla componente

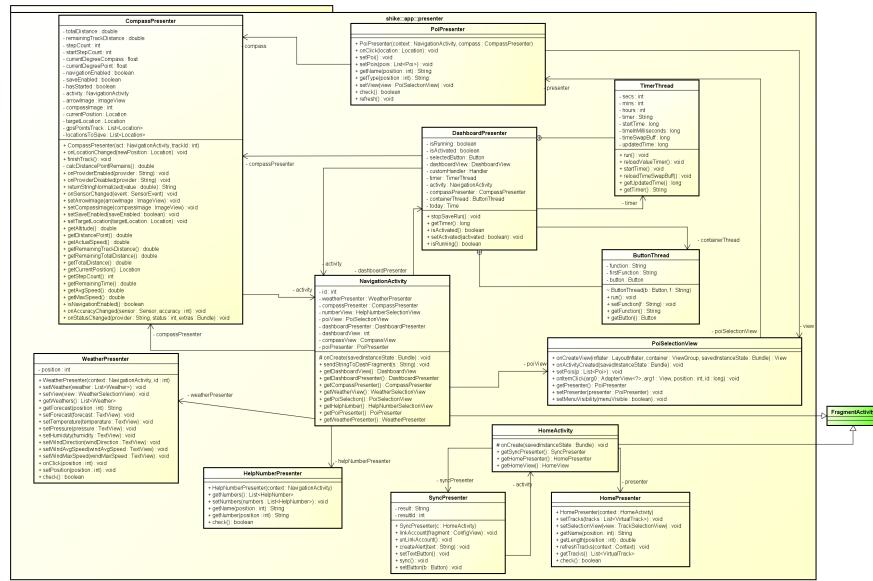


Figura 11: Diagramma di shike::app::presenter

- **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

4.3.2 Classi

4.3.2.1 shike::app::presenter::NavigationActivity

- **Tipo:** concreta
 - **Descrizione:** *View* che gestisce le viste della navigazione.
 - **Superclassi:**
 - android.support.v4.app.FragmentActivity

4.3.2.2 shike::app::presenter::WeatherPresenter

- **Tipo:** concreta
 - **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate per gestire le informazioni sul meteo.
 - **Relazioni con altre classi**
 - → WeatherSelectionView: classe che modella la visualizzazione della lista delle in-

4.3.2.3 shike::app::presenter::DashboardPresenter

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

4.3.2.4 shike::app::presenter::CompassPresenter

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

4.3.2.5 shike::app::presenter::SyncPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate nella sincronizzazione.

4.3.2.6 shike::app::presenter::PoiPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dei POI.
- **Relazioni con altre classi**
 - → PoiSelectionView: classe che modella la *view* per la visualizzazione e la possibile selezione dei POI da parte dell'utente.

4.3.2.7 shike::app::presenter::SettingPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate nella modifica delle impostazioni.

4.3.2.8 shike::app::presenter::DashboardPresenter.TimerThread

- **Tipo:** concreta
- **Descrizione:** *Thread* che si occupa di gestire il *timer*.
- **Implementa:**
 - java.lang.Runnable

4.3.2.9 shike::app::presenter::HomeActivity

- **Tipo:** concreta
- **Descrizione:** classe che gestisce le viste da istanziare in *homepage*.
- **Superclassi:**
 - android.support.v4.app.FragmentActivity

4.3.2.10 shike::app::presenter::HomePresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dalla *home-page*.
- **Relazioni con altre classi**
 - → HomeView: classe che rappresenta la visualizzazione della pagina principale dell'applicazione

4.3.2.11 shike::app::presenter::NavigationActivity.MyPagerAdapter

- **Tipo:** concreta
- **Descrizione:** classe che modella i *Fragment* all'interno della classe *ActivityNavigation*.
- **Superclassi:**
 - android.support.v4.app.FragmentPagerAdapter

4.3.2.12 shike::app::presenter::SyncPresenter

- **Tipo:** concreta
- **Descrizione:** classe che gestisce la logica della sincronizzazione con il *server web*.

4.3.2.13 shike::app::presenter::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

4.3.2.14 shike::app::presenter::HelpNumberPresenter

- **Tipo:** concreta
- **Descrizione:** classe che fa da *presenter* per tutte le funzionalità utilizzate dai numeri di soccorso.

4.4 shike::app::model

4.4.1 Informazioni sulla componente

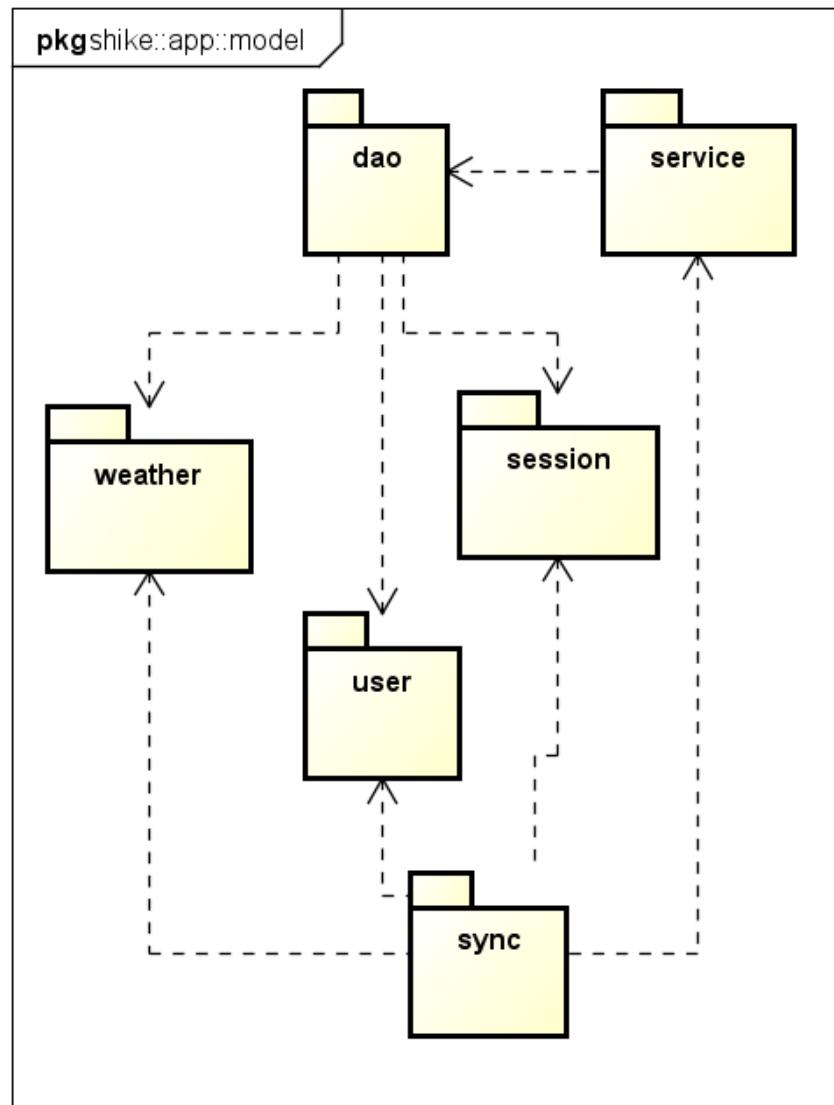


Figura 12: Diagramma di shike::app::model

- **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

4.5 shike::app::model::weather

4.5.1 Informazioni sulla componente

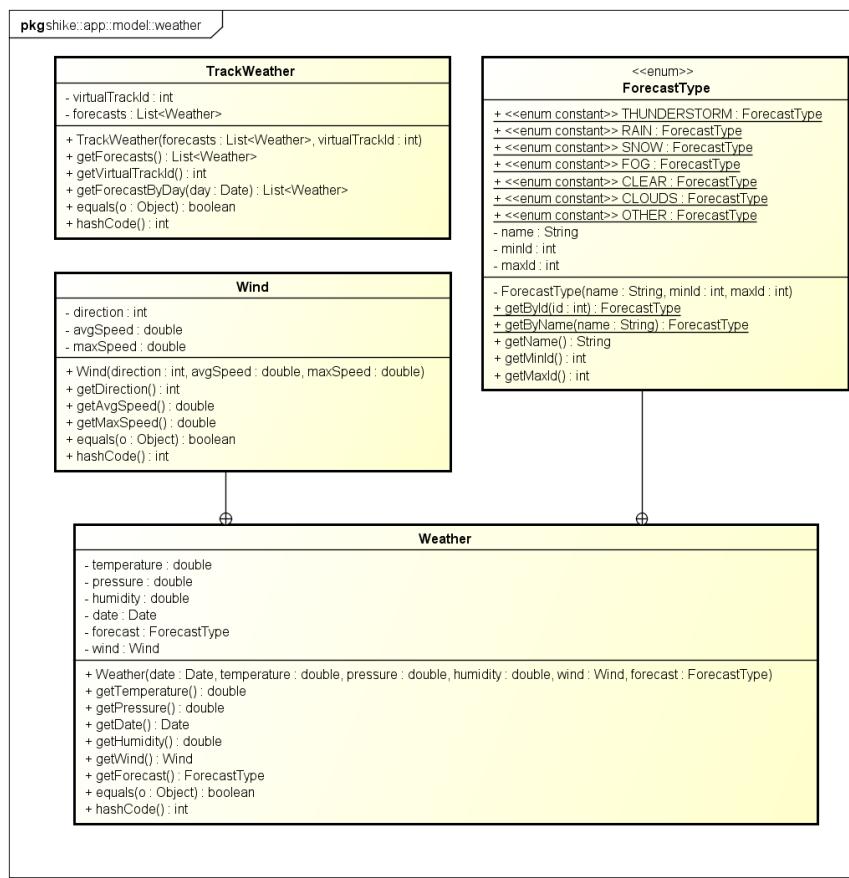


Figura 13: Diagramma di shike::app::model::weather

- **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

4.5.2 Classi

4.5.2.1 shike::app::model::weather::Weather

- **Tipo:** concreta
- **Descrizione:** classe che modella una singola previsione meteo
- **Relazioni con altre classi**
 - ← TrackWeather: classe che modella le informazioni relative alle condizioni meteo previste in un percorso.
 - → Weather.ForecastType: enum che indica il tipo di condizioni meteo previste. I codici indicati nelle condizioni si riferiscono agli identificatori di <http://openweathermap.org/weather-conditions>
 - → Weather.Wind: classe che modella le informazioni sul vento di una previsione meteo.

4.5.2.2 shike::app::model::weather::TrackWeather

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni relative alle condizioni meteo previste in un percorso.
- **Relazioni con altre classi**
 - ← VirtualTrack: classe che modella un percorso scaricato dalla piattaforma web.
 - ← TrackWeatherDao: interfaccia che fornisce i metodi per la gestione delle previsioni meteo dei percorsi.
 - ← TrackWeatherService: classe *service* che fa da ponte tra i *presenter* e il DAO delle previsioni meteo dei percorsi.
 - ← SyncDataWeb: classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
 - → Weather: classe che modella una singola previsione meteo.

4.5.2.3 shike::app::model::weather::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>
- **Relazioni con altre classi**
 - ← Weather: classe che modella una singola previsione meteo.

4.5.2.4 shike::app::model::weather::Weather.Wind

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni sul vento di una previsione meteo
- **Relazioni con altre classi**
 - ← Weather: classe che modella una singola previsione meteo.

4.6 shike::app::model::session

4.6.1 Informazioni sulla componente

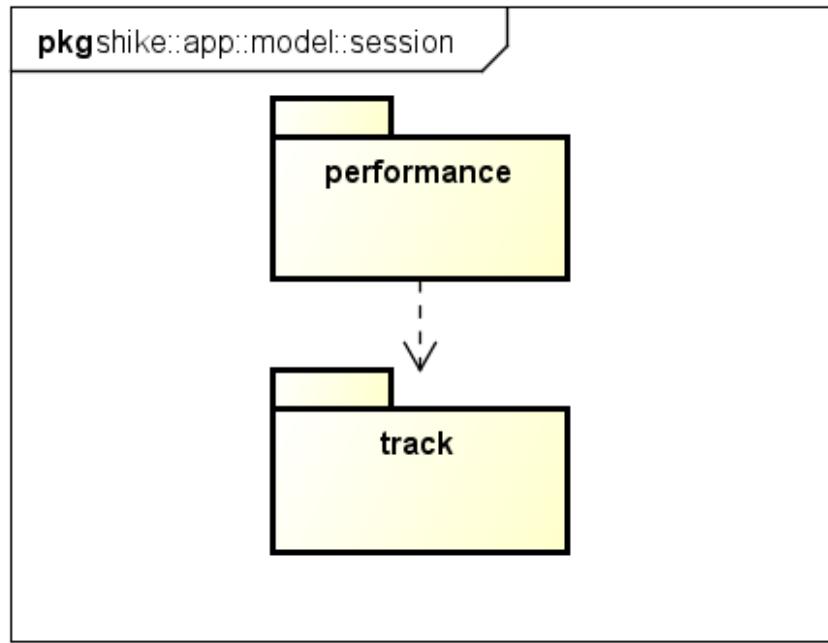


Figura 14: Diagramma di shike::app::model::session

- **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

4.7 shike::app::model::session::track

4.7.1 Informazioni sulla componente

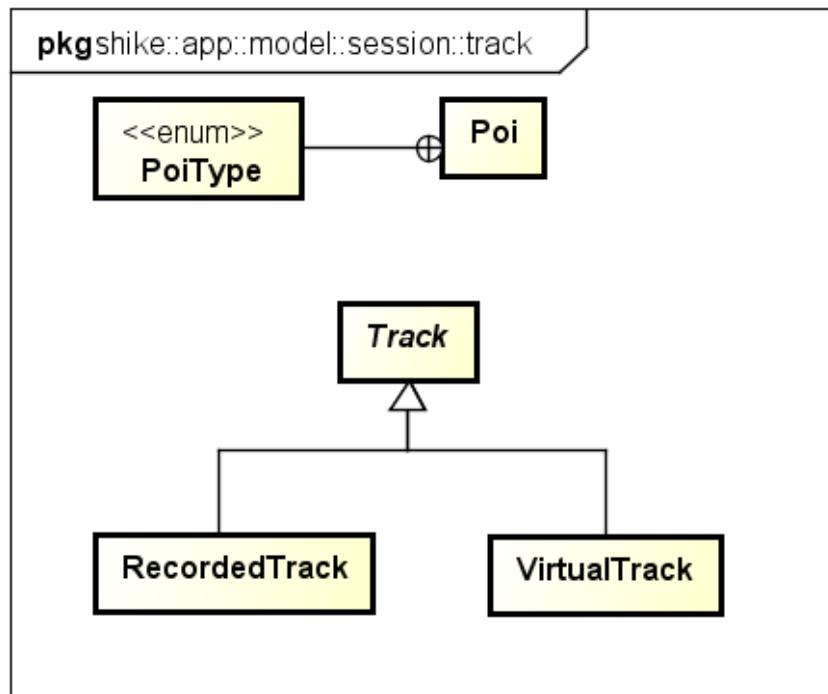


Figura 15: Diagramma di shike::app::model::session::track

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

4.7.2 Classi

4.7.2.1 shike::app::model::session::track::Poi

- **Tipo:** concreta
- **Descrizione:** classe che modella un POI.
- **Relazioni con altre classi**
 - \leftarrow PoiService: classe *service* che fa da ponte tra i presenter e il DAO dei punti di interesse.
 - \leftarrow SyncDataWeb: classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
 - \rightarrow PoiDao: interfaccia che fornisce i metodi per gestire la tabella 'pois'.
 - \rightarrow Poi.PoiType: enumeratore interno alla classe Poi contenente le varie tipologie di POI.

4.7.2.2 shike::app::model::session::track::Track

- **Tipo:** astratta
- **Descrizione:** classe astratta che modella un percorso generico
- **Sottoclassi:**
 - VirtualTrack:
 - RecordedTrack:

4.7.2.3 shike::app::model::session::track::VirtualTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella un percorso scaricato dalla piattaforma web
- **Superclassi:**
 - Track:
- **Relazioni con altre classi**
 - ← VirtualTrackDao: interfaccia che fornisce i metodi per la gestione dei percorsi scaricati dal web e salvati nel *database*.
 - ← VirtualTrackService: classe *service* che fa da ponte tra i *presenter* e il DAO dei percorsi scaricati dalla parte web.
 - ← SyncDataWeb: classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
 - → TrackWeather: classe che modella le informazioni relative alle condizioni meteo previste in un percorso.

4.7.2.4 shike::app::model::session::track::RecordedTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella un percorso creato durante una sessione
- **Superclassi:**
 - Track:
- **Relazioni con altre classi**
 - ← Performance: classe che modella una performance dell’utente

4.7.2.5 shike::app::model::session::track::Poi.PoiType

- **Tipo:** enum
- **Descrizione:** enumeratore interno alla classe Poi contenente le varie tipologie di POI
- **Relazioni con altre classi**
 - ← Poi: classe che modella un POI.
 - ← PoiDao: interfaccia che fornisce i metodi per gestire la tabella ‘pois’.

4.8 shike::app::model::session::performance

4.8.1 Informazioni sulla componente

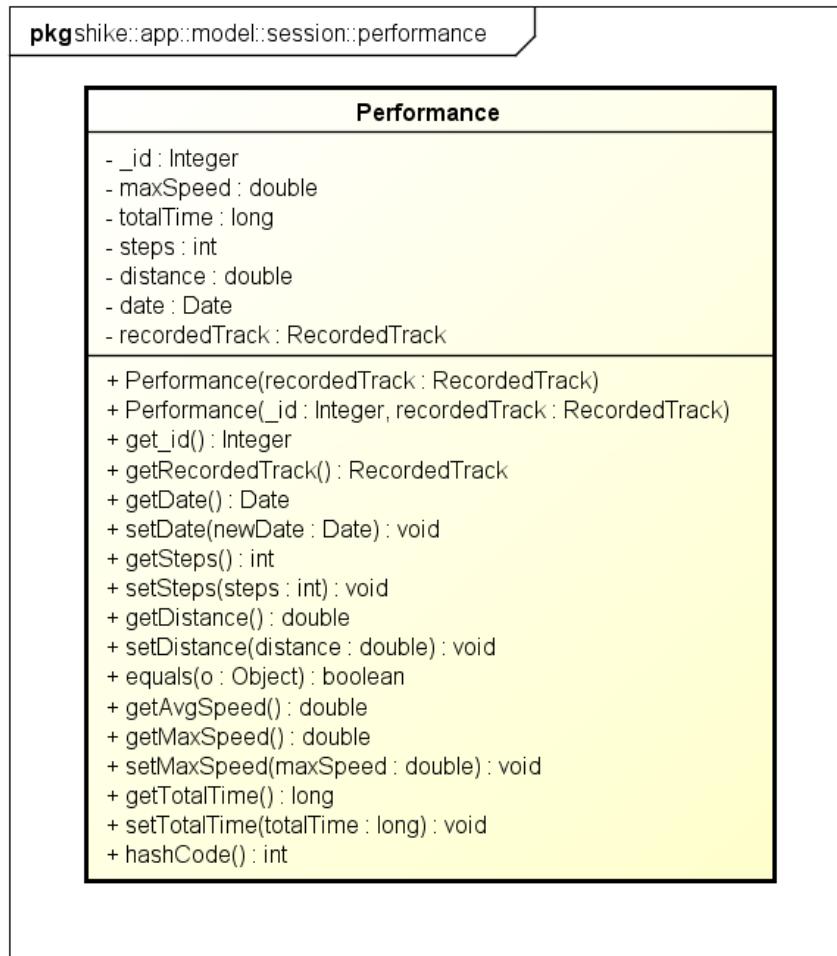


Figura 16: Diagramma di shike::app::model::session::performance

- **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

4.8.2 Classi

4.8.2.1 shike::app::model::session::performance::Performance

- **Tipo:** concreta
- **Descrizione:** classe che modella una performance dell’utente

- **Relazioni con altre classi**

- ← PerformanceDao: interfaccia che fornisce i metodi per gestire la tabella 'performance'.
- ← PerformanceService: classe *service* che fa da ponte tra i presenter e il DAO delle performance dell'utente.
- ← SyncDataApp: classe che modella i dati che l'applicazione invia alla piattaforma web durante una sincronizzazione.
- → RecordedTrack: classe che modella un percorso creato durante una sessione.

4.9 shike::app::model::user

4.9.1 Informazioni sulla componente

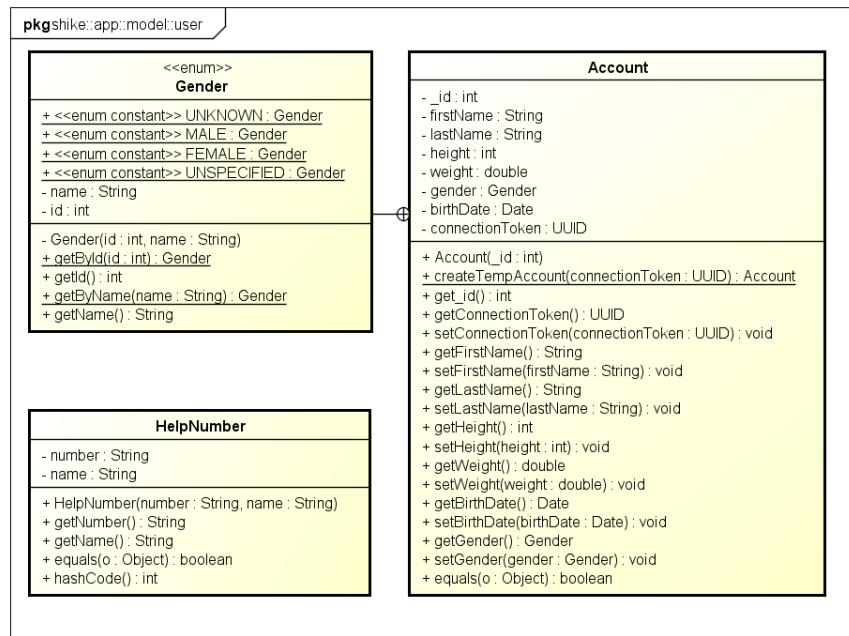


Figura 17: Diagramma di `shike::app::model::user`

- **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`

4.9.2 Classi

4.9.2.1 `shike::app::model::user::Account`

- **Tipo:** concreta

- **Descrizione:** classe che modella l' *account* utente collegato al dispositivo.
- **Relazioni con altre classi**
 - ← SyncDataWeb: classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → AccountDao: interfaccia che fornisce i metodi di interfacciamento con il *database* relativamente all' *account* associato all'applicazione.
 - → Account.Gender: enum che indica i possibili sessi indicati dallo standard ISO 5218
 - → AccountService: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dell'*account*.

4.9.2.2 shike::app::model::user::HelpNumber

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.
- **Relazioni con altre classi**
 - ← HelpNumberDao: interfaccia che fornisce i metodi per gestire la tabella 'help-numbers' del *database*.
 - ← HelpNumberService: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dei numeri di soccorso.
 - ← SyncDataWeb: classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
 - ← SyncDataWeb: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.

4.9.2.3 shike::app::model::user::Account.Gender

- **Tipo:** enum
- **Descrizione:** enum che indica i possibili sessi indicati dallo standard ISO 5218
- **Relazioni con altre classi**
 - ← Account: classe che modella l' *account* utente collegato al dispositivo.

4.10 shike::app::model::dao

4.10.1 Informazioni sulla componente

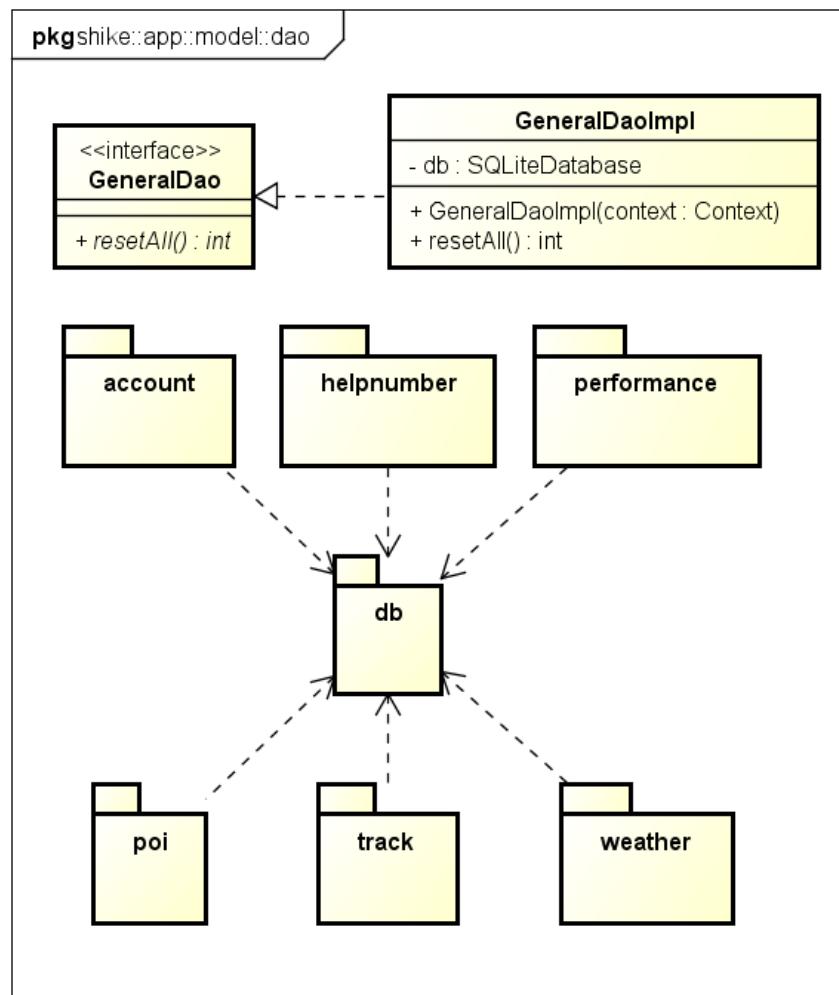


Figura 18: Diagramma di shike::app::model::dao

- **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

4.10.2 Classi

4.10.2.1 shike::app::model::dao::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:**
 - GeneralDaoImpl:
- **Relazioni con altre classi**
 - → GeneralService: servizio di gestione generale della memoria dell'applicazione.

4.10.2.2 shike::app::model::dao::GeneralDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di GeneralDao.
- **Implementata da:**
 - GeneralDao:
- **Relazioni con altre classi**
 - → AccountTable: classe contenente le informazioni necessarie per poter gestire la tabella *account* del *database*.
 - → HelpNumbersTable: classe contenente le informazioni necessarie per poter gestire la tabella *helpnumbers* del *database*.
 - → RecordedTracksTable: classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracks* del *database*.
 - → VirtualTracksTable: classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracks* del *database*.
 - → PoisTable: classe contenente le informazioni necessarie per poter gestire la tabella *pois* del *database*.
 - → GeneralService: servizio di gestione generale della memoria dell'applicazione.

4.11 shike::app::model::dao::account

4.11.1 Informazioni sulla componente

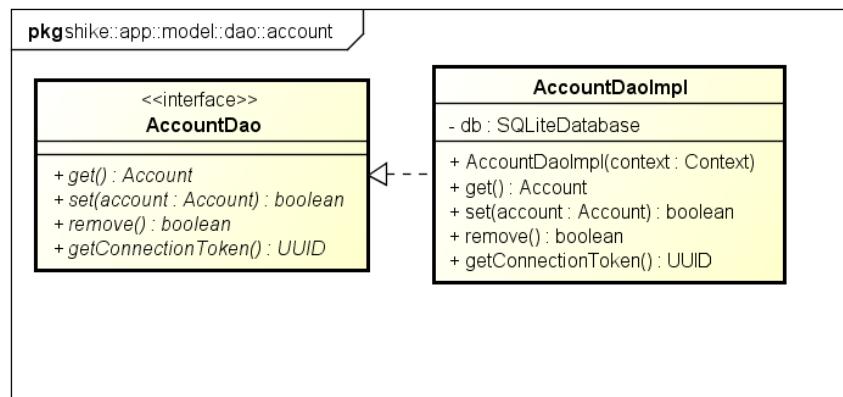


Figura 19: Diagramma di shike::app::model::dao::account

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

4.11.2 Classi

4.11.2.1 shike::app::model::dao::account::AccountDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi di interfacchiamento con il *database* relativamente all' *account* associato all'applicazione.
- **Implementata da:**
 - AccountDaoImpl;
- **Relazioni con altre classi**
 - ← Account: classe che modella l' *account* utente collegato al dispositivo.
 - → AccountService: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dell'*account*.

4.11.2.2 shike::app::model::dao::account::AccountDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di AccountDao.
- **Implementa:**
 - AccountDao;
- **Relazioni con altre classi**
 - → DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.

- → AccountTable: classe contenente le informazioni necessarie per poter gestire la tabella *account* del *database*.
- → AccountService: classe di servizio che fa da ponte tra i presenter e l’interfaccia DAO dell’*account*.

4.12 shike::app::model::dao::helpnumber

4.12.1 Informazioni sulla componente

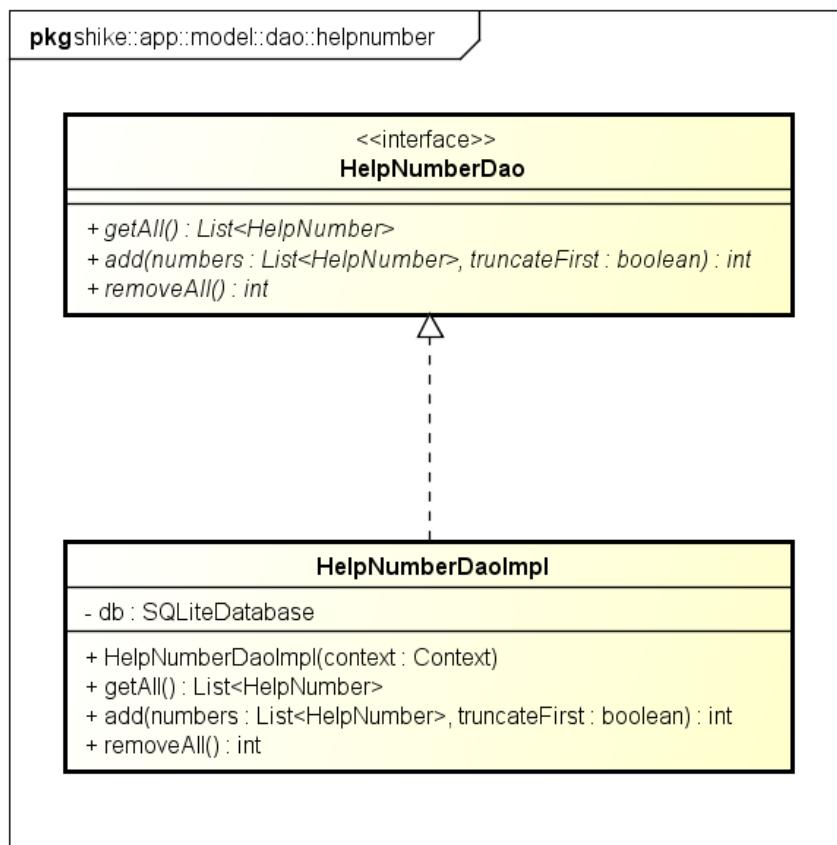


Figura 20: Diagramma di `shike::app::model::dao::helpnumber`

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

4.12.2 Classi

4.12.2.1 shike::app::model::dao::helpnumber::HelpNumberDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per gestire la tabella ‘helpnumbers’ del *database*.

- **Implementata da:**

- HelpNumberDaoImpl:

- **Relazioni con altre classi**

- ← HelpNumberService: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dei numeri di soccorso.
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.

4.12.2.2 shike::app::model::dao::helpnumber::HelpNumberDaoImpl

- **Tipo:** concreta

- **Descrizione:** implementazione di HelpNumberDao

- **Implementa:**

- HelpNumberDao:

- **Relazioni con altre classi**

- ← HelpNumberService: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dei numeri di soccorso.
 - → DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.
 - → HelpNumbersTable: classe contenente le informazioni necessarie per poter gestire la tabella *helpnumbers* del *database*.

4.13 shike::app::model::dao::performance

4.13.1 Informazioni sulla componente

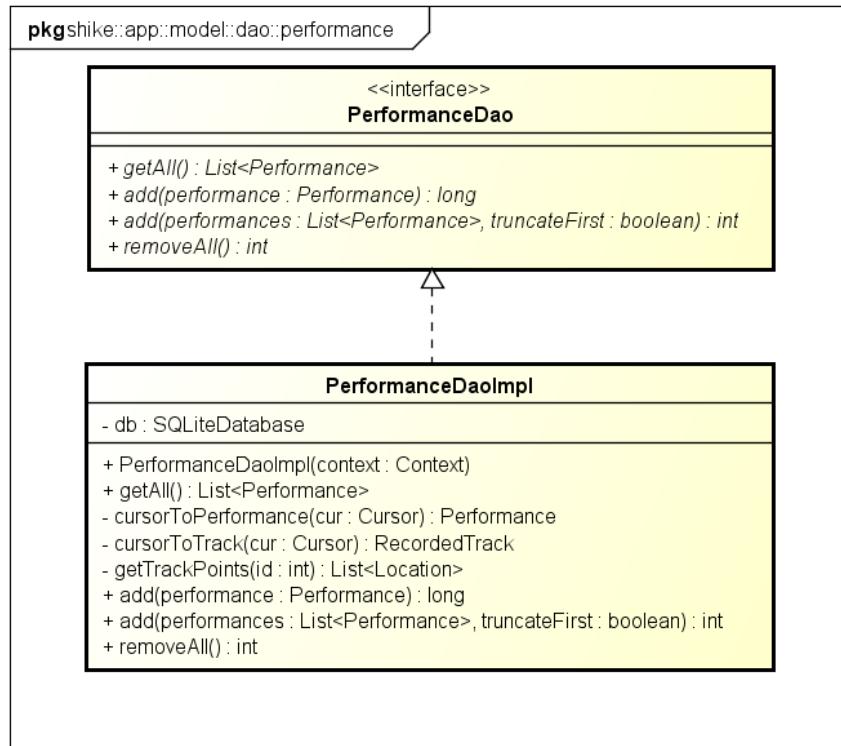


Figura 21: Diagramma di shike::app::model::dao::performance

- **Descrizione:** componente che modella il *design pattern* DAO relativo alle *performance* dell’utente.
- **Componente padre:** shike::app::model::dao

4.13.2 Classi

4.13.2.1 shike::app::model::dao::performance::PerformanceDaolmpl

- **Tipo:** concreta
- **Descrizione:** implementazione di PerformanceDao.
- **Implementa:**
 - PerformanceDao:
- **Relazioni con altre classi**
 - ← RecordedTracksTable: classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracks* del *database*.
 - ← PerformanceService: classe *service* che fa da ponte tra i presenter e il DAO delle performance dell’utente.
 - → DbUtil: classe che si occupa di gestire l’inizializzazione e la connessione al *database* dell’applicazione.

4.13.2.2 shike::app::model::dao::performance::PerformanceDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per gestire la tabella 'performance'
- **Implementata da:**
 - PerformanceDaoImpl:
- **Relazioni con altre classi**
 - → Performance: classe che modella una performance dell'utente

4.13.2.3 shike::app::model::dao::performance::PerformanceDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di PerformanceDao
- **Implementa:**
 - PerformanceDao:
- **Relazioni con altre classi**
 - → DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.

4.13.2.4 shike::app::model::dao::performance::PerformanceDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per la gestione delle performance dell'utente salvate nel database.
- **Implementata da:**
 - PerformanceDaoImpl:
- **Relazioni con altre classi**
 - ← PerformanceService: classe *service* che fa da ponte tra i presenter e il DAO delle performance dell'utente.

4.14 shike::app::model::dao::poi

4.14.1 Informazioni sulla componente

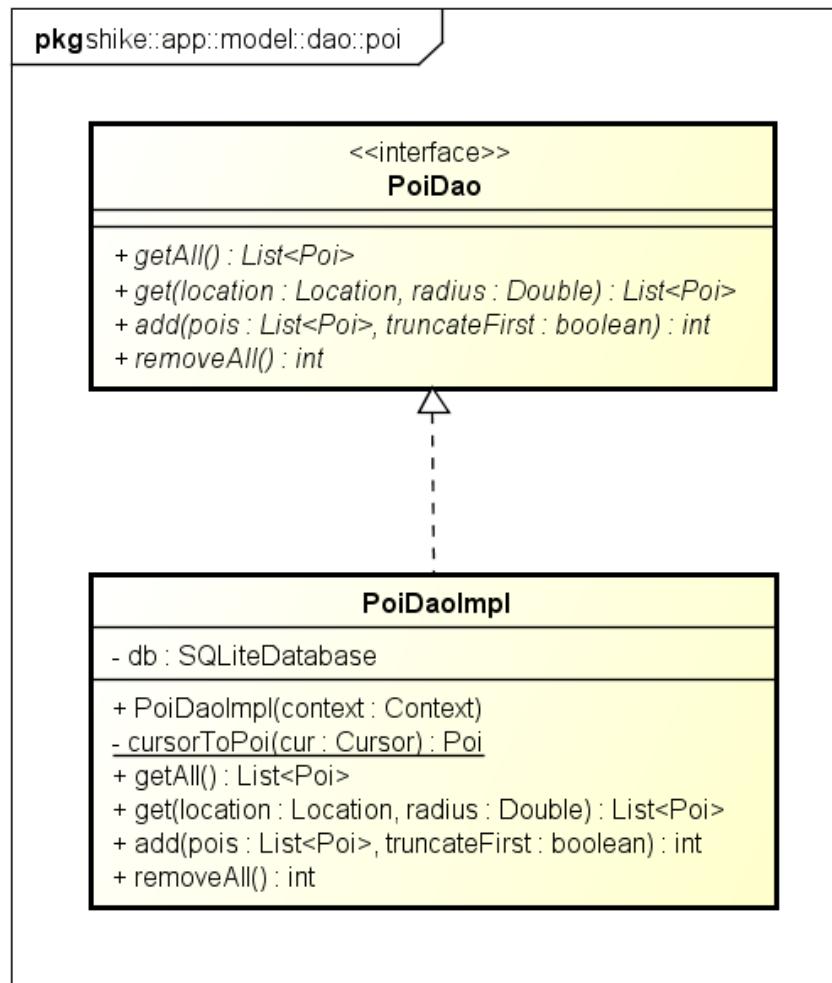


Figura 22: Diagramma di shike::app::model::dao::poi

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

4.14.2 Classi

4.14.2.1 shike::app::model::dao::poi::PoiDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di PoiDao
- **Implementa:**
 - PoiDao:

- **Relazioni con altre classi**

- ← PoiService: classe *service* che fa da ponte tra i presenter e il DAO dei punti di interesse.
- → DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.
- → PoisTable: classe contenente le informazioni necessarie per poter gestire la tabella *pois* del *database*.

4.14.2.2 shike::app::model::dao::poi::PoiDao

- **Tipo:** interfaccia

- **Descrizione:** interfaccia che fornisce i metodi per gestire la tabella 'pois'

- **Implementata da:**

- PoiDaoImpl:

- **Relazioni con altre classi**

- ← Poi: classe che modella un POI.
- ← PoiService: classe *service* che fa da ponte tra i presenter e il DAO dei punti di interesse.
- → Poi.PoiType: enumeratore interno alla classe Poi contenente le varie tipologie di POI.

4.15 shike::app::model::dao::track

4.15.1 Informazioni sulla componente

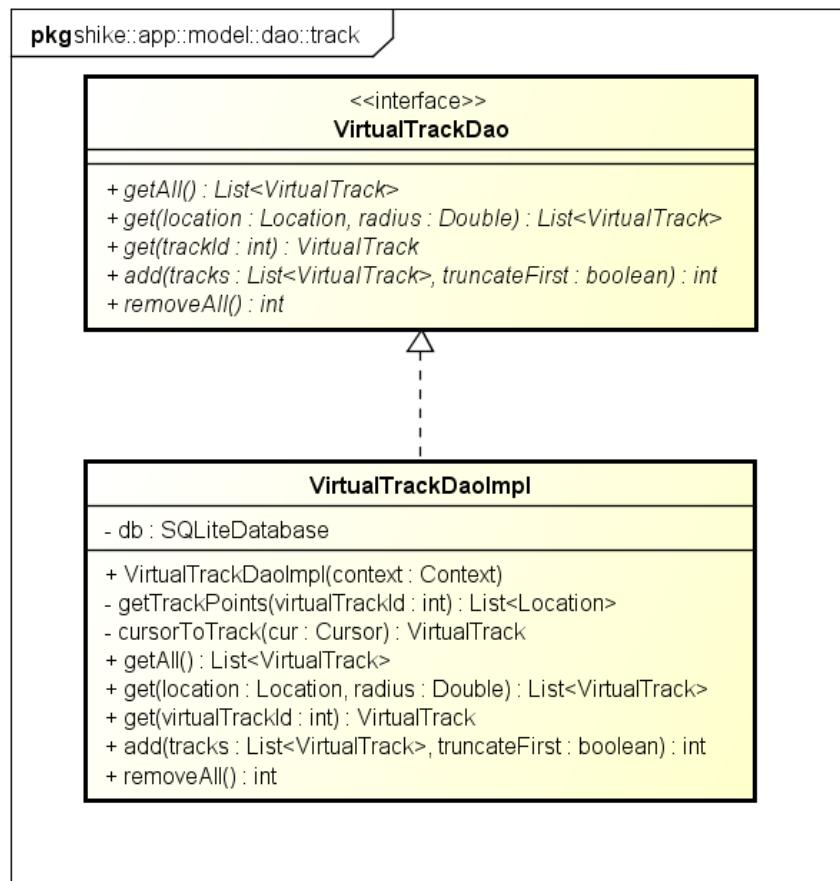


Figura 23: Diagramma di shike::app::model::dao::track

- **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

4.15.2 Classi

4.15.2.1 shike::app::model::dao::track::VirtualTrackDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di `VirtualTrackDao`
- **Implementa:**
 - `VirtualTrackDao`:
- **Relazioni con altre classi**

- ← VirtualTrackService: classe *service* che fa da ponte tra i *presenter* e il DAO dei percorsi scaricati dalla parte web.
- → DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.
- → VirtualTracksTable: classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracks* del *database*.

4.15.2.2 shike::app::model::dao::track::VirtualTrackDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per la gestione dei percorsi scaricati dal web e salvati nel *database*.
- **Implementata da:**
 - VirtualTrackDaoImpl:
- **Relazioni con altre classi**
 - ← VirtualTrackService: classe *service* che fa da ponte tra i *presenter* e il DAO dei percorsi scaricati dalla parte web.
 - → VirtualTrack: classe che modella un percorso scaricato dalla piattaforma web.

4.16 shike::app::model::dao::weather

4.16.1 Informazioni sulla componente

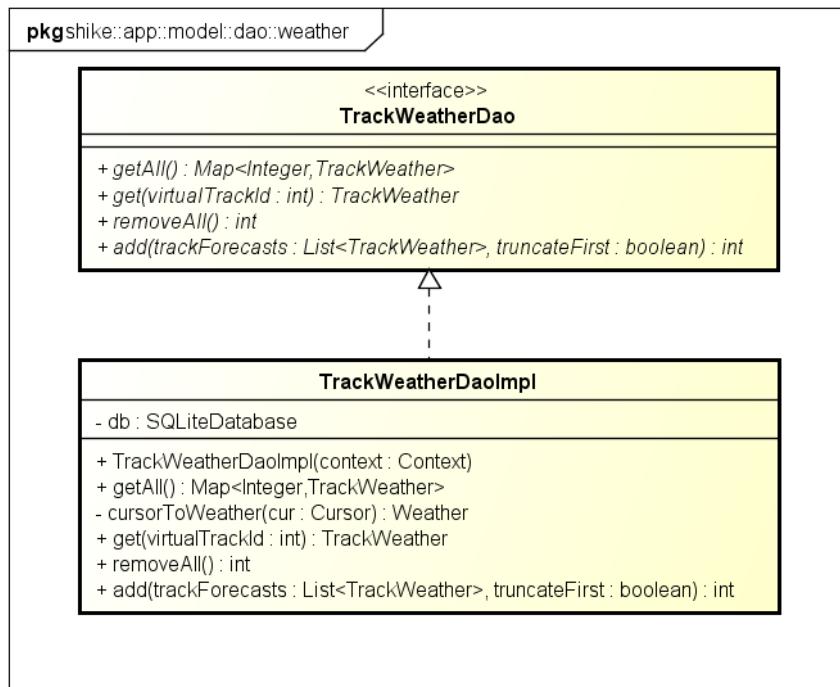


Figura 24: Diagramma di shike::app::model::dao::weather

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

4.16.2 Classi

4.16.2.1 shike::app::model::dao::weather::TrackWeatherDao

- **Tipo:** interfaccia
- **Descrizione:** interfaccia che fornisce i metodi per la gestione delle previsioni meteo dei percorsi
- **Implementata da:**
 - TrackWeatherDaoImpl:
- **Relazioni con altre classi**
 - ← TrackWeatherService: classe *service* che fa da ponte tra i *presenter* e il DAO delle previsioni meteo dei percorsi.
 - → TrackWeather: classe che modella le informazioni relative alle condizioni meteo previste in un percorso.

4.16.2.2 shike::app::model::dao::weather::TrackWeatherDaoImpl

- **Tipo:** concreta
- **Descrizione:** implementazione di TrackWeatherDao
- **Implementa:**
 - TrackWeatherDao:
- **Relazioni con altre classi**
 - ← WeatherForecastsTable: classe contenente le informazioni necessarie per poter gestire la tabella *weatherforecasts* del *database*.
 - ← TrackWeatherService: classe *service* che fa da ponte tra i *presenter* e il DAO delle previsioni meteo dei percorsi.
 - → DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.

4.17 shike::app::model::dao::db

4.17.1 Informazioni sulla componente

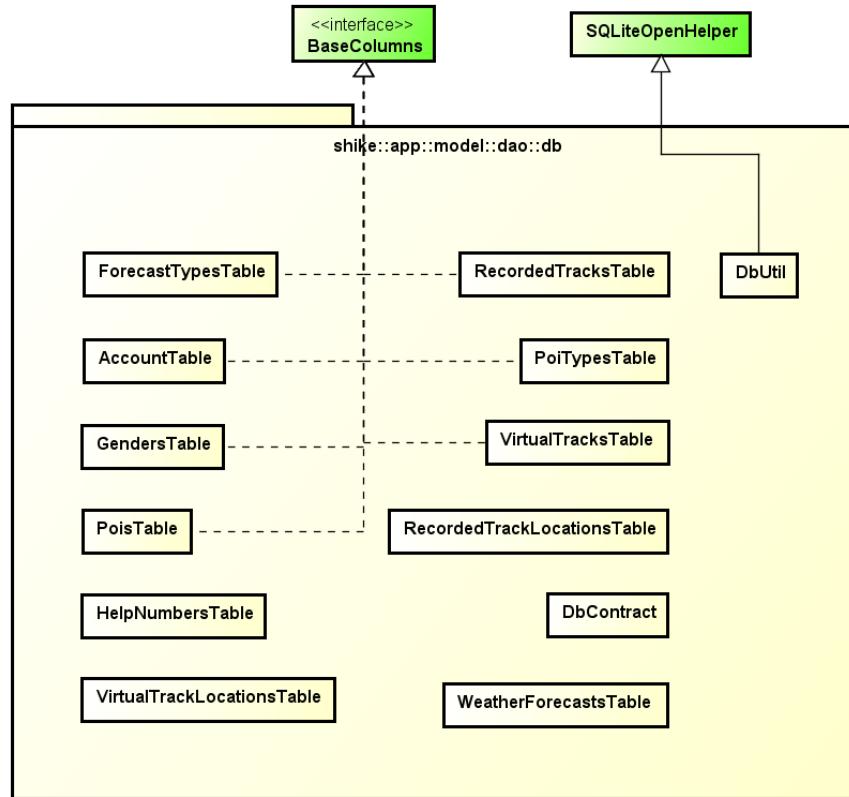


Figura 25: Diagramma di shike::app::model::dao::db

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

4.17.2 Classi

4.17.2.1 shike::app::model::dao::db::AccountTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *account* del *database*.
- **Implementa:**
 - android.provider.BaseColumns
- **Relazioni con altre classi**
 - ← `DbContract`: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.

- ← AccountDaoImpl: implementazione di AccountDao.
- ← GeneralDaoImpl: implementazione di GeneralDao.

4.17.2.2 shike::app::model::dao::db::GendersTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *genders* del *database*.
- **Implementa:**
 - android.provider.BaseColumns
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.

4.17.2.3 shike::app::model::dao::db::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.
- **Relazioni con altre classi**
 - ← DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.
 - → AccountTable: classe contenente le informazioni necessarie per poter gestire la tabella *account* del *database*.
 - → GendersTable: classe contenente le informazioni necessarie per poter gestire la tabella *genders* del *database*.
 - → HelpNumbersTable: classe contenente le informazioni necessarie per poter gestire la tabella *helpnumbers* del *database*.
 - → RecordedTracksTable: classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracks* del *database*.
 - → RecordedTrackLocationsTable: classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracklocations* del *database*.
 - → VirtualTracksTable: classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracks* del *database*.
 - → VirtualTrackLocationsTable: classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracklocations* del *database*.
 - → WeatherForecastsTable: classe contenente le informazioni necessarie per poter gestire la tabella *weatherforecasts* del *database*.

- → ForecastTypesTable: classe contenente le informazioni necessarie per poter gestire la tabella *forecasttypes* del *database*.
- → PoisTable: classe contenente le informazioni necessarie per poter gestire la tabella *pois* del *database*.
- → PoiTypesTable: classe contenente le informazioni necessarie per poter gestire la tabella *poitypes* del *database*.

4.17.2.4 shike::app::model::dao::db::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
- **Relazioni con altre classi**
 - ← AccountDaoImpl: implementazione di AccountDao.
 - ← HelpNumberDaoImpl: implementazione di HelpNumberDao.
 - ← PerformanceDaoImpl: implementazione di PerformanceDao.
 - ← PoiDaoImpl: implementazione di PoiDao.
 - ← PerformanceDaoImpl: implementazione di PerformanceDao.
 - ← VirtualTrackDaoImpl: implementazione di VirtualTrackDao.
 - ← TrackWeatherDaoImpl: implementazione di TrackWeatherDao.
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.

4.17.2.5 shike::app::model::dao::db::HelpNumbersTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *helpnumbers* del *database*.
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.
 - ← HelpNumberDaoImpl: implementazione di HelpNumberDao.
 - ← GeneralDaoImpl: implementazione di GeneralDao.

4.17.2.6 shike::app::model::dao::db::RecordedTracksTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracks* del *database*.
- **Implementa:**
 - android.provider.BaseColumns
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.
 - ← GeneralDaoImpl: implementazione di GeneralDao.
 - → PerformanceDaoImpl: implementazione di PerformanceDao.

4.17.2.7 shike::app::model::dao::db::RecordedTrackLocationsTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *recordedtracklocations* del *database*.
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.

4.17.2.8 shike::app::model::dao::db::VirtualTracksTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracks* del *database*.
- **Implementa:**
 - android.provider.BaseColumns
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.
 - ← VirtualTrackDaoImpl: implementazione di VirtualTrackDao.
 - ← GeneralDaoImpl: implementazione di GeneralDao.

4.17.2.9 shike::app::model::dao::db::VirtualTrackLocationsTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *virtualtracklocations* del *database*.
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.

4.17.2.10 shike::app::model::dao::db::WeatherForecastsTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *weatherforecasts* del *database*.
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.
 - → TrackWeatherDaoImpl: implementazione di TrackWeatherDao.

4.17.2.11 shike::app::model::dao::db::ForecastTypesTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *forecasttypes* del *database*.
- **Implementa:**
 - android.provider.BaseColumns
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.

4.17.2.12 shike::app::model::dao::db::PoisTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *pois* del *database*.
- **Implementa:**
 - android.provider.BaseColumns
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.
 - ← PoiDaoImpl: implementazione di PoiDao.
 - ← GeneralDaoImpl: implementazione di GeneralDao.

4.17.2.13 shike::app::model::dao::db::PoiTypesTable

- **Tipo:** concreta
- **Descrizione:** classe contenente le informazioni necessarie per poter gestire la tabella *poitypes* del *database*.
- **Implementa:**
 - android.provider.BaseColumns
- **Relazioni con altre classi**
 - ← DbContract: classe che contiene tutte le informazioni riguardanti lo schema del *database* dell'applicazione.

4.18 shike::app::model::service

4.18.1 Informazioni sulla componente

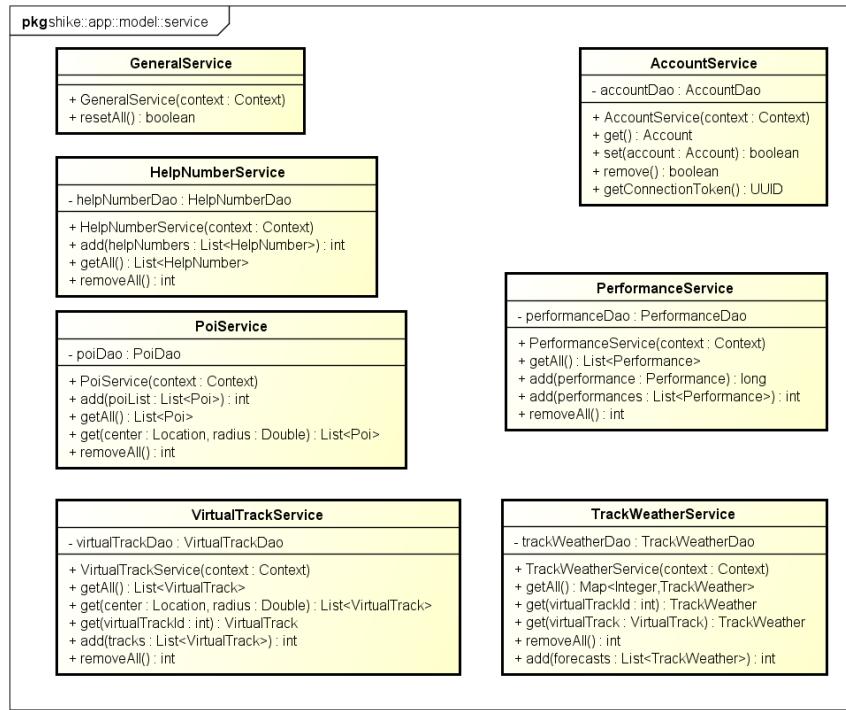


Figura 26: Diagramma di shike::app::model::service

- **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

4.18.2 Classi

4.18.2.1 shike::app::model::service::AccountService

- **Tipo:** concreta
- **Descrizione:** classe di servizio che fa da ponte tra i presenter e l’interfaccia DAO dell’*account*.
- **Relazioni con altre classi**
 - ← Account: classe che modella l’*account* utente collegato al dispositivo.
 - ← AccountDao: interfaccia che fornisce i metodi di interfacciamento con il *database* relativamente all’*account* associato all’applicazione.
 - ← AccountDaoImpl: implementazione di AccountDao.

- ← SyncDataApp: classe che modella i dati che l'applicazione invia alla piattaforma web durante una sincronizzazione.
- ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.

4.18.2.2 shike::app::model::service::GeneralService

- **Tipo:** concreta
- **Descrizione:** servizio di gestione generale della memoria dell'applicazione.
- **Relazioni con altre classi**

- ← GeneralDao: 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).
- ← GeneralDaoImpl: implementazione di GeneralDao.
- ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.

4.18.2.3 shike::app::model::service::HelpNumberService

- **Tipo:** concreta
 - **Descrizione:** classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dei numeri di soccorso.
 - **Relazioni con altre classi**
- ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.
 - → HelpNumberDao: interfaccia che fornisce i metodi per gestire la tabella 'help-numbers' del database.
 - → HelpNumberDaoImpl: implementazione di HelpNumberDao.

4.18.2.4 shike::app::model::service::PerformanceService

- **Tipo:** concreta
 - **Descrizione:** classe service che fa da ponte tra i presenter e il DAO delle performance dell'utente.
 - **Relazioni con altre classi**
- ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → Performance: classe che modella una performance dell'utente
 - → PerformanceDaoImpl: implementazione di PerformanceDao.
 - → PerformanceDao: interfaccia che fornisce i metodi per la gestione delle performance dell'utente salvate nel database.

4.18.2.5 shike::app::model::service::PoiService

- **Tipo:** concreta
- **Descrizione:** classe *service* che fa da ponte tra i *presenter* e il DAO dei punti di interesse.
- **Relazioni con altre classi**
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → Poi: classe che modella un POI.
 - → PoiDao: interfaccia che fornisce i metodi per gestire la tabella 'pois'.
 - → PoiDaoImpl: implementazione di PoiDao.

4.18.2.6 shike::app::model::service::TrackWeatherService

- **Tipo:** concreta
- **Descrizione:** classe *service* che fa da ponte tra i *presenter* e il DAO delle previsioni meteo dei percorsi.
- **Relazioni con altre classi**
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → TrackWeather: classe che modella le informazioni relative alle condizioni meteo previste in un percorso.
 - → TrackWeatherDao: interfaccia che fornisce i metodi per la gestione delle previsioni meteo dei percorsi.
 - → TrackWeatherDaoImpl: implementazione di TrackWeatherDao.

4.18.2.7 shike::app::model::service::VirtualTrackService

- **Tipo:** concreta
- **Descrizione:** classe *service* che fa da ponte tra i *presenter* e il DAO dei percorsi scaricati dalla parte web.
- **Relazioni con altre classi**
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → VirtualTrack: classe che modella un percorso scaricato dalla piattaforma web.
 - → VirtualTrackDao: interfaccia che fornisce i metodi per la gestione dei percorsi scaricati dal web e salvati nel *database*.
 - → VirtualTrackDaoImpl: implementazione di VirtualTrackDao.

4.19 shike::app::model::sync

4.19.1 Informazioni sulla componente

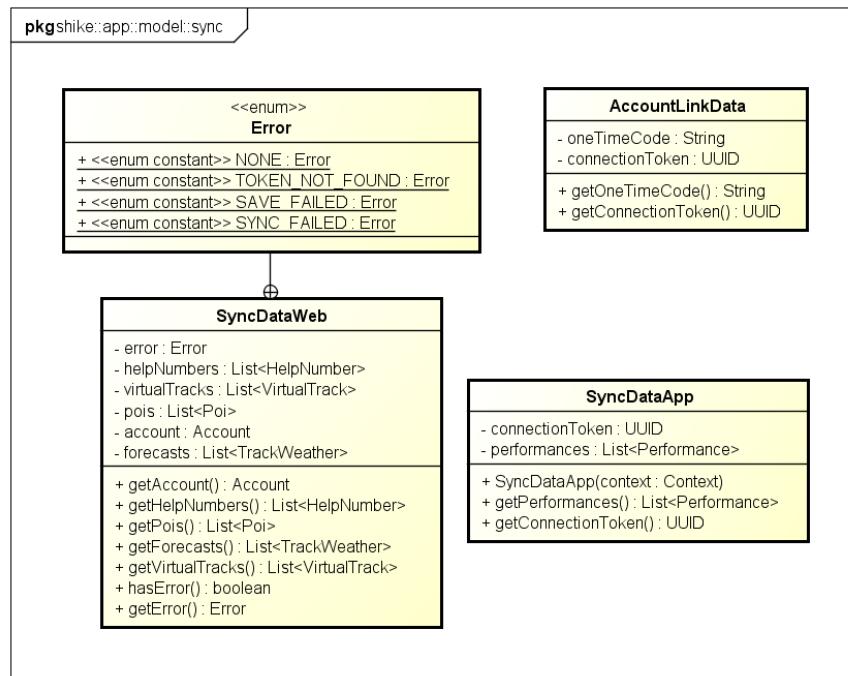


Figura 27: Diagramma di shike::app::model::sync

- **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`

4.19.2 Classi

4.19.2.1 shike::app::model::sync::SyncDataApp

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati che l'applicazione invia alla piattaforma web durante una sincronizzazione.
- **Relazioni con altre classi**
 - \leftarrow `JsonConverter`: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.
 - \rightarrow `Performance`: classe che modella una performance dell'utente
 - \rightarrow `AccountService`: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dell'*account*.

4.19.2.2 shike::app::model::sync::AccountLinkData

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati che vengono ricevuti dalla piattaforma web durante il primo collegamento dell' *account*.
- **Relazioni con altre classi**
 - ← JsonConverter: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.

4.19.2.3 shike::app::model::sync::SyncDataWeb

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
- **Relazioni con altre classi**
 - ← JsonConverter: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → Account: classe che modella l' *account* utente collegato al dispositivo.
 - → Poi: classe che modella un POI.
 - → VirtualTrack: classe che modella un percorso scaricato dalla piattaforma web.
 - → TrackWeather: classe che modella le informazioni relative alle condizioni meteo previste in un percorso.
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.

4.19.2.4 shike::app::model::sync::SyncDataWeb.Error

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

4.20 shike::app::view

4.20.1 Informazioni sulla componente

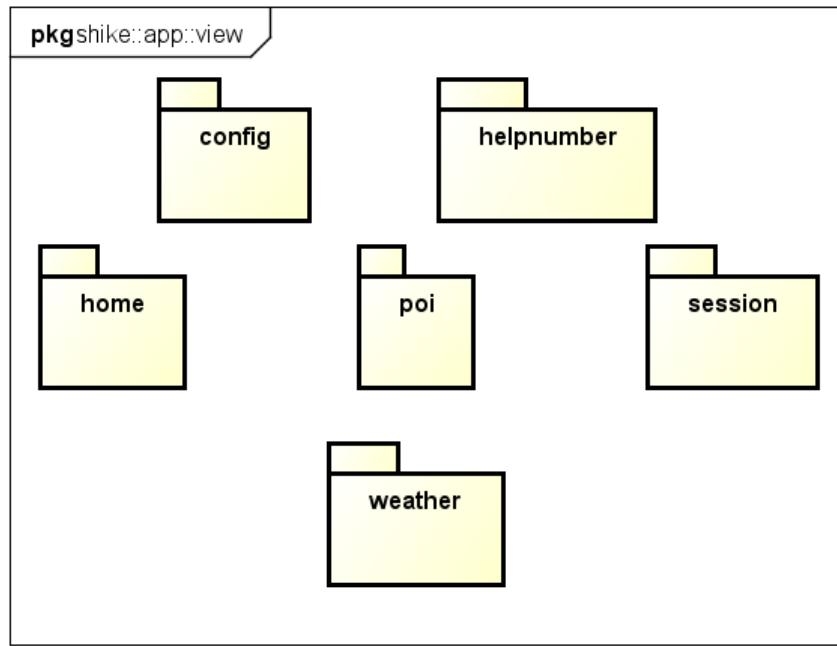


Figura 28: Diagramma di shike::app::view

- **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

4.21 shike::app::view::config

4.21.1 Informazioni sulla componente

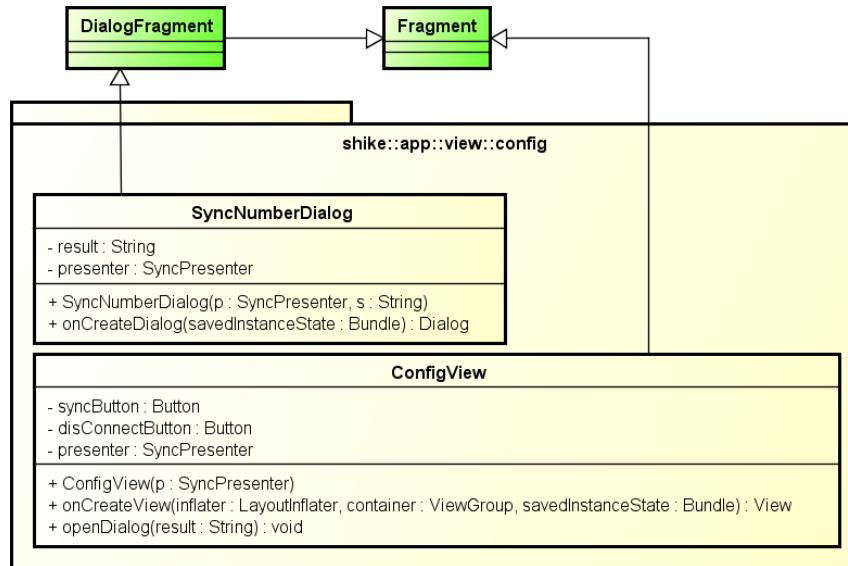


Figura 29: Diagramma di shike::app::view::config

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

4.21.2 Classi

4.21.2.1 shike::app::view::config::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
- **Relazioni con altre classi**
 - ← HomeView: classe che rappresenta la visualizzazione della pagina principale dell'applicazione

4.21.2.2 shike::app::view::config::SyncNumberDialog

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

4.22 shike::app::view::helpnumber

4.22.1 Informazioni sulla componente

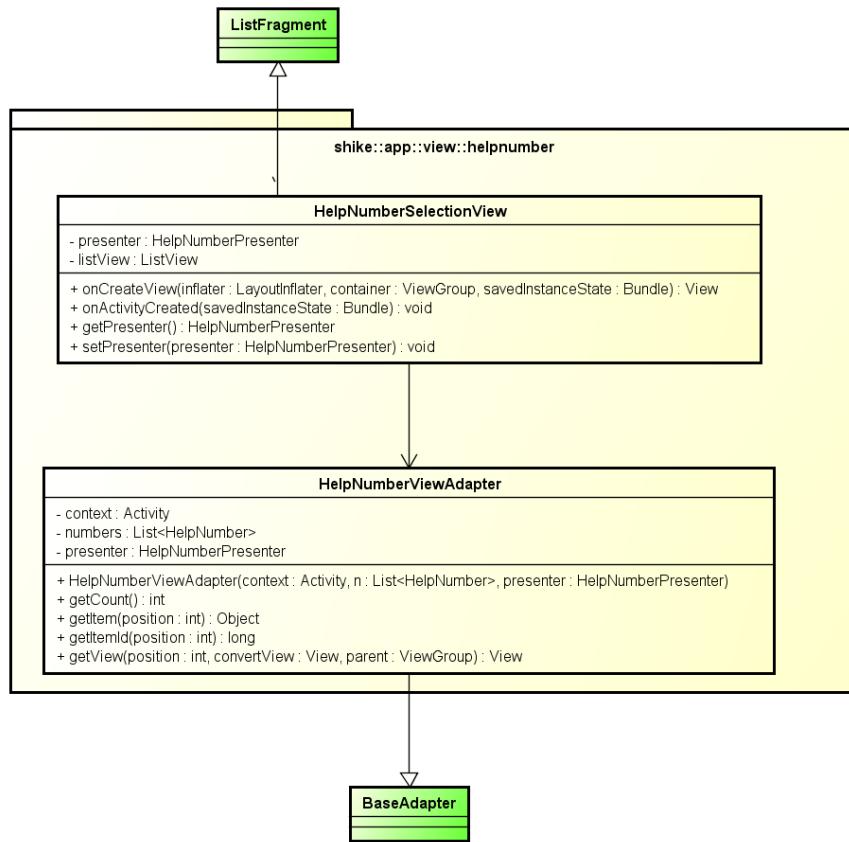


Figura 30: Diagramma di `shike::app::view::helpnumber`

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

4.22.2 Classi

4.22.2.1 shike::app::view::helpnumber::HelpNumberSelectionView

- **Tipo:** concreta
- **Descrizione:** classe che modella la visualizzazione dei numeri di soccorso.
- **Superclassi:**
 - `android.support.v4.app.ListFragment`
- **Relazioni con altre classi**

- HelpNumberViewAdapter: classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei numeri di soccorso.

4.22.2.2 shike::app::view::helpnumber::HelpNumberViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei numeri di soccorso.
- **Superclassi:**
 - android.widget.BaseAdapter
- **Relazioni con altre classi**
 - ← HelpNumberSelectionView: classe che modella la visualizzazione dei numeri di soccorso.

4.23 shike::app::view::home

4.23.1 Informazioni sulla componente

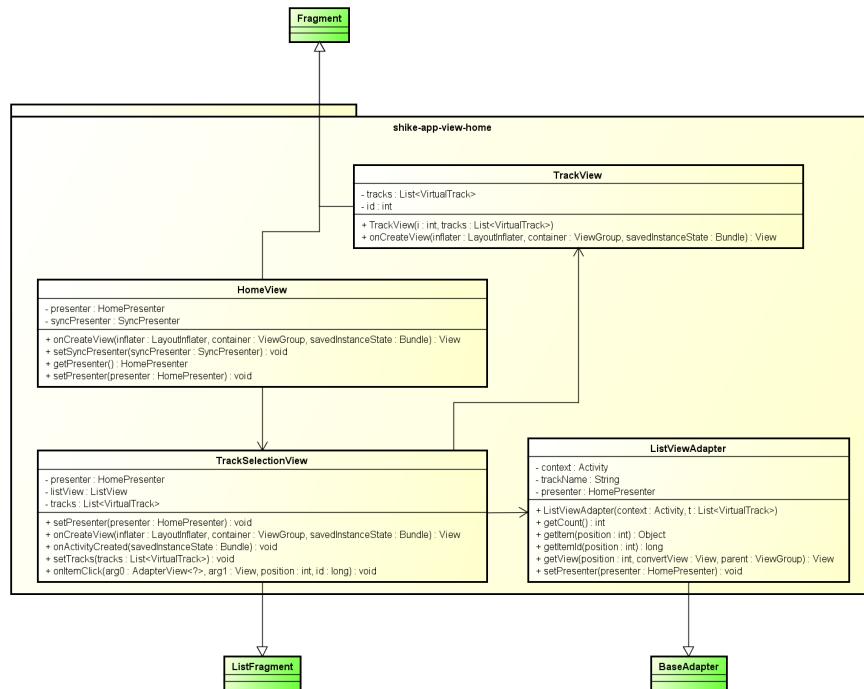


Figura 31: Diagramma di shike::app::view::home

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

4.23.2 Classi

4.23.2.1 shike::app::view::home::TrackView

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per la visualizzazione dei dati di un singolo percorso
- **Superclassi:**
 - android.support.v4.app.Fragment
- **Relazioni con altre classi**
 - ← TrackSelectionView: classe utilizzata per la visualizzazione della lista di tutti i *VirtualTrack* presenti nel dispositivo.

4.23.2.2 shike::app::view::home::HomeView

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta la visualizzazione della pagina principale dell'applicazione
- **Superclassi:**
 - android.support.v4.app.Fragment
- **Relazioni con altre classi**
 - ← HomePresenter: classe che fa da *presenter* per tutte le funzionalità utilizzate dalla *homepage*.
 - → ConfigView: classe che rappresenta la *view* che contiene le impostazioni modificabili dall'utente e la possibilità di scolare il profilo dalla piattaforma web.
 - → TrackSelectionView: classe utilizzata per la visualizzazione della lista di tutti i *VirtualTrack* presenti nel dispositivo.

4.23.2.3 shike::app::view::home::ListViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei tracciati.
- **Superclassi:**
 - android.widget.BaseAdapter
- **Relazioni con altre classi**
 - ← TrackSelectionView: classe utilizzata per la visualizzazione della lista di tutti i *VirtualTrack* presenti nel dispositivo.

4.23.2.4 shike::app::view::home::TrackSelectionView

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per la visualizzazione della lista di tutti i *VirtualTrack* presenti nel dispositivo.
- **Superclassi:**
 - android.support.v4.app.ListFragment
- **Relazioni con altre classi**
 - ← HomeView: classe che rappresenta la visualizzazione della pagina principale dell'applicazione
 - → ListViewAdapter: classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei tracciati.
 - → TrackView: classe utilizzata per la visualizzazione dei dati di un singolo percorso.

4.24 shike::app::view::poi

4.24.1 Informazioni sulla componente

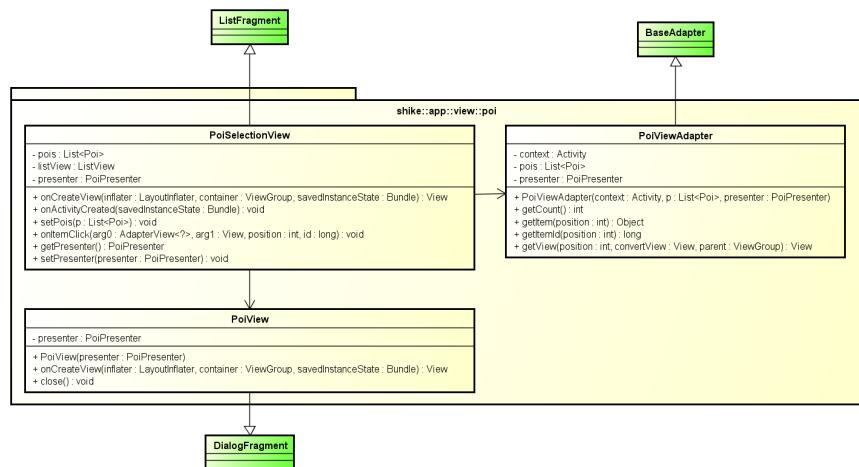


Figura 32: Diagramma di shike::app::view::poi

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

4.24.2 Classi

4.24.2.1 shike::app::view::poi::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
- **Relazioni con altre classi**
 - ← PoiPresenter: classe che fa da *presenter* per tutte le funzionalità utilizzate dei POI.
 - → PoiViewAdapter: classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei punti di interesse.
 - → PoiView: classe che modella la visualizzazione delle informazioni globali su un punto di interesse.

4.24.2.2 shike::app::view::poi::PoiView

- **Tipo:** concreta
- **Descrizione:** classe che modella la visualizzazione delle informazioni globali su un punto di interesse.
- **Superclassi:**
 - android.support.v4.app.DialogFragment
- **Relazioni con altre classi**
 - ← PoiSelectionView: classe che modella la *view* per la visualizzazione e la possibile selezione dei POI da parte dell'utente.

4.24.2.3 shike::app::view::poi::PoiViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei punti di interesse.
- **Superclassi:**
 - android.widget.BaseAdapter
- **Relazioni con altre classi**
 - ← PoiSelectionView: classe che modella la *view* per la visualizzazione e la possibile selezione dei POI da parte dell'utente.

4.25 shike::app::view::session

4.25.1 Informazioni sulla componente

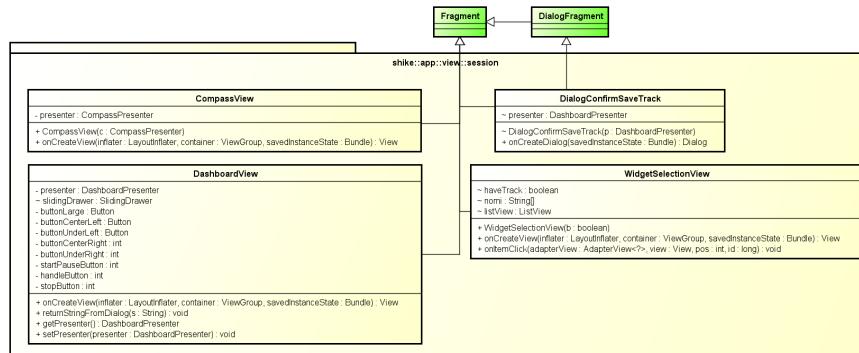


Figura 33: Diagramma di shike::app::view::session

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

4.25.2 Classi

4.25.2.1 shike::app::view::session::DashboardView

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

4.25.2.2 shike::app::view::session::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

4.25.2.3 shike::app::view::session::WidgetSelectionView

- **Tipo:** concreta
- **Descrizione:** vista che visualizza i possibili *widget* utilizzabili.
- **Superclassi:**
 - android.support.v4.app.DialogFragment

4.25.2.4 shike::app::view::session::DialogConfirmSaveTrack

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

4.26 shike::app::view::weather

4.26.1 Informazioni sulla componente

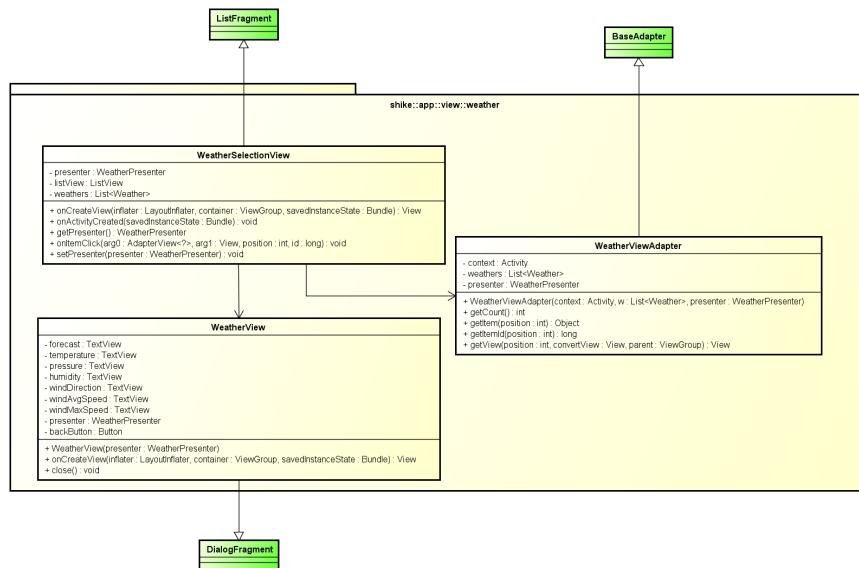


Figura 34: Diagramma di shike::app::view::weather

- **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

4.26.2 Classi

4.26.2.1 shike::app::view::weather::WeatherView

- **Tipo:** concreta
- **Descrizione:** classe che modella la visualizzazione delle informazioni meteo salvate.
- **Superclassi:**
 - android.support.v4.app.DialogFragment
- **Relazioni con altre classi**
 - ← `WeatherSelectionView`: classe che modella la visualizzazione della lista delle informazioni meteo previste.

4.26.2.2 shike::app::view::weather::WeatherSelectionView

- **Tipo:** concreta
- **Descrizione:** classe che modella la visualizzazione della lista delle informazioni meteo previste.
- **Superclassi:**
 - android.support.v4.app.ListFragment
- **Relazioni con altre classi**
 - ← WeatherPresenter: classe che fa da *presenter* per tutte le funzionalità utilizzate per gestire le informazioni sul meteo.
 - → WeatherView: classe che modella la visualizzazione delle informazioni meteo salvate.
 - → WeatherViewAdapter: classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei meteo.

4.26.2.3 shike::app::view::weather::WeatherViewAdapter

- **Tipo:** concreta
- **Descrizione:** classe utilizzata per modellare la visualizzazione di ogni singolo elemento della lista dei meteo.
- **Superclassi:**
 - android.widget.BaseAdapter
- **Relazioni con altre classi**
 - ← WeatherSelectionView: classe che modella la visualizzazione della lista delle informazioni meteo previste.

4.27 shike::app::helper

4.27.1 Informazioni sulla componente

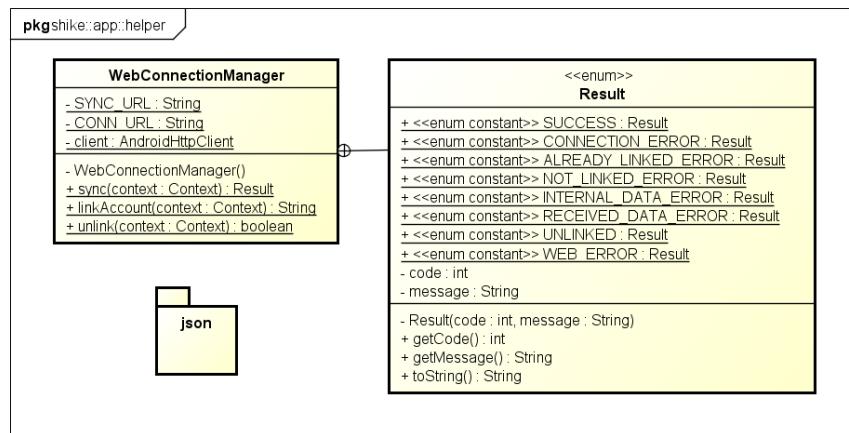


Figura 35: Diagramma di shike::app::helper

- **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

4.27.2 Classi

4.27.2.1 shike::app::helper::WebConnectionManager

- **Tipo:** concreta

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

- **Relazioni con altre classi**

- → Account: classe che modella l' *account* utente collegato al dispositivo.
 - → DbUtil: classe che si occupa di gestire l'inizializzazione e la connessione al *database* dell'applicazione.
 - → AccountService: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dell'*account*.
 - → GeneralService: servizio di gestione generale della memoria dell'applicazione.
 - → HelpNumberService: classe di servizio che fa da ponte tra i presenter e l'interfaccia DAO dei numeri di soccorso.
 - → PerformanceService: classe *service* che fa da ponte tra i presenter e il DAO delle performance dell'utente.
 - → PoiService: classe *service* che fa da ponte tra i presenter e il DAO dei punti di interesse.
 - → TrackWeatherService: classe *service* che fa da ponte tra i *presenter* e il DAO delle previsioni meteo dei percorsi.
 - → VirtualTrackService: classe *service* che fa da ponte tra i *presenter* e il DAO dei percorsi scaricati dalla parte web.
 - → AccountLinkData: classe che modella i dati che vengono ricevuti dalla piattaforma web durante il primo collegamento dell' *account*.
 - → SyncDataWeb: classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
 - → JsonConverter: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.
 - → WebConnectionManager.Result: enum interna a WebConnectionManager che indica i possibili risultati ottenibili da un'operazione di sincronizzazione o collegamento *account*.

4.27.2.2 shike::app::helper::WebConnectionManager.Result

- **Tipo:** concreta
- **Descrizione:** enum interna a WebConnectionManager che indica i possibili risultati ottenibili da un'operazione di sincronizzazione o collegamento *account*.
- **Relazioni con altre classi**
 - ← WebConnectionManager: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.

4.28 shike::app::helper::json

4.28.1 Informazioni sulla componente

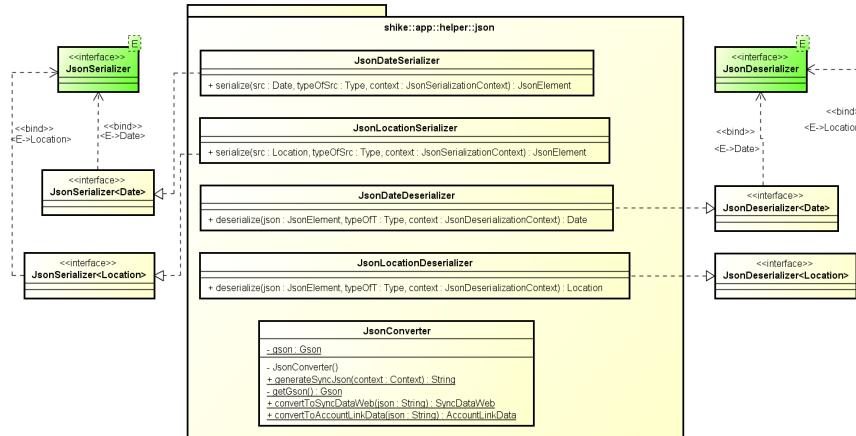


Figura 36: Diagramma di shike::app::helper::json

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

4.28.2 Classi

4.28.2.1 shike::app::helper::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
- **Relazioni con altre classi**
 - ← JsonConverter: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.

4.28.2.2 shike::app::helper::json::JsonDateDeserializer

- **Tipo:** concreta
- **Descrizione:** deserializzatore personalizzato per il tipo `Date` di Java. Creato per via del serializzatore personalizzato, `JsonDateSerializer`, che serializza la data in maniera diversa da quello di default. Implementa `JsonDeserializer` utilizzando `Date` come tipo T.
- **Implementa:**
 - `com.google.gson.JsonDeserializer`
- **Relazioni con altre classi**
 - ← `JsonConverter`: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.

4.28.2.3 shike::app::helper::json::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.
- **Relazioni con altre classi**
 - ← `WebConnectionManager`: classe contenente metodi statici che consentono il collegamento alla piattaforma web, lo scollegamento e la sincronizzazione.
 - → `AccountLinkData`: classe che modella i dati che vengono ricevuti dalla piattaforma web durante il primo collegamento dell' *account*.
 - → `SyncDataApp`: classe che modella i dati che l'applicazione invia alla piattaforma web durante una sincronizzazione.
 - → `SyncDataWeb`: classe che modella le informazioni ricevute dalla piattaforma web durante la sincronizzazione.
 - → `JsonDateSerializer`: serializzatore personalizzato per il tipo `Date` di Java.
 - → `JsonDateDeserializer`: deserializzatore personalizzato per il tipo `Date` di Java.
 - → `JsonLocationSerializer`: serializzatore personalizzato per il tipo `Location` di Android.
 - → `JsonLocationDeserializer`: deserializzatore personalizzato per il tipo `Location` di Android.

4.28.2.4 shike::app::helper::json::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`
- **Relazioni con altre classi**
 - ← `JsonConverter`: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.

4.28.2.5 shike::app::helper::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`
- **Relazioni con altre classi**
 - ← `JsonConverter`: classe che fornisce metodi statici per la conversione di oggetti nel formato JSON e viceversa.

4.29 shike::web

4.29.1 Informazioni sulla componente

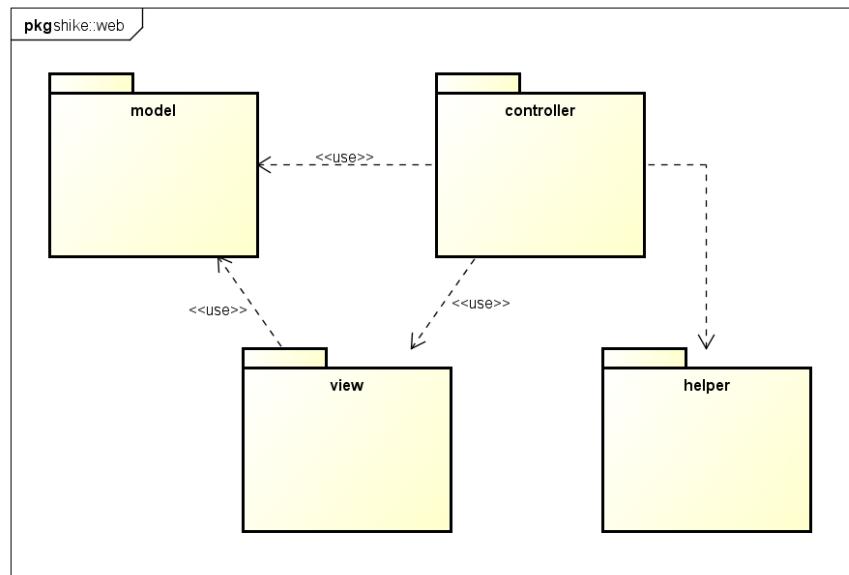


Figura 37: Diagramma di shike::web

- **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`

4.30 shike::web::controller

4.30.1 Informazioni sulla componente

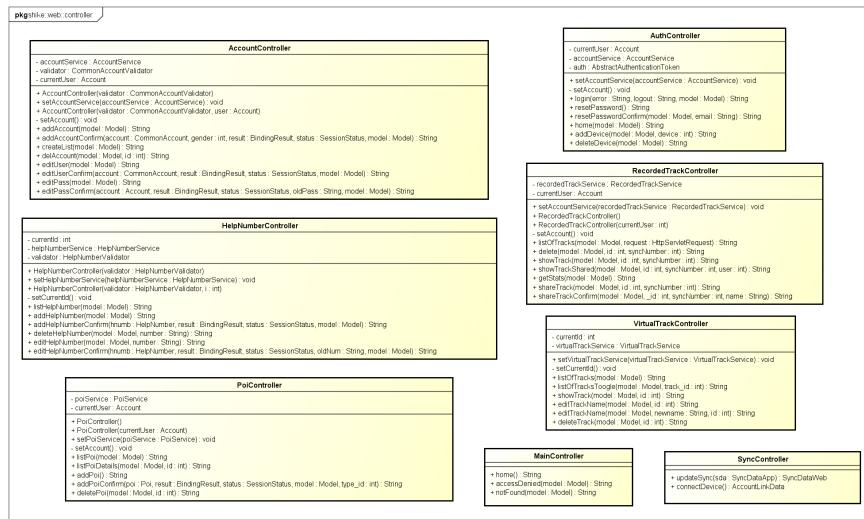


Figura 38: Diagramma di shike::web::controller

- 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

4.30.2 Classi

4.30.2.1 shike::web::controller::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).
- Relazioni con altre classi**
 - → Message: classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
 - → Home: classe descrittiva del portale web, visualizzabile anche senza aver fatto il *login*.
 - → Message: *View* generica che permette la visualizzazione di un messaggio.

4.30.2.2 shike::web::controller::RecordedTrackController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
- **Relazioni con altre classi**
 - → RecordedTrack: classe che modella la struttura di un percorso registrato dall’utente e carico sul web ma non condiviso.
 - → Message: classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
 - → Message: *View* generica che permette la visualizzazione di un messaggio.
 - → DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.
 - → RecordedTrackService: classe che permette di recuperare un riferimento ad un *RecordedTrack* tramite la classe DAO corrispondente.
 - → Stats: schermata di riepilogo contenente le statistiche globali dell’utente.
 - → ShareTrack: vista dei tracciati condivisi.
 - → ListTrack: visualizza la lista dei tracciati presenti.

4.30.2.3 shike::web::controller::PoiController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
- **Relazioni con altre classi**
 - → Poi: classe che modella la struttura di un punto di interesse (POI).
 - → Message: classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
 - → Message: *View* generica che permette la visualizzazione di un messaggio.
 - → ListPoi: vista generica dei POI che visualizza l’elenco di tutti i POI presenti nel *database*.
 - → DetailsPoi: permette di visualizzare i dettagli di un POI.
 - → AddPoi: permette di aggiungere un POI nel sistema.
 - → PoiService: classe che permette di recuperare un riferimento ad un *poi* tramite la classe DAO corrispondente.

4.30.2.4 shike::web::controller::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.
- **Relazioni con altre classi**
 - → Message: classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.

- → Login: pagina che permette l'accesso al portale. Vengono visualizzati anche messaggi di errore o successo riguardanti la procedura di *login*.
- → Dashboard: *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.
- → Dashboard: *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.
- → Message: *View* generica che permette la visualizzazione di un messaggio.
- → MailManager: classe di supporto che permette l'invio di *email*.
- → ResetPassword: *View* che permette l'inserimento della propria *email* alla quale verrà inviata la nuova *password*.
- → AccountService: classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.

4.30.2.5 shike::web::controller::AccountController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
- **Relazioni con altre classi**
 - → CommonAccount: classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.
 - → CommonAccountValidator: 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.
 - → Message: classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
 - → ListCommonAccount: viene visualizzata la lista degli utenti registrati al portale permettendo all'amministratore di eliminare gli *account*.
 - → Message: *View* generica che permette la visualizzazione di un messaggio.
 - → AddCommonAccount: *View* che permette la registrazione di un nuovo utente.
 - → EditUser: *View* che permette la modifica dei dati del proprio profilo. Disponibile solo per gli utenti.
 - → AccountService: classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.

4.30.2.6 shike::web::controller::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.
- **Relazioni con altre classi**
 - → AccountLinkData: classe che modella i dati che sono inviati dalla piattaforma web per il collegamento del dispositivo all' *account*.

- → SyncDataApp: classe che modella i dati inviati dal device alla piattaforma web durante una sincronizzazione.
- → SyncDataWeb: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.

4.30.2.7 shike::web::controller::HelpNumberController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numeri di pronto soccorso.
- **Relazioni con altre classi**
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.
 - → HelpNumberValidator: 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.
 - → Message: classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
 - → Message: *View* generica che permette la visualizzazione di un messaggio.
 - → AddHelpNumber: form che permette l'inserimento di un nuovo numero utile.
 - → EditHelpNumber: permette di modificare un numero utile aggiunto in precedenza.
 - → ListHelpNumbers: lista dei numeri utili associati al proprio profilo.
 - → HelpNumberService: classe che permette di recuperare un riferimento ad un *HelpNumber* tramite la classe DAO corrispondente.

4.30.2.8 shike::web::controller::VirtualTrackController

- **Tipo:** concreta
- **Descrizione:** classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.
- **Relazioni con altre classi**
 - → Share: interfaccia che implementa il *pattern Strategy* relativo alla funzionalità di condivisione di una *performance* e del relativo tracciato sui *social network* (Facebook e Twitter).
 - → VirtualTrackDao: classe che modella il riferimento ad un percorso che è stato condiviso dagli utenti.
 - → Message: *View* generica che permette la visualizzazione di un messaggio.
 - → ListTrack: visualizza la lista dei tracciati presenti.
 - → EditTrack: permette di modificare il nome di un tracciato condiviso.
 - → DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.

4.31 shike::web::view

4.31.1 Informazioni sulla componente

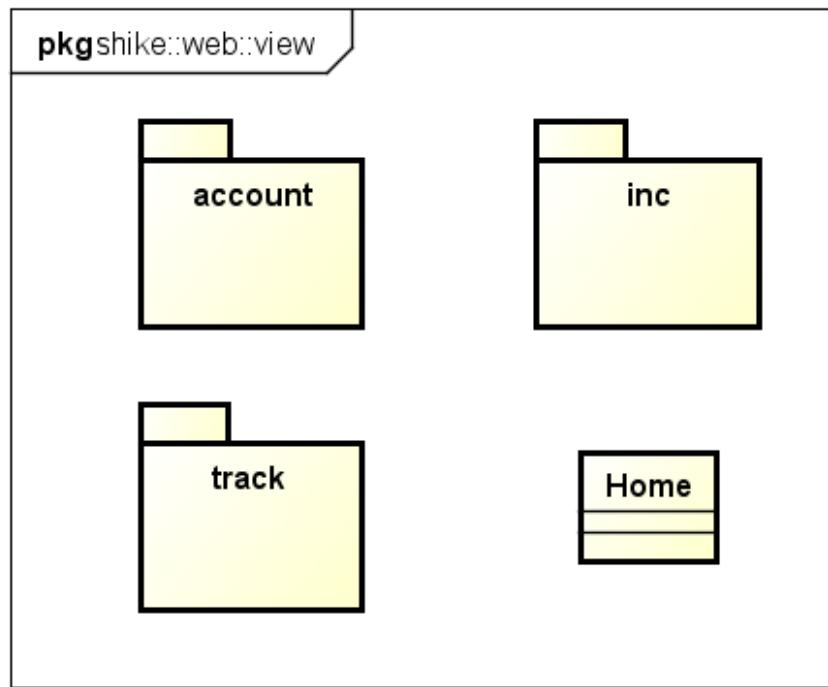


Figura 39: Diagramma di shike::web::view

- **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 delle 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

4.31.2 Classi

4.31.2.1 shike::web::view::Home

- **Tipo:** concreta

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

- **Relazioni con altre classi**

- ← MainController: *Controller* base che permette il *mapping* delle pagine principali della piattaforma come la *home* e quelle di errore (403 e 404).
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.32 shike::web::view::account

4.32.1 Informazioni sulla componente

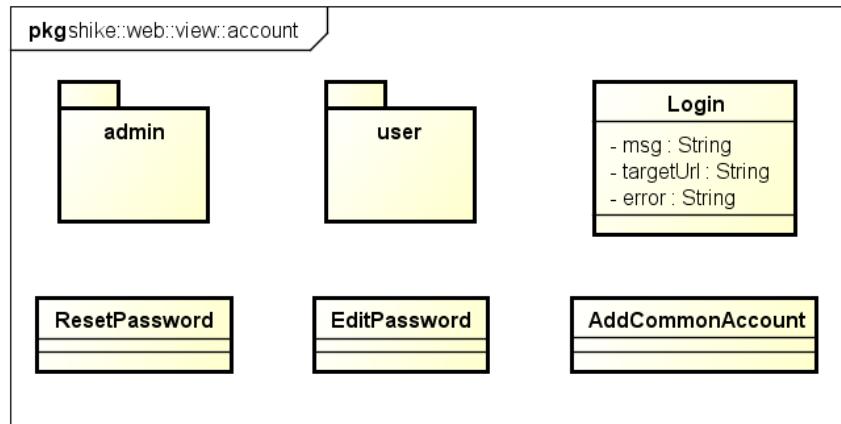


Figura 40: Diagramma di shike::web::view::account

- **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

4.3.2.2 Classi

4.3.2.2.1 shike::web::view::account::ResetPassword

- **Tipo:** concreta
- **Descrizione:** View che permette l'inserimento della propria *email* alla quale verrà inviata la nuova *password*.
- **Relazioni con altre classi**
 - ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.3.2.2.2 shike::web::view::account::Login

- **Tipo:** concreta
- **Descrizione:** pagina che permette l'accesso al portale. Vengono visualizzati anche messaggi di errore o successo riguardanti la procedura di *login*.
- **Relazioni con altre classi**
 - ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.3.2.2.3 shike::web::view::account::AddCommonAccount

- **Tipo:** concreta
- **Descrizione:** View che permette la registrazione di un nuovo utente.
- **Relazioni con altre classi**
 - ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.32.2.4 shike::web::view::account::EditPassword

- **Tipo:** concreta
- **Descrizione:** permette di cambiare la *password*. La *view* è la stessa sia per amministratori che per utenti.
- **Relazioni con altre classi**
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.33 shike::web::view::account::user

4.33.1 Informazioni sulla componente

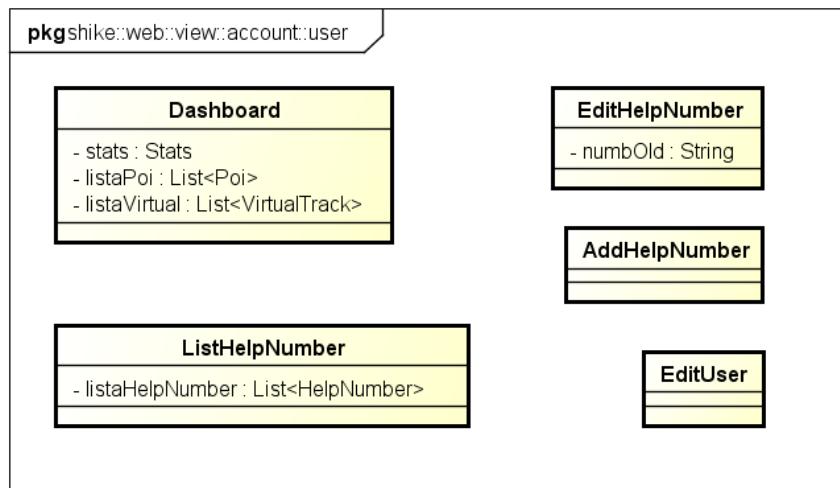


Figura 41: Diagramma di `shike::web::view::account::user`

- **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`

4.33.2 Classi

4.33.2.1 shike::web::view::account::user::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.

- **Relazioni con altre classi**

- ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.33.2.2 shike::web::view::account::user::AddHelpNumber

- **Tipo:** concreta

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

- **Relazioni con altre classi**

- ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numero di pronto soccorso.
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.33.2.3 shike::web::view::account::user::EditHelpNumber

- **Tipo:** concreta

- **Descrizione:** permette di modificare un numero utile aggiunto in precedenza.

- **Relazioni con altre classi**

- ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numero di pronto soccorso.
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.33.2.4 shike::web::view::account::user::EditUser

- **Tipo:** concreta

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

- **Relazioni con altre classi**

- ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.33.2.5 shike::web::view::account::user::ListHelpNumbers

- **Tipo:** concreta
- **Descrizione:** lista dei numeri utili associati al proprio profilo.
- **Relazioni con altre classi**
 - ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numeri di pronto soccorso.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.34 shike::web::view::account::admin

4.34.1 Informazioni sulla componente

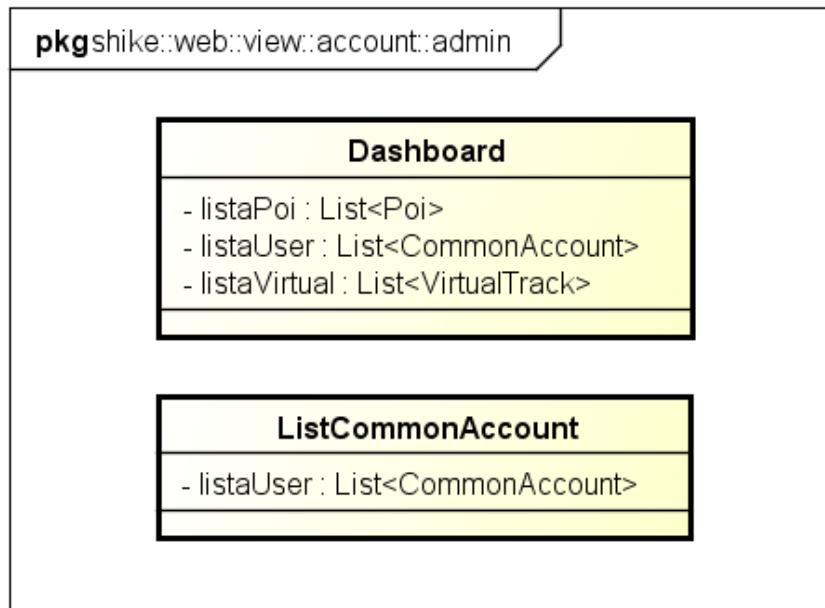


Figura 42: Diagramma di shike::web::view::account::admin

- **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`

4.34.2 Classi

4.34.2.1 shike::web::view::account::admin::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.
- **Relazioni con altre classi**
 - ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.34.2.2 shike::web::view::account::admin::ListCommonAccount

- **Tipo:** concreta
- **Descrizione:** viene visualizzata la lista degli utenti registrati al portale permettendo all'amministratore di eliminare gli *account*.
- **Relazioni con altre classi**
 - ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.35 shike::web::view::track

4.35.1 Informazioni sulla componente

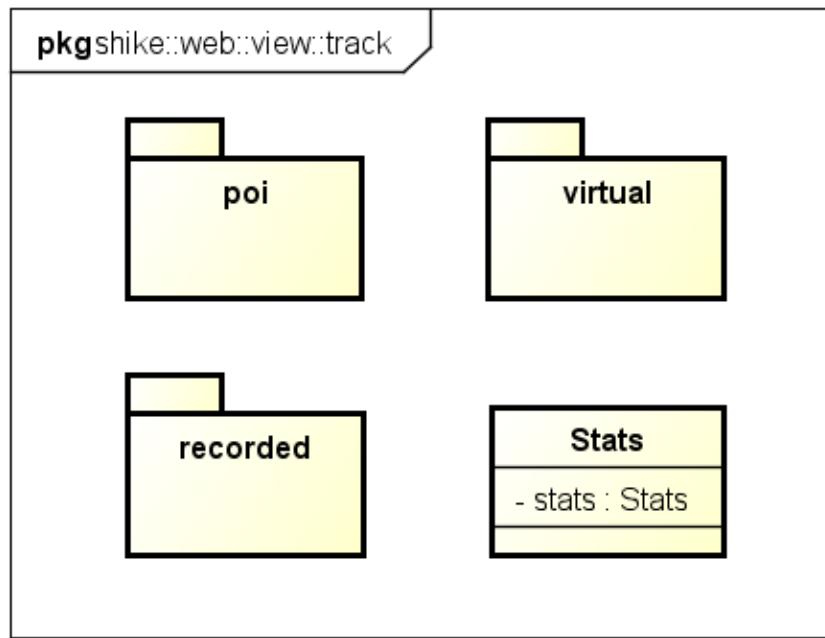


Figura 43: Diagramma di shike::web::view::track

- **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

4.35.2 Classi

4.35.2.1 shike::web::view::track::Stats

- **Tipo:** concreta
- **Descrizione:** schermata di riepilogo contenente le statistiche globali dell'utente.
- **Relazioni con altre classi**
 - ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.

- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.36 shike::web::view::track::poi

4.36.1 Informazioni sulla componente

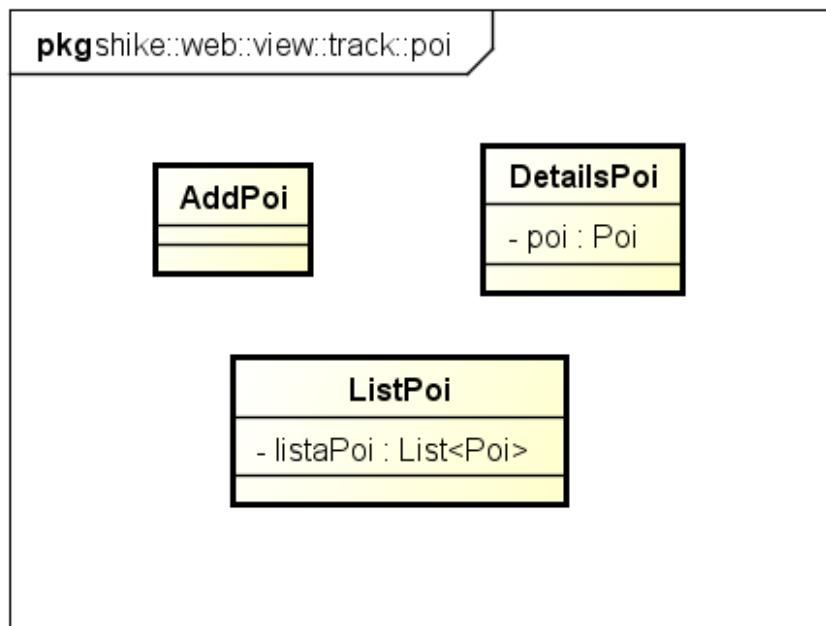


Figura 44: Diagramma di shike::web::view::track::poi

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

4.36.2 Classi

4.36.2.1 shike::web::view::track::poi::DetailsPoi

- **Tipo:** concreta
- **Descrizione:** permette di visualizzare i dettagli di un POI.
- **Relazioni con altre classi**
 - ← PoiController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.36.2.2 shike::web::view::track::poi::AddPoi

- **Tipo:** concreta
- **Descrizione:** permette di aggiungere un POI nel sistema.
- **Relazioni con altre classi**
 - ← PoiController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.36.2.3 shike::web::view::track::poi::ListPoi

- **Tipo:** concreta
- **Descrizione:** vista generica dei POI che visualizza l'elenco di tutti i POI presenti nel *database*.
- **Relazioni con altre classi**
 - ← PoiController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.37 shike::web::view::track::virtual

4.37.1 Informazioni sulla componente

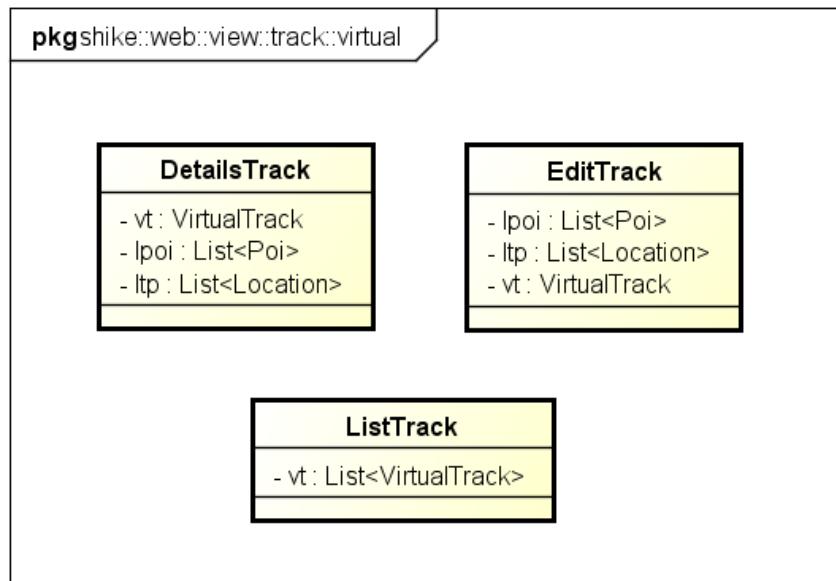


Figura 45: Diagramma di shike::web::view::track::virtual

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

4.37.2 Classi

4.37.2.1 shike::web::view::track::virtual::ListTrack

- **Tipo:** concreta
- **Descrizione:** visualizza la lista dei tracciati presenti.
- **Relazioni con altre classi**
 - ← VirtualTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.37.2.2 shike::web::view::track::virtual::EditTrack

- **Tipo:** concreta
- **Descrizione:** permette di modificare il nome di un tracciato condiviso.
- **Relazioni con altre classi**

- ← VirtualTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.37.2.3 shike::web::view::track::virtual::DetailsTrack

- **Tipo:** concreta

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

- **Relazioni con altre classi**

- ← VirtualTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.38 shike::web::view::track::recorded

4.38.1 Informazioni sulla componente

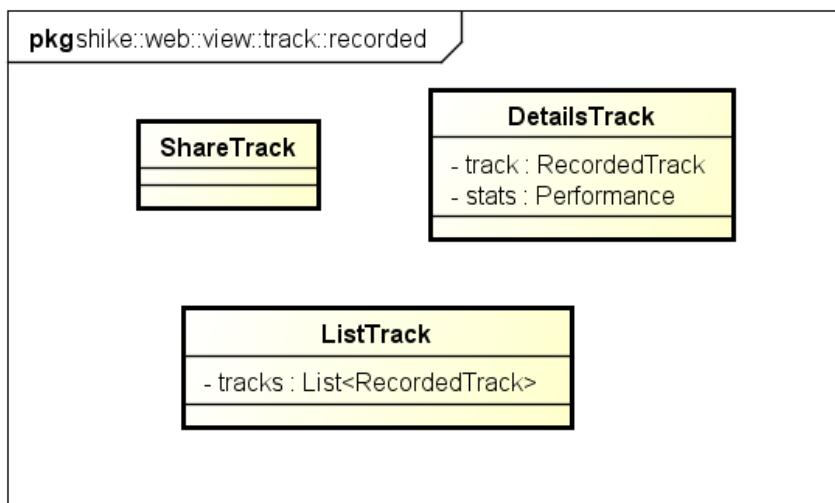


Figura 46: Diagramma di shike::web::view::track::recorded

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

4.38.2 Classi

4.38.2.1 shike::web::view::track::recorded::DetailsTrack

- **Tipo:** concreta
- **Descrizione:** permette di visualizzare in dettaglio i dati di un tracciato.
- **Relazioni con altre classi**
 - ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.38.2.2 shike::web::view::track::recorded::ShareTrack

- **Tipo:** concreta
- **Descrizione:** vista dei tracciati condivisi.
- **Relazioni con altre classi**
 - ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.38.2.3 shike::web::view::track::recorded::ListTrack

- **Tipo:** concreta
- **Descrizione:** visualizza la lista dei tracciati presenti.
- **Relazioni con altre classi**
 - ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
 - → Header: pagina JSP contenente l'header delle pagine web.
 - → Menu: pagina JSP contenente il menu delle pagine web.
 - → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.39 shike::web::view::inc

4.39.1 Informazioni sulla componente

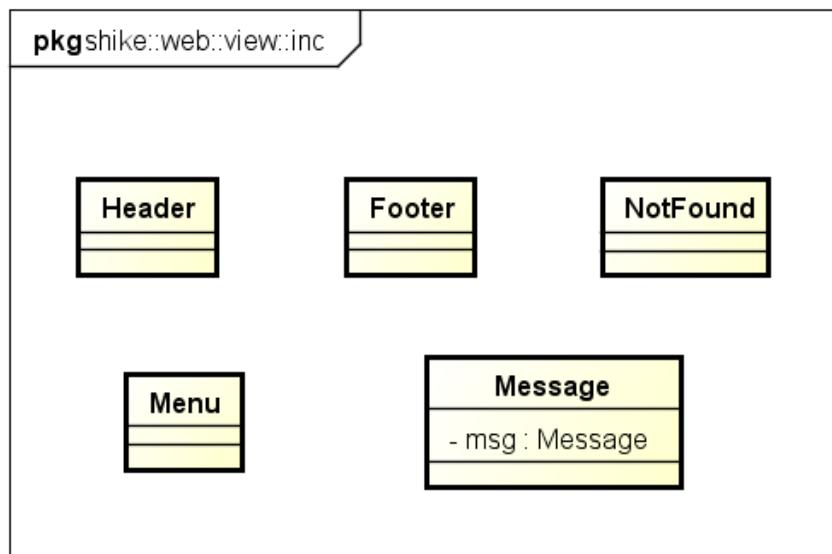


Figura 47: Diagramma di shike::web::view::inc

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

4.39.2 Classi

4.39.2.1 shike::web::view::inc::Footer

- **Tipo:** concreta
- **Descrizione:** parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.
- **Relazioni con altre classi**
 - ← Home: classe descrittiva del portale web, visualizzabile anche senza aver fatto il *login*.
 - ← Login: pagina che permette l'accesso al portale. Vengono visualizzati anche messaggi di errore o successo riguardanti la procedura di *login*.
 - ← Dashboard: *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.
 - ← Dashboard: *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.
 - ← ListCommonAccount: viene visualizzata la lista degli utenti registrati al portale permettendo all'amministratore di eliminare gli *account*.

- ← Message: *View* generica che permette la visualizzazione di un messaggio.
- ← DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.
- ← List Track: visualizza la lista dei tracciati presenti.
- ← AddCommonAccount: *View* che permette la registrazione di un nuovo utente.
- ← AddHelpNumber: form che permette l'inserimento di un nuovo numero utile.
- ← EditHelpNumber: permette di modificare un numero utile aggiunto in precedenza.
- ← EditPassword: permette di cambiare la *password*. La *view* è la stessa sia per amministratori che per utenti.
- ← EditUser: *View* che permette la modifica dei dati del proprio profilo. Disponibile solo per gli utenti.
- ← ListHelpNumbers: lista dei numeri utili associati al proprio profilo.
- ← ResetPassword: *View* che permette l'inserimento della propria *email* alla quale verrà inviata la nuova *password*.
- ← ListPoi: vista generica dei POI che visualizza l'elenco di tutti i POI presenti nel *database*.
- ← DetailsPoi: permette di visualizzare i dettagli di un POI.
- ← AddPoi: permette di aggiungere un POI nel sistema.
- ← Stats: schermata di riepilogo contenente le statistiche globali dell'utente.
- ← EditTrack: permette di modificare il nome di un tracciato condiviso.
- ← DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.
- ← ShareTrack: vista dei tracciati condivisi.
- ← List Track: visualizza la lista dei tracciati presenti.

4.39.2.2 shike::web::view::inc::NotFound

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

4.39.2.3 shike::web::view::inc::Header

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

- ← Home: classe descrittiva del portale web, visualizzabile anche senza aver fatto il *login*.
- ← Login: pagina che permette l'accesso al portale. Vengono visualizzati anche messaggi di errore o successo riguardanti la procedura di *login*.
- ← Dashboard: *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.
- ← Dashboard: *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.
- ← ListCommonAccount: viene visualizzata la lista degli utenti registrati al portale permettendo all'amministratore di eliminare gli *account*.

- ← Message: *View* generica che permette la visualizzazione di un messaggio.
- ← DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.
- ← ListTrack: visualizza la lista dei tracciati presenti.
- ← AddCommonAccount: *View* che permette la registrazione di un nuovo utente.
- ← AddHelpNumber: form che permette l'inserimento di un nuovo numero utile.
- ← EditHelpNumber: permette di modificare un numero utile aggiunto in precedenza.
- ← EditPassword: permette di cambiare la *password*. La *view* è la stessa sia per amministratori che per utenti.
- ← EditUser: *View* che permette la modifica dei dati del proprio profilo. Disponibile solo per gli utenti.
- ← ListHelpNumbers: lista dei numeri utili associati al proprio profilo.
- ← ResetPassword: *View* che permette l'inserimento della propria *email* alla quale verrà inviata la nuova *password*.
- ← ListPoi: vista generica dei POI che visualizza l'elenco di tutti i POI presenti nel *database*.
- ← DetailsPoi: permette di visualizzare i dettagli di un POI.
- ← AddPoi: permette di aggiungere un POI nel sistema.
- ← Stats: schermata di riepilogo contenente le statistiche globali dell'utente.
- ← EditTrack: permette di modificare il nome di un tracciato condiviso.
- ← DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.
- ← ShareTrack: vista dei tracciati condivisi.
- ← ListTrack: visualizza la lista dei tracciati presenti.

4.39.2.4 shike::web::view::inc::Menu

- **Tipo:** concreta

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

- **Relazioni con altre classi**

- ← Home: classe descrittiva del portale web, visualizzabile anche senza aver fatto il *login*.
- ← Login: pagina che permette l'accesso al portale. Vengono visualizzati anche messaggi di errore o successo riguardanti la procedura di *login*.
- ← Dashboard: *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.
- ← Dashboard: *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.
- ← ListCommonAccount: viene visualizzata la lista degli utenti registrati al portale permettendo all'amministratore di eliminare gli *account*.
- ← Message: *View* generica che permette la visualizzazione di un messaggio.
- ← DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.
- ← ListTrack: visualizza la lista dei tracciati presenti.

- ← AddCommonAccount: *View* che permette la registrazione di un nuovo utente.
- ← AddHelpNumber: form che permette l'inserimento di un nuovo numero utile.
- ← EditHelpNumber: permette di modificare un numero utile aggiunto in precedenza.
- ← EditPassword: permette di cambiare la *password*. La *view* è la stessa sia per amministratori che per utenti.
- ← EditUser: *View* che permette la modifica dei dati del proprio profilo. Disponibile solo per gli utenti.
- ← ListHelpNumbers: lista dei numeri utili associati al proprio profilo.
- ← ResetPassword: *View* che permette l'inserimento della propria *email* alla quale verrà inviata la nuova *password*.
- ← ListPoi: vista generica dei POI che visualizza l'elenco di tutti i POI presenti nel *database*.
- ← DetailsPoi: permette di visualizzare i dettagli di un POI.
- ← AddPoi: permette di aggiungere un POI nel sistema.
- ← Stats: schermata di riepilogo contenente le statistiche globali dell'utente.
- ← EditTrack: permette di modificare il nome di un tracciato condiviso.
- ← DetailsTrack: permette di visualizzare in dettaglio i dati di un tracciato.
- ← ShareTrack: vista dei tracciati condivisi.
- ← ListTrack: visualizza la lista dei tracciati presenti.

4.39.2.5 shike::web::view::inc::Message

- **Tipo:** concreta

- **Descrizione:** *View* generica che permette la visualizzazione di un messaggio.

- **Relazioni con altre classi**

- ← MainController: *Controller* base che permette il *mapping* delle pagine principali della piattaforma come la *home* e quelle di errore (403 e 404).
- ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
- ← PoiController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
- ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
- ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
- ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numeri di pronto soccorso.
- ← Message: classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
- ← VirtualTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.
- → Header: pagina JSP contenente l'header delle pagine web.
- → Menu: pagina JSP contenente il menu delle pagine web.
- → Footer: parte finale della pagina contenente principalmente degli script che possono essere caricati per ultimi.

4.40 shike::web::helper

4.40.1 Informazioni sulla componente

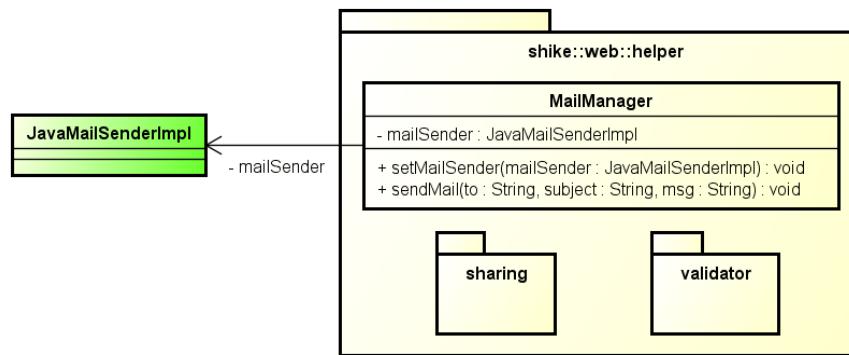


Figura 48: Diagramma di shike::web::helper

- **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

4.40.2 Classi

4.40.2.1 shike::web::helper::MailManager

- **Tipo:** concreta
- **Descrizione:** classe di supporto che permette l'invio di *email*.
- **Relazioni con altre classi**
 - ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.

4.41 shike::web::helper::sharing

4.41.1 Informazioni sulla componente

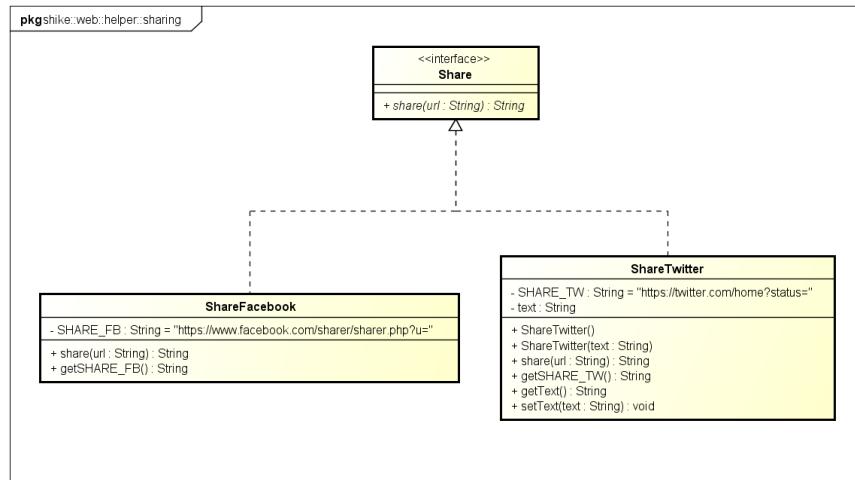


Figura 49: Diagramma di shike::web::helper::sharing

- **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

4.41.2 Classi

4.41.2.1 shike::web::helper::sharing::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:**
 - ShareFacebook:
 - ShareTwitter:
- **Relazioni con altre classi**
 - ← VirtualTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.

4.41.2.2 shike::web::helper::sharing::ShareFacebook

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

4.41.2.3 shike::web::helper::sharing::ShareTwitter

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

4.42 shike::web::helper::validator

4.42.1 Informazioni sulla componente

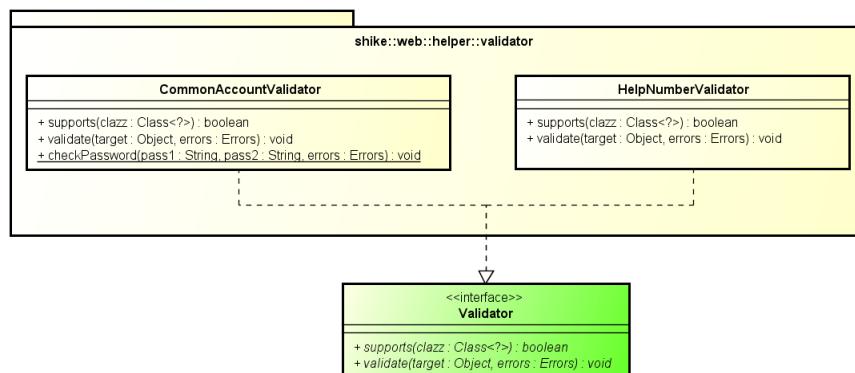


Figura 50: Diagramma di shike::web::helper::validator

- **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

4.42.2 Classi

4.42.2.1 shike::web::helper::validator::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
- **Relazioni con altre classi**
 - ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numero di pronto soccorso.

4.42.2.2 shike::web::helper::validator::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
- **Relazioni con altre classi**
 - ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.

4.43 shike::web::model

4.43.1 Informazioni sulla componente

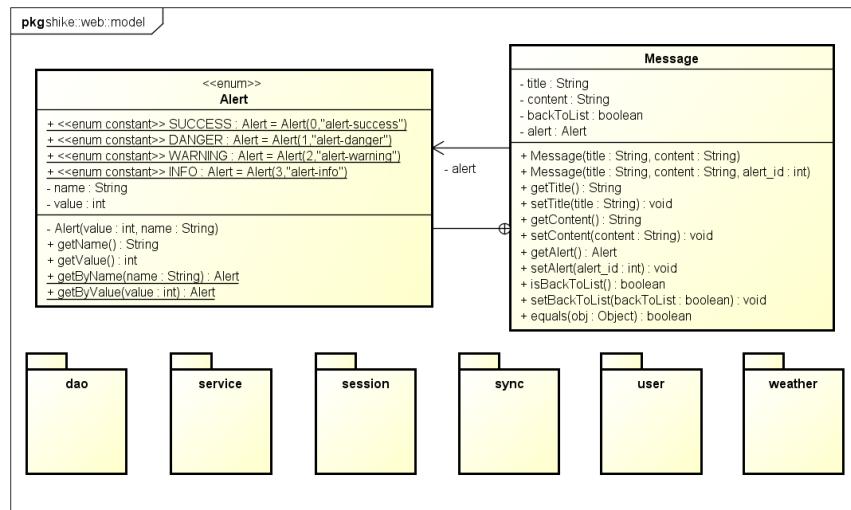


Figura 51: Diagramma di shike::web::model

- **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

4.43.2 Classi

4.43.2.1 shike::web::model::Message.Alert

- **Tipo:** enum
- **Descrizione:** enumeratore interno alla classe Message contenente le varie tipologie di *alert* disponibili.

4.43.2.2 shike::web::model::Message

- **Tipo:** concreta
- **Descrizione:** classe che modella un messaggio. Il messaggio viene poi fornito alla relativa *view* dal *controller* che utilizza il *Message*.
- **Relazioni con altre classi**
 - ← MainController: *Controller* base che permette il *mapping* delle pagine principali della piattaforma come la *home* e quelle di errore (403 e 404).
 - ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
 - ← PoiController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
 - ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
 - ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
 - ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numero di pronto soccorso.
 - → Message: *View* generica che permette la visualizzazione di un messaggio.

4.44 shike::web::model::weather

4.44.1 Informazioni sulla componente

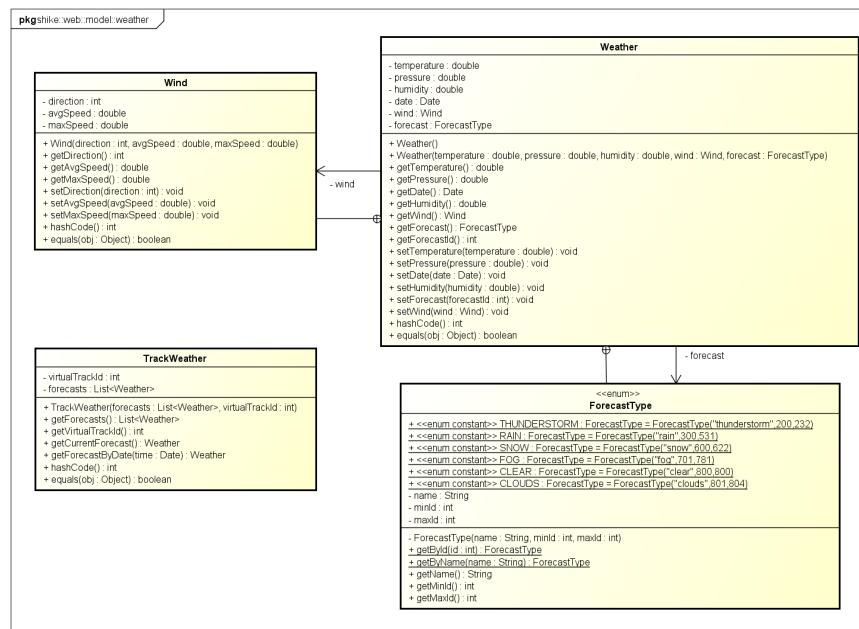


Figura 52: Diagramma di shike::web::model::weather

- **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`

4.44.2 Classi

4.44.2.1 shike::web::model::weather::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.
- **Relazioni con altre classi**
 - ← `VirtualTrack`: classe che modella la struttura di un percorso condiviso nel sito.
 - ← `SyncDataWeb`: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
 - → `Weather`: classe che modella le informazioni sulle condizioni meteo possibili.

4.44.2.2 shike::web::model::weather::Weather

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni sulle condizione meteo possibili.
- **Relazioni con altre classi**
 - ← TrackWeather: 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.
 - ← WeatherDao: classe che modella il riferimento ad una previsione meteo presente nel *database*.
 - ← SyncDataWeb: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
 - → Weather.Wind: classe interna di Weather che rappresenta il vento.

4.44.2.3 shike::web::model::weather::Weather.Wind

- **Tipo:** concreta
- **Descrizione:** classe interna di Weather che rappresenta il vento.
- **Relazioni con altre classi**
 - ← Weather: classe che modella le informazioni sulle condizione meteo possibili.

4.44.2.4 shike::web::model::weather::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>

4.45 shike::web::model::user

4.45.1 Informazioni sulla componente

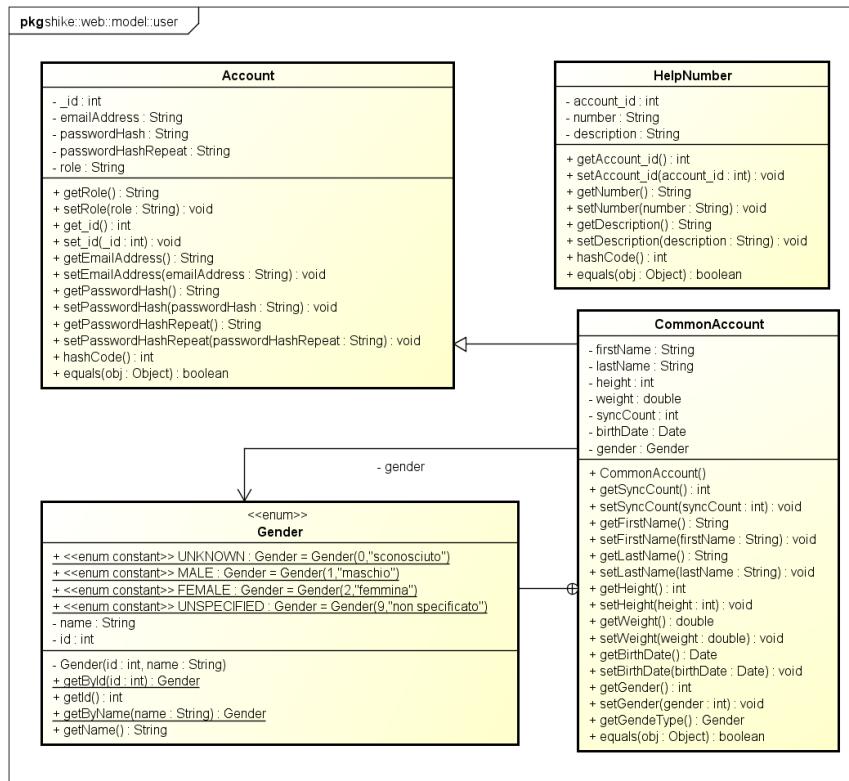


Figura 53: Diagramma di shike::web::model::user

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

4.45.2 Classi

4.45.2.1 shike::web::model::user::Account

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un utente comune, tale classe è identica alla controparte lato applicazione.
- **Sottoclassi:**
 - `CommonAccount`:
- **Relazioni con altre classi**
 - \leftarrow `AccountDao`: classe che modella il riferimento ad un *account*, sia esso utente o amministratore, presente nel *database*.

- ← AccountDaoImpl: classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *AccountDao*.
- ← AccountService: classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.

4.45.2.2 shike::web::model::user::CommonAccount

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.
- **Superclassi:**
 - Account:
- **Relazioni con altre classi**
 - ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
 - ← Performance: classe che modella la struttura di una *performance* svolta da un utente.
 - ← Stats: classe che modella le statistiche generali dell'utente.
 - ← AccountDaoImpl: classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *AccountDao*.
 - ← AccountService: classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.
 - ← SyncDataWeb: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
 - → CommonAccount.Gender: enum che indica i possibili sessi indicati dallo standard ISO 5218.

4.45.2.3 shike::web::model::user::HelpNumber

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.
- **Relazioni con altre classi**
 - ← HelpNumberDao: classe che modella il riferimento ad un numero di telefono di soccorso presente nel *database*.
 - ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numeri di pronto soccorso.
 - ← HelpNumberDaoImpl: classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi dell' *HelpNumberDao*.
 - ← HelpNumberService: classe che permette di recuperare un riferimento ad un *HelpNumber* tramite la classe DAO corrispondente.

4.45.2.4 shike::web::model::user::CommonAccount.Gender

- **Tipo:** enum
- **Descrizione:** enum che indica i possibili sessi indicati dallo standard ISO 5218
- **Relazioni con altre classi**
 - ← CommonAccount: classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.

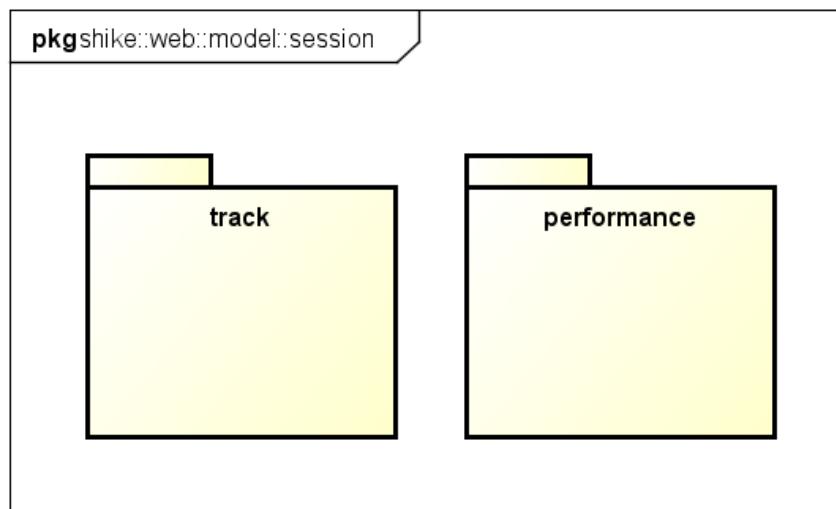
4.46 shike::web::model::session**4.46.1 Informazioni sulla componente**

Figura 54: 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

4.47 shike::web::model::session::track

4.47.1 Informazioni sulla componente

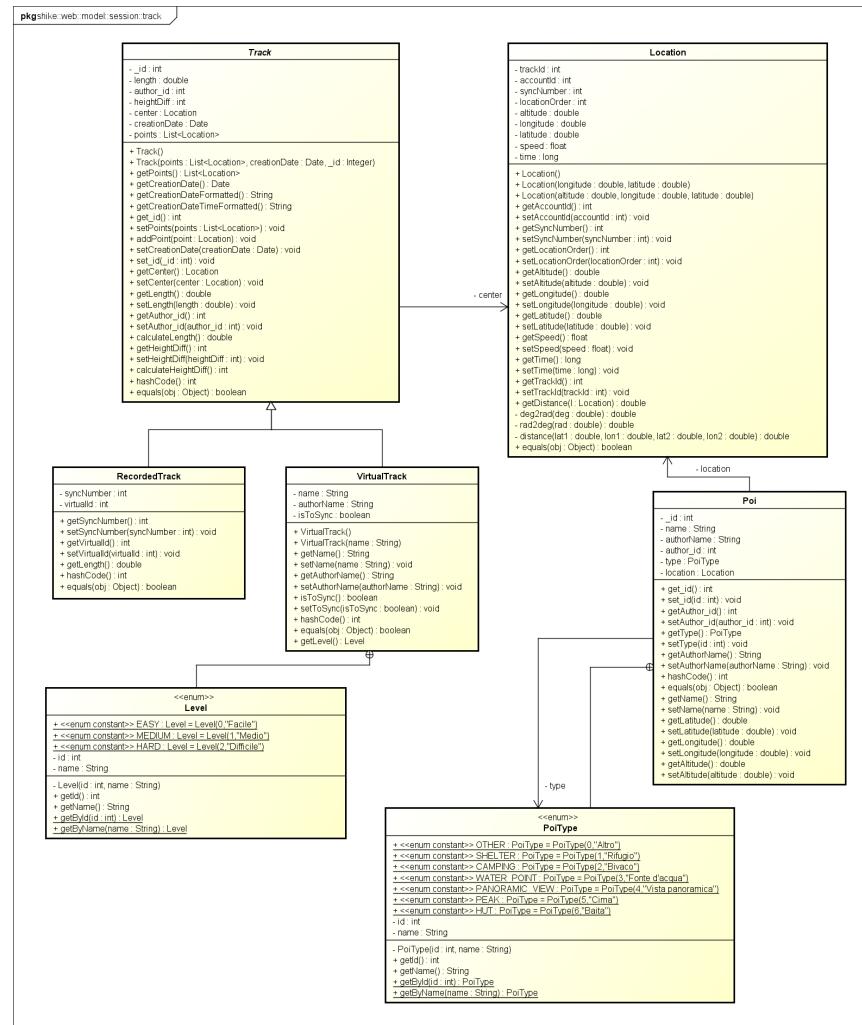


Figura 55: Diagramma di shike::web::model::session::track

- **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

4.47.2 Classi

4.47.2.1 shike::web::model::session::track::Track

- **Tipo:** astratta

- **Descrizione:** interfaccia che modella la struttura dei percorsi.
- **Sottoclassi:**
 - RecordedTrack;
 - VirtualTrack;
- **Relazioni con altre classi**
 - → PoiDaoImpl: classe che implementa l’interfaccia del DAO di un POI. Vengono definiti tutti i metodi del PoiDao.

4.47.2.2 shike::web::model::session::track::RecordedTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un percorso registrato dall’utente e carico sul web ma non condiviso.
- **Superclassi:**
 - Track;
- **Relazioni con altre classi**
 - ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
 - ← RecordedTrackDao: classe che modella il riferimento ad un percorso presente nel *database*.
 - ← RecordedTrackDaoImpl: classe che implementa l’interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.
 - ← RecordedTrackService: classe che permette di recuperare un riferimento ad un *Recorded Track* tramite la classe DAO corrispondente.
 - ← SyncDataWeb: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
 - → PoiDaoImpl: classe che implementa l’interfaccia del DAO di un POI. Vengono definiti tutti i metodi del PoiDao.

4.47.2.3 shike::web::model::session::track::VirtualTrack

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un percorso condiviso nel sito.
- **Superclassi:**
 - Track;
- **Relazioni con altre classi**
 - ← VirtualTrackDao: classe che modella il riferimento ad un percorso che è stato condiviso dagli utenti.
 - ← VirtualTrackDaoImpl: classe che implementa l’interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *VirtualTrackDao*.
 - → Poi: classe che modella la struttura di un punto di interesse (POI).

- → TrackWeather: 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.
- → PoiDaoImpl: classe che implementa l'interfaccia del DAO di un POI. Vengono definiti tutti i metodi del PoiDao.
- → VirtualTrack.Level: enumerazione dei livelli disponibili del percorso.

4.47.2.4 shike::web::model::session::track::Poi

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di un punto di interesse (POI).
- **Relazioni con altre classi**
 - ← PoiController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
 - ← VirtualTrack: classe che modella la struttura di un percorso condiviso nel sito.
 - ← PoiDao: classe che modella il riferimento ad un punto di interesse (POI) presente nel *database*.
 - ← PoiService: classe che permette di recuperare un riferimento ad un *poi* tramite la classe DAO corrispondente.
 - ← SyncDataWeb: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
 - → PoiDaoImpl: classe che implementa l'interfaccia del DAO di un POI. Vengono definiti tutti i metodi del PoiDao.
 - → Poi.PoiType: enumeratore interno alla classe Poi contenente le varie tipologie di POI.

4.47.2.5 shike::web::model::session::track::Location

- **Tipo:** concreta
- **Descrizione:** classe che determina un punto nello spazio, corrispondente alla classe *Location* di Android.
- **Relazioni con altre classi**
 - ← RecordedTrackDao: classe che modella il riferimento ad un percorso presente nel *database*.
 - ← VirtualTrackDao: classe che modella il riferimento ad un percorso che è stato condiviso dagli utenti.
 - ← VirtualTrackDaoImpl: classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *VirtualTrackDao*.
 - ← RecordedTrackDaoImpl: classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.

4.47.2.6 shike::web::model::session::track::Poi.PoiType

- **Tipo:** enum
- **Descrizione:** enumeratore interno alla classe Poi contenente le varie tipologie di POI.
- **Relazioni con altre classi**
 - ← Poi: classe che modella la struttura di un punto di interesse (POI).

4.47.2.7 shike::web::model::session::track::VirtualTrack.Level

- **Tipo:** enum
- **Descrizione:** enumerazione dei livelli disponibili del percorso.
- **Relazioni con altre classi**
 - ← VirtualTrack: classe che modella la struttura di un percorso condiviso nel sito.

4.48 shike::web::model::session::performance

4.48.1 Informazioni sulla componente

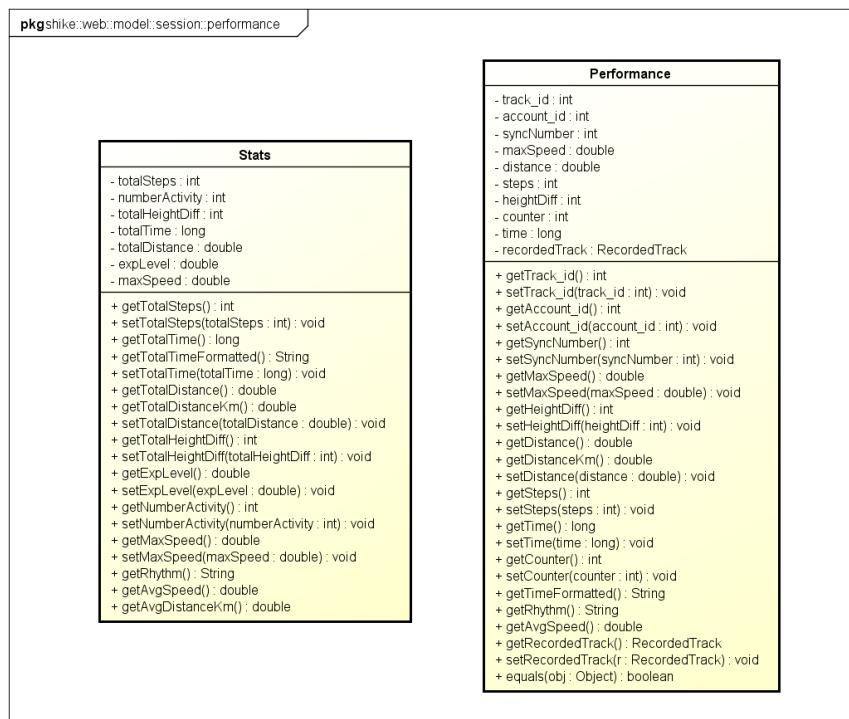


Figura 56: Diagramma di shike::web::model::session::performance

- **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

4.48.2 Classi

4.48.2.1 shike::web::model::session::performance::Performance

- **Tipo:** concreta
- **Descrizione:** classe che modella la struttura di una performance svolta da un utente.

- **Relazioni con altre classi**

- ← RecordedTrackDao: classe che modella il riferimento ad un percorso presente nel *database*.
- ← RecordedTrackService: classe che permette di recuperare un riferimento ad un *Recorded Track* tramite la classe DAO corrispondente.
- ← SyncDataWeb: classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
- → CommonAccount: classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.

4.48.2.2 shike::web::model::session::performance::Stats

- **Tipo:** concreta

- **Descrizione:** classe che modella le statistiche generali dell'utente.

- **Relazioni con altre classi**

- ← RecordedTrackDao: classe che modella il riferimento ad un percorso presente nel *database*.
- ← RecordedTrackService: classe che permette di recuperare un riferimento ad un *Recorded Track* tramite la classe DAO corrispondente.
- → CommonAccount: classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.

4.49 shike::web::model::dao

4.49.1 Informazioni sulla componente

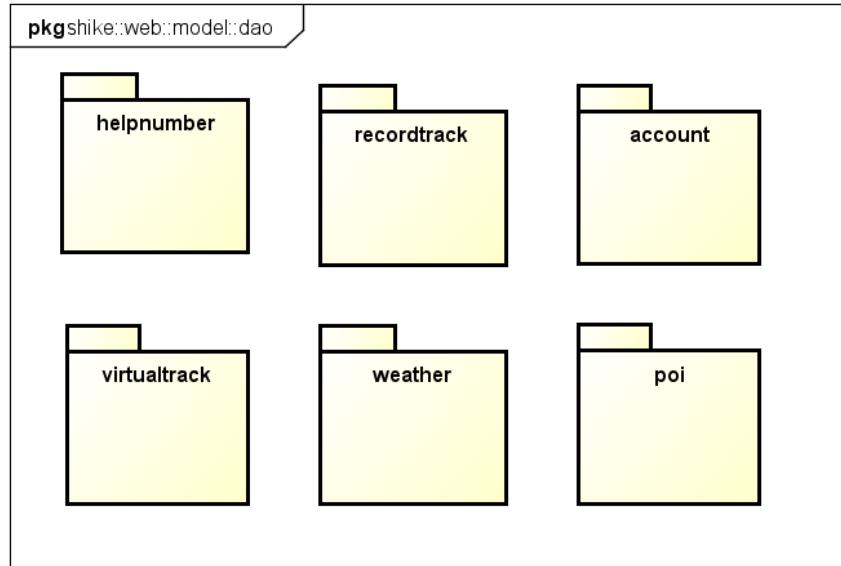


Figura 57: Diagramma di shike::web::model::dao

- **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

4.50 shike::web::model::dao::account

4.50.1 Informazioni sulla componente

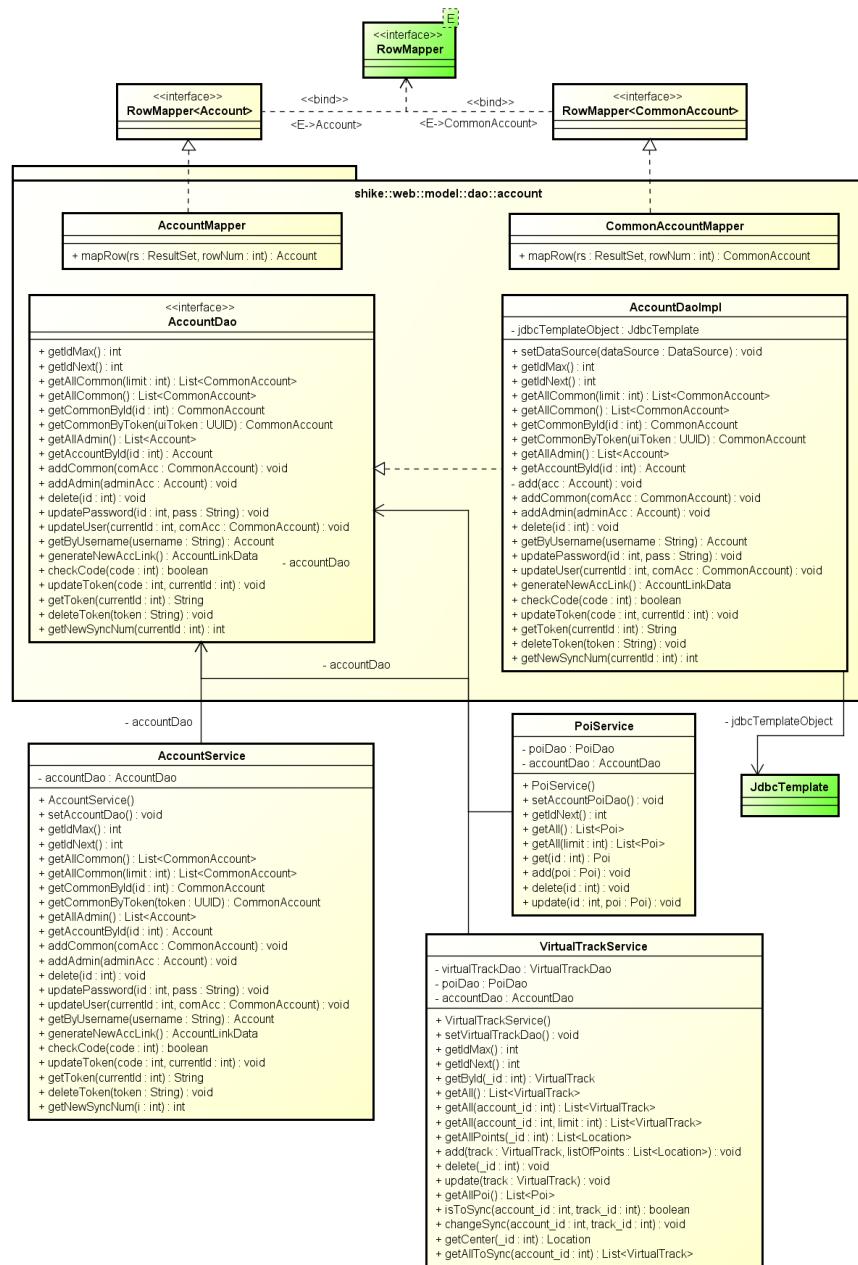


Figura 58: Diagramma di shike::web::model::dao::account

- **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

4.50.2 Classi

4.50.2.1 shike::web::model::dao::account::CommonAccountMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *CommonAccount*.
- **Implementa:**
 - org.springframework.jdbc.core.RowMapper
- **Relazioni con altre classi**
 - ← AccountDaoImpl: classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *AccountDao*.

4.50.2.2 shike::web::model::dao::account::AccountDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un *account*, sia esso utente o amministratore, presente nel *database*.
- **Implementata da:**
 - AccountDaoImpl:
- **Relazioni con altre classi**
 - ← AccountService: classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.
 - → Account: classe che modella la struttura di un utente comune, tale classe è identica alla controparte lato applicazione.

4.50.2.3 shike::web::model::dao::account::AccountMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *Account*.
- **Implementa:**
 - org.springframework.jdbc.core.RowMapper
- **Relazioni con altre classi**
 - ← AccountDaoImpl: classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *AccountDao*.

4.50.2.4 shike::web::model::dao::account::AccountDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *AccountDao*.
- **Implementa:**
 - AccountDao:
- **Relazioni con altre classi**
 - → Account: classe che modella la struttura di un utente comune, tale classe è identica alla controparte lato applicazione.
 - → CommonAccount: classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.
 - → CommonAccountMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *CommonAccount*.
 - → AccountMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *Account*.

4.51 shike::web::model::dao::helpnumber

4.51.1 Informazioni sulla componente

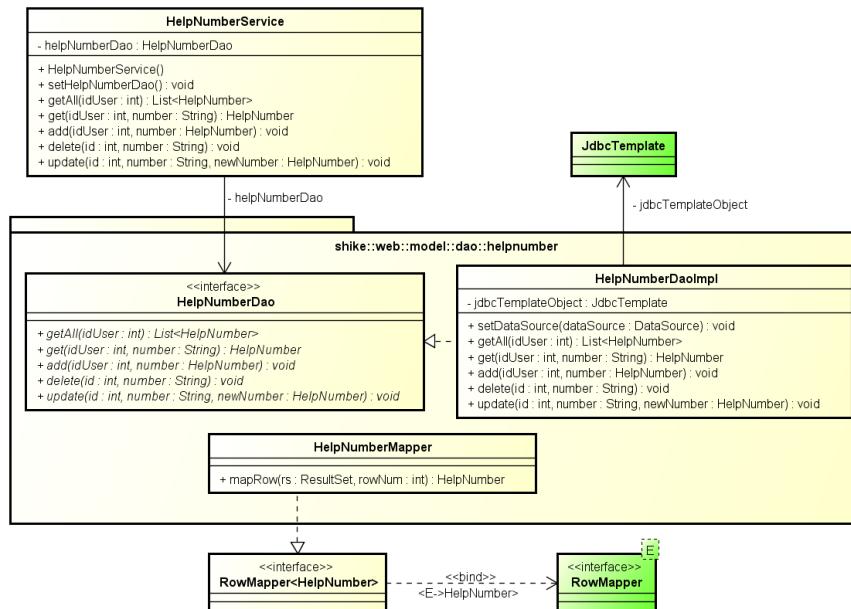


Figura 59: Diagramma di shike::web::model::dao::helpnumber

- **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`

4.51.2 Classi

4.51.2.1 shike::web::model::dao::helpnumber::HelpNumberDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un numero di telefono di soccorso presente nel *database*.
- **Implementata da:**
 - HelpNumberDaoImpl:
- **Relazioni con altre classi**
 - ← HelpNumberService: classe che permette di recuperare un riferimento ad un *HelpNumber* tramite la classe DAO corrispondente.
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.

4.51.2.2 shike::web::model::dao::helpnumber::HelpNumberMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *HelpNumber*.
- **Implementa:**
 - org.springframework.jdbc.core.RowMapper
- **Relazioni con altre classi**
 - ← HelpNumberDaoImpl: classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi dell' *HelpNumberDao*.

4.51.2.3 shike::web::model::dao::helpnumber::HelpNumberDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi dell' *HelpNumberDao*.
- **Implementa:**
 - HelpNumberDao:
- **Relazioni con altre classi**
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.
 - → HelpNumberMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *HelpNumber*.

4.52 shike::web::model::dao::performance

4.52.1 Informazioni sulla componente

- 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

4.53 shike::web::model::dao::poi

4.53.1 Informazioni sulla componente

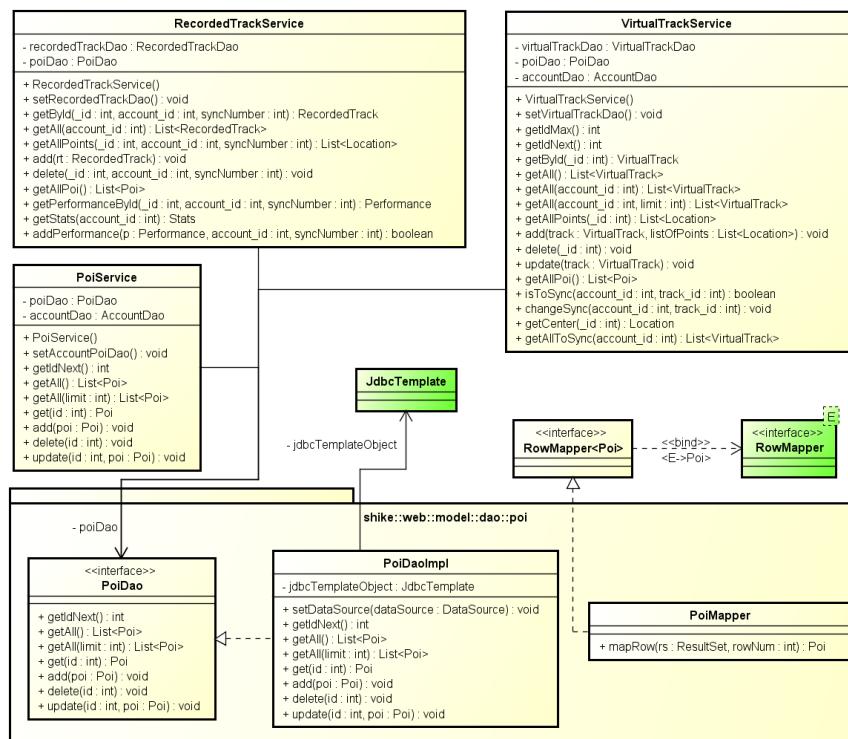


Figura 60: Diagramma di shike::web::model::dao::poi

- 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

4.53.2 Classi

4.53.2.1 shike::web::model::dao::poi::PoiDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un punto di interesse (POI) presente nel *database*.
- **Implementata da:**
 - PoiDaoImpl:
- **Relazioni con altre classi**
 - ← PoiService: classe che permette di recuperare un riferimento ad un *poi* tramite la classe DAO corrispondente.
 - → Poi: classe che modella la struttura di un punto di interesse (POI).

4.53.2.2 shike::web::model::dao::poi::PoiMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un POI.
- **Implementa:**
 - org.springframework.jdbc.core.RowMapper
- **Relazioni con altre classi**
 - ← PoiDaoImpl: classe che implementa l'interfaccia del DAO di un POI. Vengono definiti tutti i metodi del PoiDao.

4.53.2.3 shike::web::model::dao::poi::PoiDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un POI. Vengono definiti tutti i metodi del PoiDao.
- **Implementa:**
 - PoiDao:
- **Relazioni con altre classi**
 - ← Track: interfaccia che modella la struttura dei percorsi.
 - ← RecordedTrack: classe che modella la struttura di un percorso registrato dall'utente e carico sul web ma non condiviso.
 - ← VirtualTrack: classe che modella la struttura di un percorso condiviso nel sito.
 - ← Poi: classe che modella la struttura di un punto di interesse (POI).
 - → PoiMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un POI.

4.54 shike::web::model::dao::recordtrack

4.54.1 Informazioni sulla componente

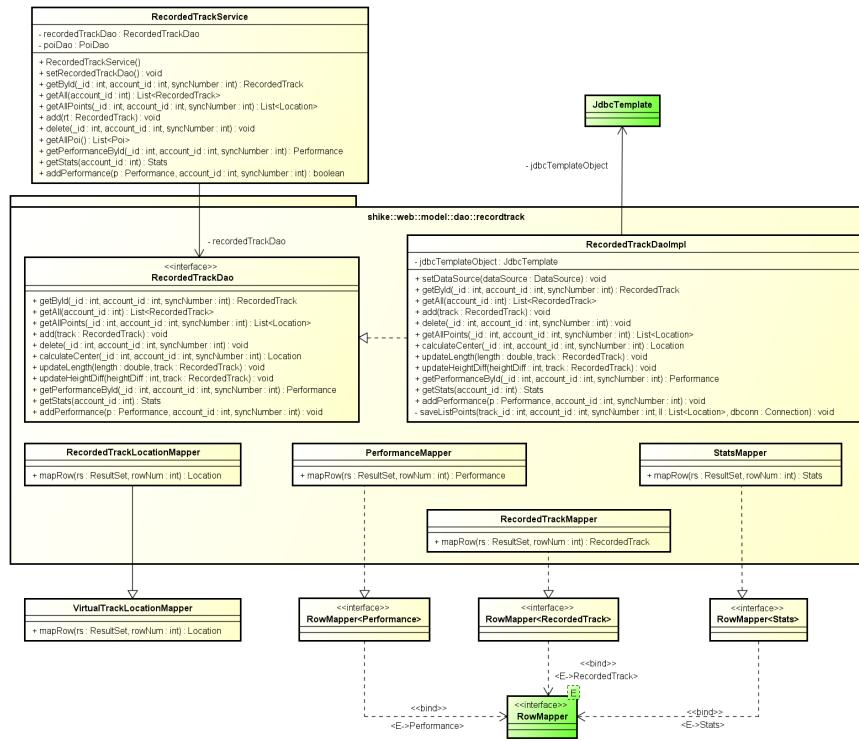


Figura 61: Diagramma di shike::web::model::dao::recordtrack

- 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

4.54.2 Classi

4.54.2.1 shike::web::model::dao::recordtrack::RecordedTrackDao

- Tipo:** interfaccia
- Descrizione:** classe che modella il riferimento ad un percorso presente nel *database*.
- Implementata da:**
 - RecordedTrackDaoImpl:
- Relazioni con altre classi**
 - ← RecordedTrackService: classe che permette di recuperare un riferimento ad un *Recorded Track* tramite la classe DAO corrispondente.
 - Performance: classe che modella la struttura di una *performance* svolta da un utente.

- → RecordedTrack: classe che modella la struttura di un percorso registrato dall’utente e carico sul web ma non condiviso.
- → Stats: classe che modella le statistiche generali dell’utente.
- → Location: classe che determina un punto nello spazio, corrispondente alla classe *Location* di Android.

4.54.2.2 shike::web::model::dao::recordtrack::PerformanceMapper

- **Tipo:** concreta
- **Descrizione:** mapper che permette il collegamento al DAO per prelevare i dati di una *Performance*.
- **Implementa:**
 - org.springframework.jdbc.core.RowMapper
- **Relazioni con altre classi**
 - ← RecordedTrackDaoImpl: classe che implementa l’interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.

4.54.2.3 shike::web::model::dao::recordtrack::RecordedTrackDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l’interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.
- **Implementa:**
 - RecordedTrackDao:
- **Relazioni con altre classi**
 - → RecordedTrack: classe che modella la struttura di un percorso registrato dall’utente e carico sul web ma non condiviso.
 - → PerformanceMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di una *Performance*.
 - → Location: classe che determina un punto nello spazio, corrispondente alla classe *Location* di Android.
 - → RecordedTrackMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *RecordedTrack*.
 - → RecordedTrackLocationMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *RecordedTrackLocation*.
 - → StatsMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *Stats*.

4.54.2.4 shike::web::model::dao::recordtrack::RecordedTrackMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *RecordedTrack*.
- **Implementa:**

– org.springframework.jdbc.core.RowMapper

- **Relazioni con altre classi**

– ← RecordedTrackDaoImpl: classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.

4.54.2.5 shike::web::model::dao::recordtrack::RecordedTrackLocationMapper

- **Tipo:** concreta

- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *RecordedTrackLocation*.

- **Implementa:**

– org.springframework.jdbc.core.RowMapper

- **Relazioni con altre classi**

– ← RecordedTrackDaoImpl: classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.

4.54.2.6 shike::web::model::dao::recordtrack::StatsMapper

- **Tipo:** concreta

- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *Stats*.

- **Implementa:**

– org.springframework.jdbc.core.RowMapper

- **Relazioni con altre classi**

– ← RecordedTrackDaoImpl: classe che implementa l'interfaccia del DAO di un numero utile. Vengono definiti tutti i metodi di *RecordedTrackDao*.

4.55 shike::web::model::dao::weather

4.55.1 Informazioni sulla componente

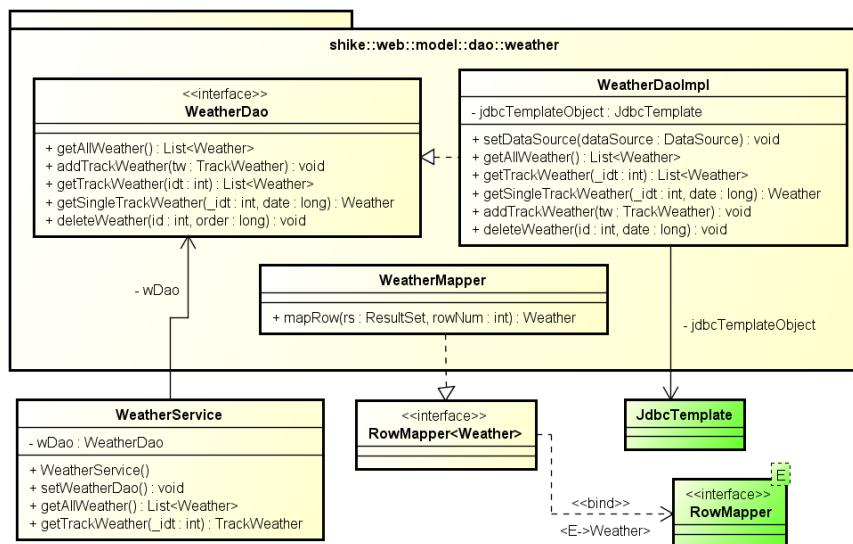


Figura 62: Diagramma di shike::web::model::dao::weather

- **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

4.55.2 Classi

4.55.2.1 shike::web::model::dao::weather::WeatherDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad una previsione meteo presente nel *database*.
- **Implementata da:**
 - WeatherDaoImpl:
- **Relazioni con altre classi**
 - → Weather: classe che modella le informazioni sulle condizioni meteo possibili.

4.55.2.2 shike::web::model::dao::weather::WeatherMapper

- **Tipo:** concreta
- **Descrizione:** mapper che permette il collegamento al DAO per prelevare i dati di un Weather.

- **Implementa:**

- org.springframework.jdbc.core.RowMapper

4.55.2.3 shike::web::model::dao::weather::WeatherDaoImpl

- **Tipo:** concreta

- **Descrizione:** classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *WeatherDao*.

- **Implementa:**

- WeatherDao:

4.56 shike::web::model::dao::virtualtrack

4.56.1 Informazioni sulla componente

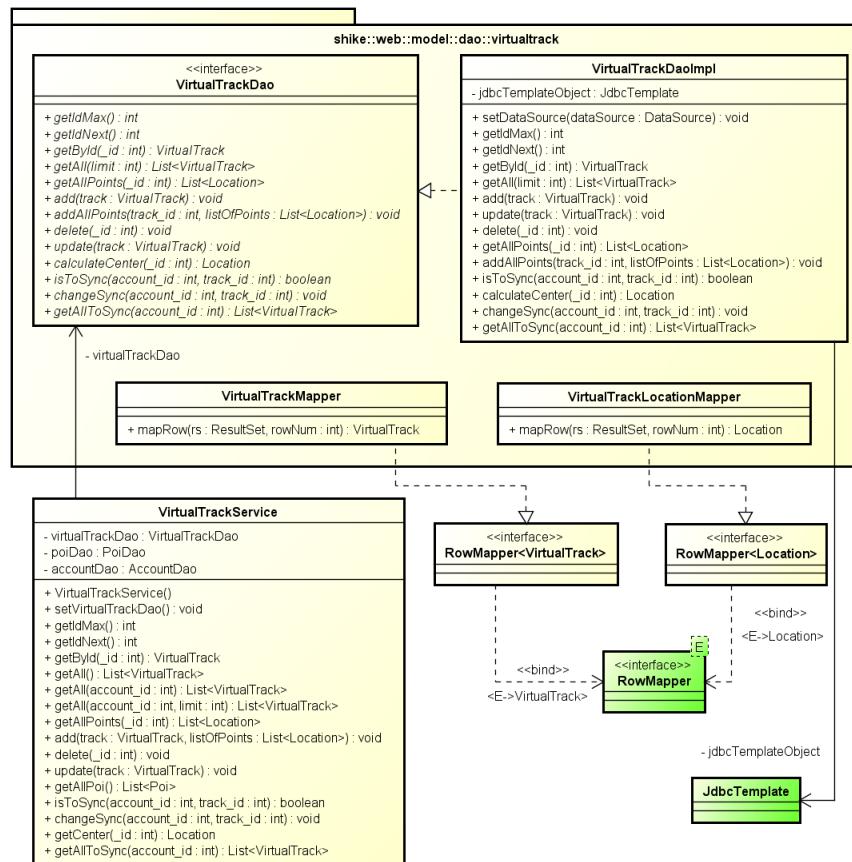


Figura 63: Diagramma di shike::web::model::dao::virtualtrack

- **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

4.56.2 Classi

4.56.2.1 shike::web::model::dao::virtualtrack::VirtualTrackDao

- **Tipo:** interfaccia
- **Descrizione:** classe che modella il riferimento ad un percorso che è stato condiviso dagli utenti.
- **Implementata da:**
 - VirtualTrackDaoImpl:
- **Relazioni con altre classi**
 - ← VirtualTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi condivisi nella piattaforma.
 - → VirtualTrack: classe che modella la struttura di un percorso condiviso nel sito.
 - → Location: classe che determina un punto nello spazio, corrispondente alla classe *Location* di Android.

4.56.2.2 shike::web::model::dao::virtualtrack::VirtualTrackMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *VirtualTrack*.
- **Implementa:**
 - org.springframework.jdbc.core.RowMapper
- **Relazioni con altre classi**
 - ← VirtualTrackDaoImpl: classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *VirtualTrackDao*.

4.56.2.3 shike::web::model::dao::virtualtrack::VirtualTrackLocationMapper

- **Tipo:** concreta
- **Descrizione:** *Mapper* che permette il collegamento al DAO per prelevare i dati di un *VirtualTrackLocation*.
- **Implementa:**
 - org.springframework.jdbc.core.RowMapper
- **Relazioni con altre classi**
 - ← VirtualTrackDaoImpl: classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *VirtualTrackDao*.

4.56.2.4 shike::web::model::dao::virtualtrack::VirtualTrackDaoImpl

- **Tipo:** concreta
- **Descrizione:** classe che implementa l'interfaccia del DAO di un *account*. Vengono definiti tutti i metodi di *VirtualTrackDao*.
- **Implementa:**
 - VirtualTrackDao:
- **Relazioni con altre classi**
 - → VirtualTrack: classe che modella la struttura di un percorso condiviso nel sito.
 - → Location: classe che determina un punto nello spazio, corrispondente alla classe *Location* di Android.
 - → VirtualTrackMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *VirtualTrack*.
 - → VirtualTrackLocationMapper: *Mapper* che permette il collegamento al DAO per prelevare i dati di un *VirtualTrackLocation*.

4.57 shike::web::model::service

4.57.1 Informazioni sulla componente

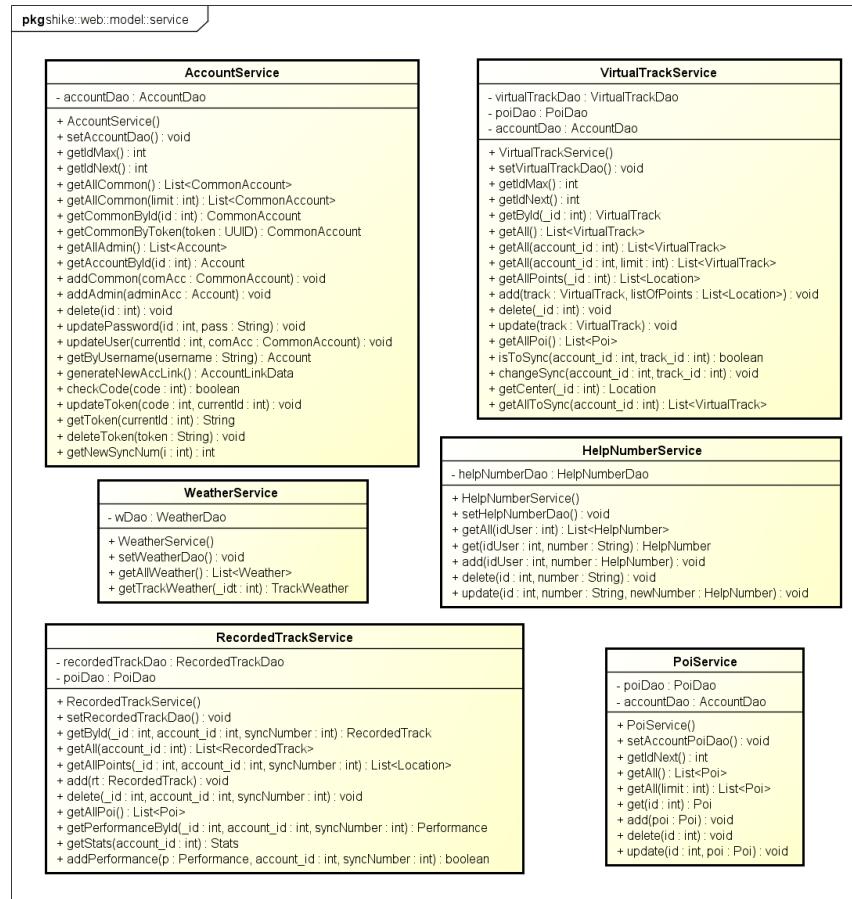


Figura 64: Diagramma di shike::web::model::service

- **Descrizione:** componente contenente i servizi che permettono ai controller di accedere ai DAO restituendo l'oggetto desiderato.
- **Componente padre:** shike::web::model

4.57.2 Classi

4.57.2.1 shike::web::model::service::VirtualTrackService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *VirtualTrack* tramite la classe DAO corrispondente.

4.57.2.2 shike::web::model::service::WeatherService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *Weather* tramite la classe DAO corrispondente.

4.57.2.3 shike::web::model::service::AccountService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.
- **Relazioni con altre classi**
 - ← AuthController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di autentificazione dei dati sulla piattaforma web.
 - ← AccountController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti la gestione delle informazioni personali dell'utente.
 - → Account: classe che modella la struttura di un utente comune, tale classe è identica alla controparte lato applicazione.
 - → CommonAccount: classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.
 - → AccountDao: classe che modella il riferimento ad un *account*, sia esso utente o amministratore, presente nel *database*.
 - → AccountLinkData: classe che modella i dati che sono inviati dalla piattaforma web per il collegamento del dispositivo all' *account*.
 - → SyncDataApp: classe che modella i dati inviati dal device alla piattaforma web durante una sincronizzazione.

4.57.2.4 shike::web::model::service::HelpNumberService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *HelpNumber* tramite la classe DAO corrispondente.
- **Relazioni con altre classi**
 - ← HelpNumberController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i numero di pronto soccorso.
 - → HelpNumberDao: classe che modella il riferimento ad un numero di telefono di soccorso presente nel *database*.
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.

4.57.2.5 shike::web::model::service::PoiService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *poi* tramite la classe DAO corrispondente.
- **Relazioni con altre classi**
 - ← PoiController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i punti di interesse (POI).
 - → Poi: classe che modella la struttura di un punto di interesse (POI).
 - → PoiDao: classe che modella il riferimento ad un punto di interesse (POI) presente nel *database*.

4.57.2.6 shike::web::model::service::RecordedTrackService

- **Tipo:** concreta
- **Descrizione:** classe che permette di recuperare un riferimento ad un *RecordedTrack* tramite la classe DAO corrispondente.
- **Relazioni con altre classi**
 - ← RecordedTrackController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti i percorsi.
 - → Performance: classe che modella la struttura di una *performance* svolta da un utente.
 - → RecordedTrack: classe che modella la struttura di un percorso registrato dall'utente e carico sul web ma non condiviso.
 - → Stats: classe che modella le statistiche generali dell'utente.
 - → RecordedTrackDao: classe che modella il riferimento ad un percorso presente nel *database*.

4.58 shike::web::model::sync

4.58.1 Informazioni sulla componente

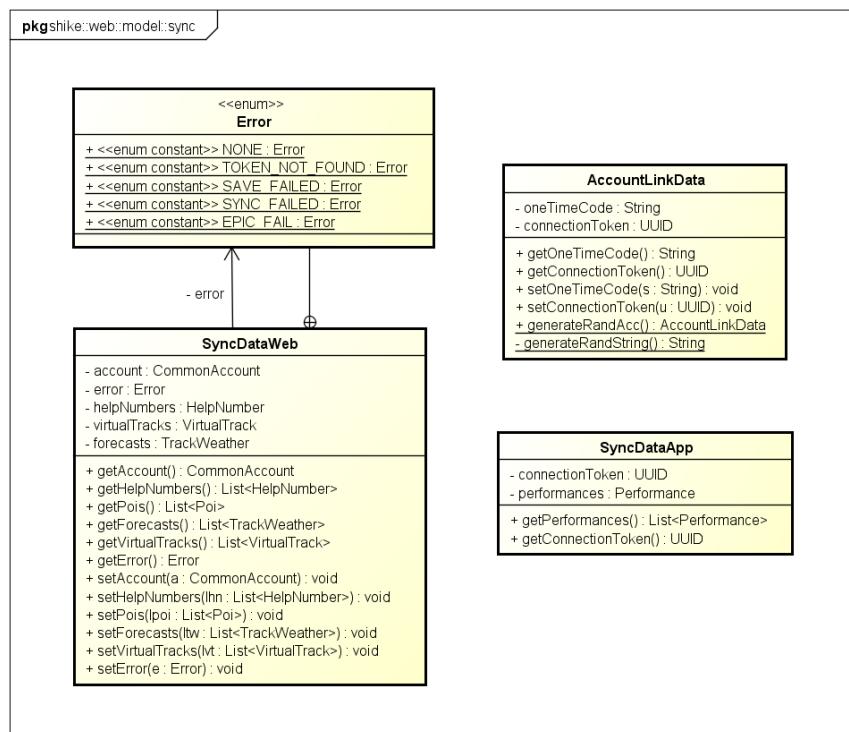


Figura 65: Diagramma di `shike::web::model::sync`

- **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`

4.58.2 Classi

4.58.2.1 shike::web::model::sync::AccountLinkData

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati che sono inviati dalla piattaforma web per il collegamento del dispositivo all' *account*.
- **Relazioni con altre classi**
 - ← SyncController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di sincronizzazione tra dispositivo e piattaforma web.
 - ← AccountService: classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.

4.58.2.2 shike::web::model::sync::SyncDataApp

- **Tipo:** concreta
- **Descrizione:** classe che modella i dati inviati dal device alla piattaforma web durante una sincronizzazione.
- **Relazioni con altre classi**
 - ← SyncController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di sincronizzazione tra dispositivo e piattaforma web.
 - ← AccountService: classe che permette di recuperare un riferimento ad un *account*, sia esso utente o amministratore, tramite la classe DAO corrispondente.

4.58.2.3 shike::web::model::sync::SyncDataWeb

- **Tipo:** concreta
- **Descrizione:** classe che modella le informazioni inviate dalla piattaforma web durante la sincronizzazione.
- **Relazioni con altre classi**
 - ← SyncController: classe che rappresenta tutti i comandi utilizzati per gestire le richieste HTTP riguardanti le operazioni di sincronizzazione tra dispositivo e piattaforma web.
 - → CommonAccount: classe che modella la struttura di un utente comune con aggiunta dei dati di accesso alla piattaforma web.
 - → Performance: classe che modella la struttura di una *performance* svolta da un utente.
 - → RecordedTrack: classe che modella la struttura di un percorso registrato dall'utente e carico sul web ma non condiviso.
 - → Poi: classe che modella la struttura di un punto di interesse (*POI*).
 - → TrackWeather: 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.
 - → Weather: classe che modella le informazioni sulle condizione meteo possibili.
 - → HelpNumber: classe che modella le informazioni relative ai numeri di soccorso salvati dall'utente.

4.58.2.4 shike::web::model::sync::SyncDataWeb.Error

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

5 Diagrammi di attività

In questa sezione vengono rappresentati i diagrammi di attività che descrivono le interazioni dell’utente con il prodotto Android e la piattaforma web.

5.1 Applicazione mobile

5.1.1 Attività principali dell’utente

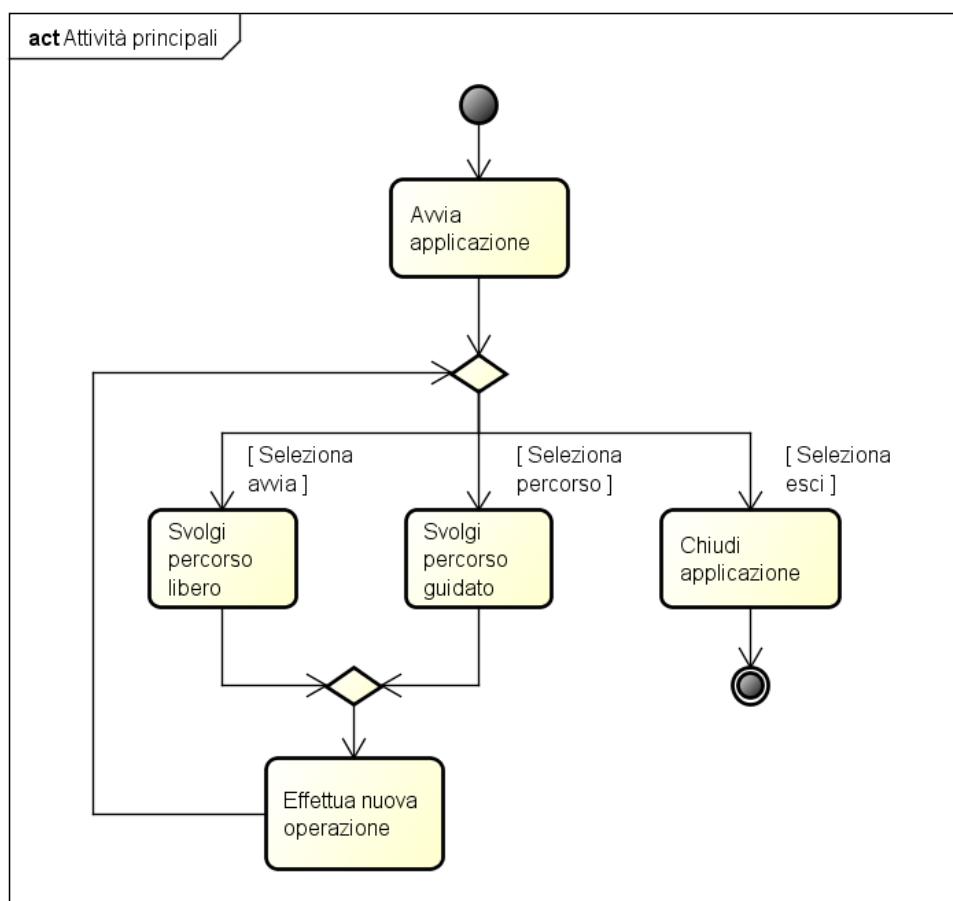


Figura 66: Diagramma delle attività principali dell’utente

Una volta che l’utente si è autenticato all’interno dell’applicazione, questi avrà la scelta di che operazione effettuare. La scelta può ricadere tra:

- svolgere un percorso libero
- svolgere un percorso guidato

Dopo aver individuato l’operazione e averla eseguita, può decidere se proseguire con un’altra azione oppure chiudere l’applicazione. L’utente potrebbe non eseguire alcuna operazione. Di conseguenza chiuderà solamente l’applicazione.

5.1.2 Sessione utente con percorso

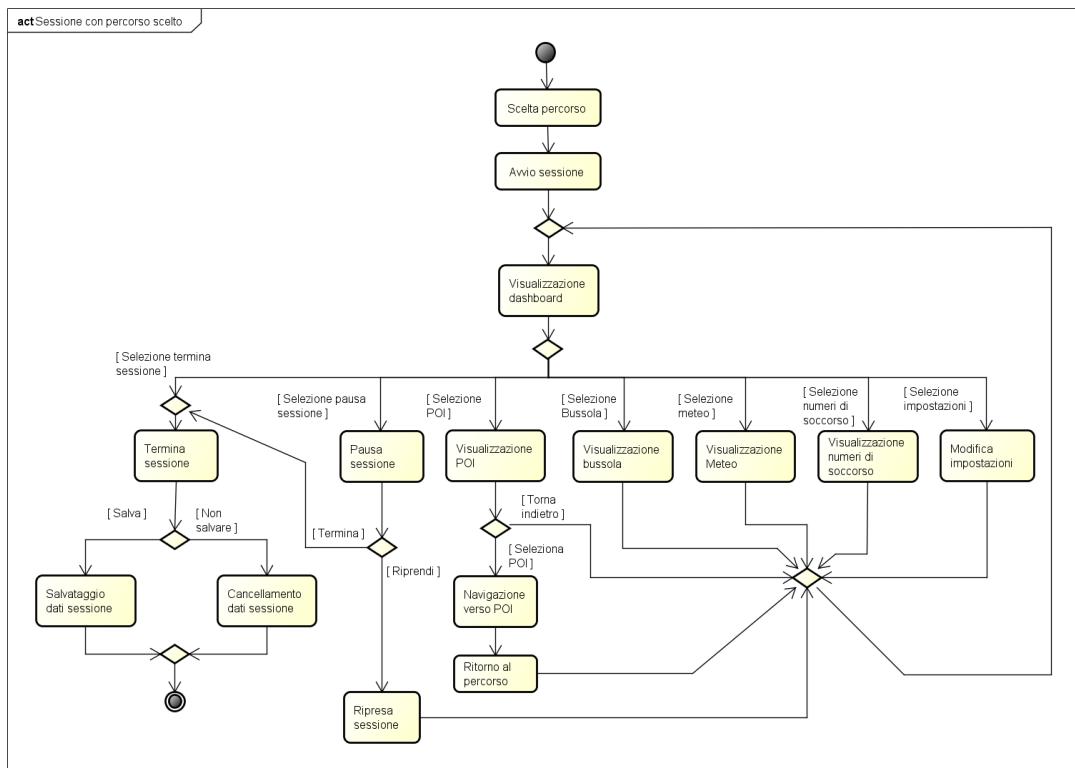


Figura 67: Diagramma per l'avvio di una nuova sessione con percorso

Una volta che si è scelto il percorso e avviato la sessione, il sistema passerà alla visualizzazione della *dashboard*. Da qui l'utente potrà decidere di effettuare diverse azioni come:

- terminare la sessione si chiede se l'utente vuole salvare o meno la sessione appena effettuata
- mettere in pausa la sessione, per poi terminarla definitivamente o riprenderla
- visualizzare un POI e decidere di effettuare la navigazione verso di questo
- visualizzare una bussola
- visualizzare informazioni meteo
- visualizzare i numeri di soccorso
- modificare le impostazioni

5.1.3 Sessione utente libera

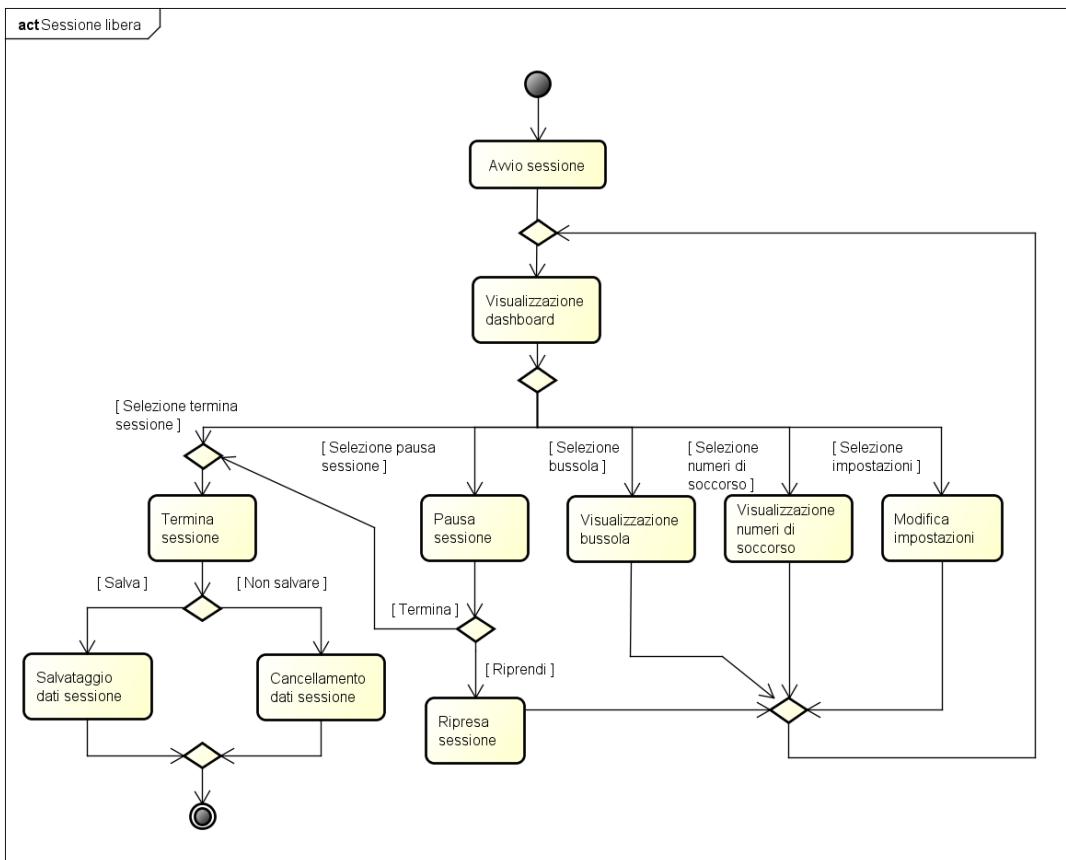


Figura 68: Diagramma per l'avvio di una nuova sessione libera

Una volta avviata la sessione il sistema passerà alla visualizzazione della *dashboard*. Da qui l'utente potrà decidere di effettuare diverse azioni come:

- terminare la sessione si chiede se l'utente vuole salvare o meno la sessione appena effettuata
- mettere in pausa la sessione, per poi terminarla definitivamente o riprenderla
- visualizzare una bussola
- visualizzare i numeri di soccorso
- modificare le impostazioni

5.1.4 Visualizza dashboard

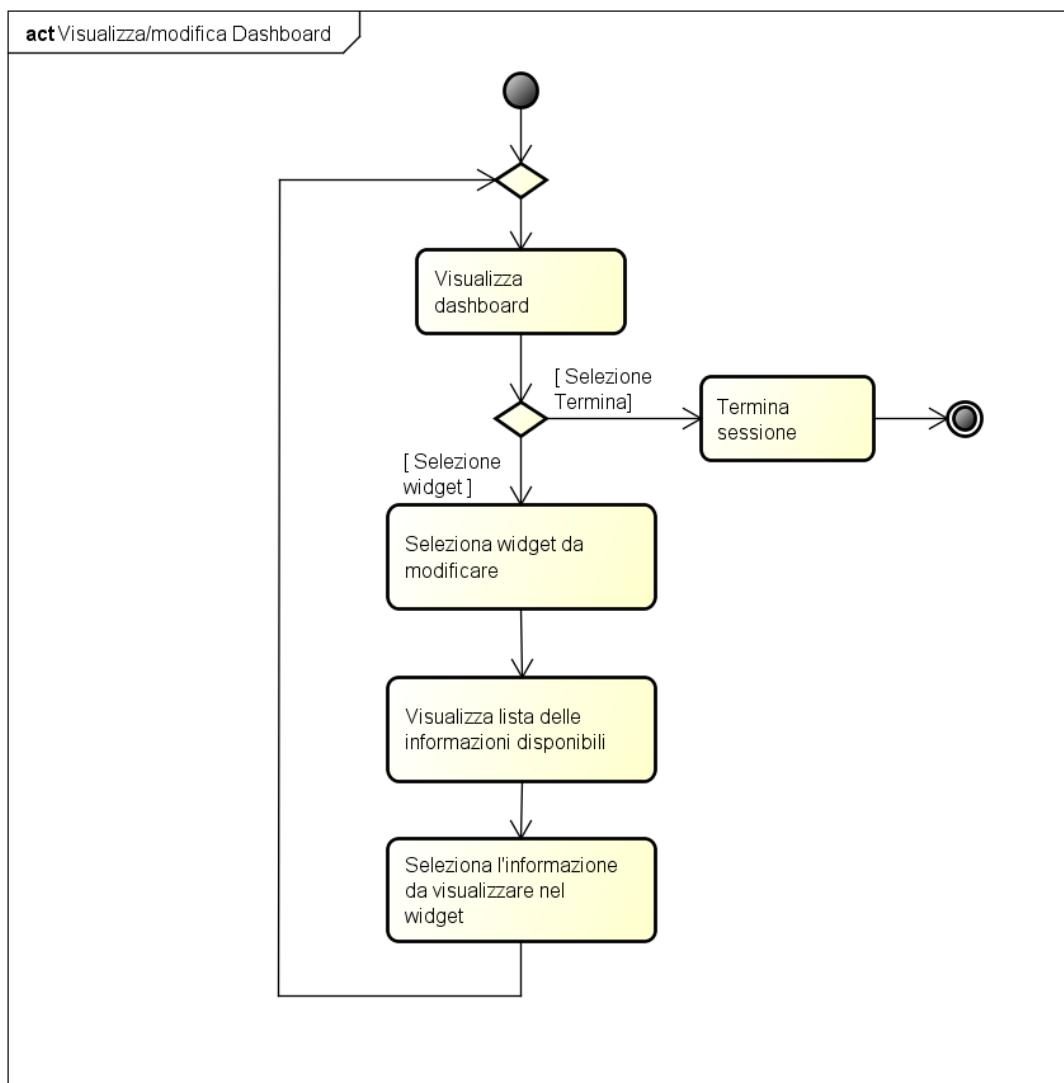


Figura 69: Diagramma per la gestione della dashboard

Una volta visualizzata la *dashboard* l'utente può decider se terminare la sessione in corso o selezionare il widget da modificare. Una volta fatto ciò è possibile decidere che informazioni visualizzare nel widget selezionato.

5.2 Applicazione web

5.2.1 Registrazione utente

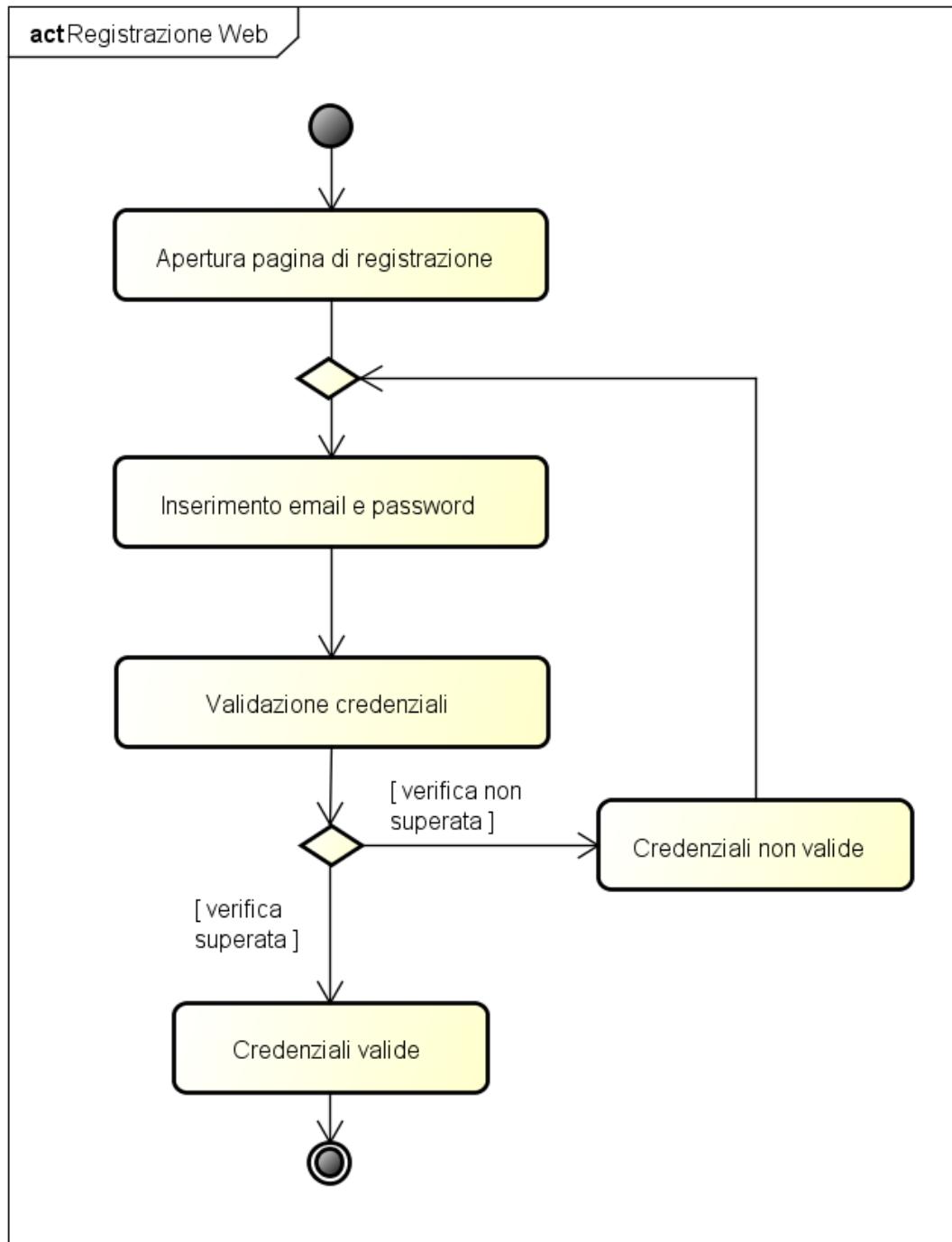


Figura 70: Diagramma per la registrazione di un nuovo utente

Dopo aver aperto la pagina di registrazione l'utente inserisce l'email e la password e conferma i campi, il sistema verificherà che siano corretti e, in caso di esito positivo conferma l'avvenuta registrazione. In caso contrario fa reinserire il campo non corretto.

5.2.2 Attività principali dell'utente

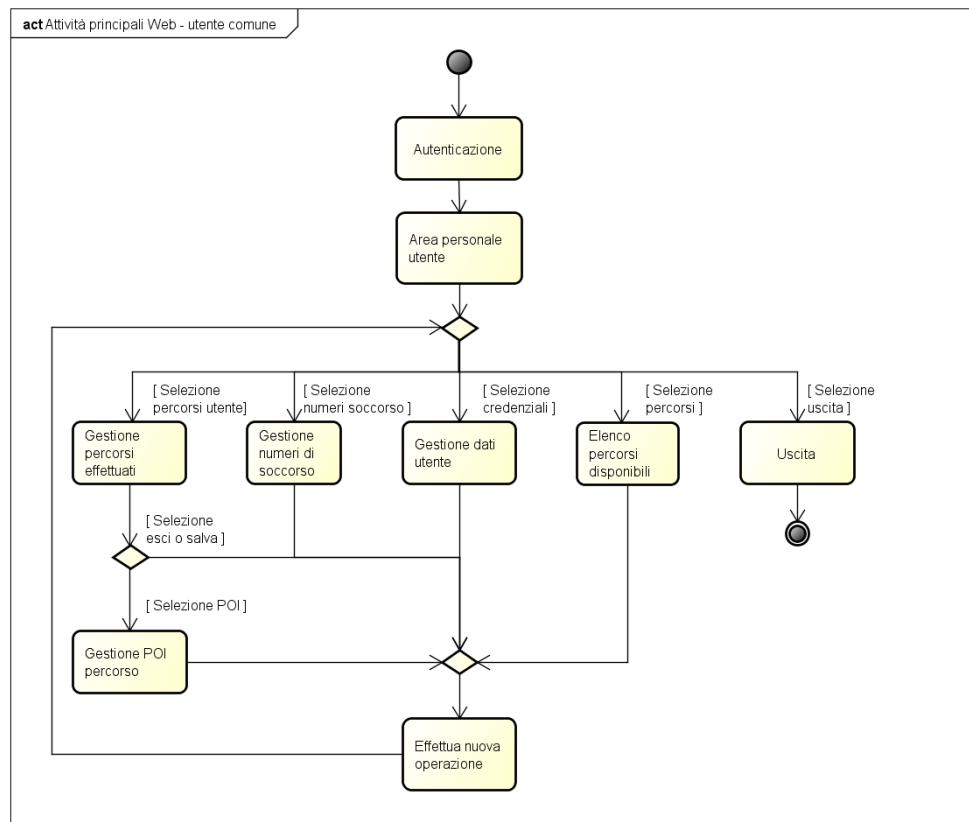


Figura 71: Diagramma delle attività principali dell'utente

Una volta che l'utente si è autenticato all'interno della piattaforma web, questi avrà la scelta di che operazione effettuare. La scelta può ricadere tra:

- gestire i percorsi effettuati (sezione 5.2.4)
- gestire i numeri di soccorso
- gestire i dati
- visualizzare l'elenco dei percorsi ricercando all'interno e scaricarli

Dopo aver individuato l'operazione e averla eseguita, può decidere se proseguire con un'altra azione oppure uscire. L'utente potrebbe non eseguire alcuna operazione, di conseguenza uscirà direttamente dalla piattaforma.

5.2.3 Attività principali dell'amministratore della piattaforma

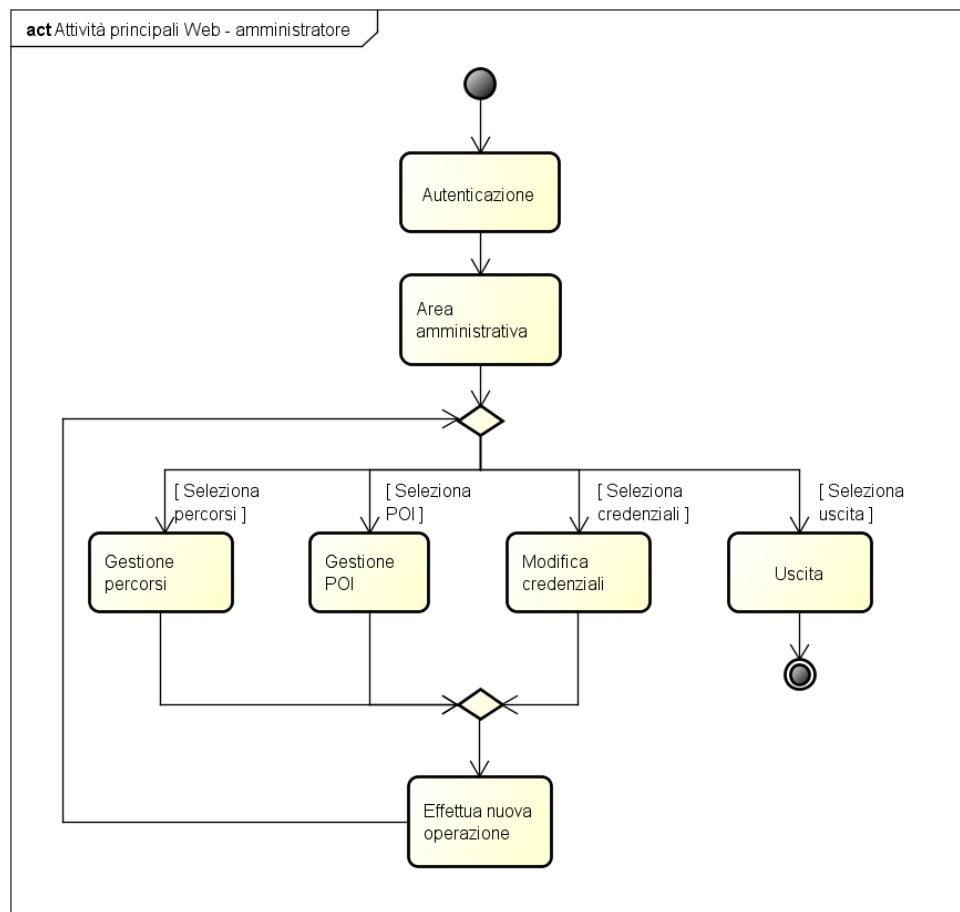


Figura 72: Diagramma delle attività principali dell'amministratore della piattaforma

Una volta che l'amministratore si è autenticato all'interno della piattaforma web, questi avrà la scelta di che operazione effettuare. La scelta può ricadere tra:

- gestire dei percorsi disponibili
- gestire i POI (sezione 5.2.5)
- modificare le credenziali

Dopo aver individuato l'operazione e averla eseguita, può decidere se proseguire con un'altra azione oppure uscire. L'amministratore potrebbe non eseguire alcuna operazione, di conseguenza uscirà direttamente dall'area amministrativa.

5.2.4 Gestione dei Percorsi

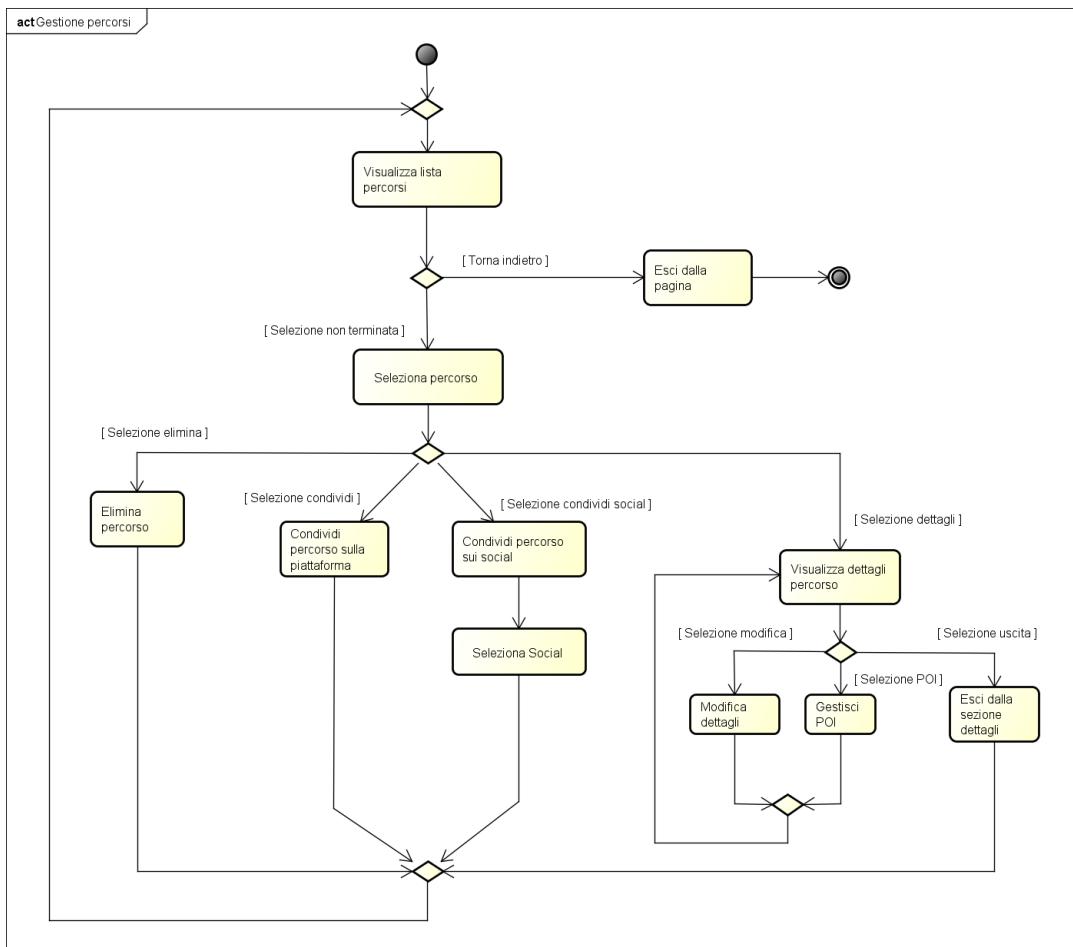


Figura 73: Diagramma per la gestione dei Percorsi

All'interno della sezione per la gestione dei percorsi l'utente visualizzerà la lista dei percorsi. Da qui può decidere se uscire dalla pagina o selezionare un percorso. Per ogni percorso è possibile effettuare diverse azioni come:

- eliminazione di un percorso
- condivisione di un percorso sulla piattaforma o sui social
- visualizzazione dei dettagli

Se si decide di visualizzare i dettagli di un percorso poi sarà possibile modificare i dettagli di questo o accedere alla pagina di **Gestione dei POI**.

5.2.5 Gestione dei POI

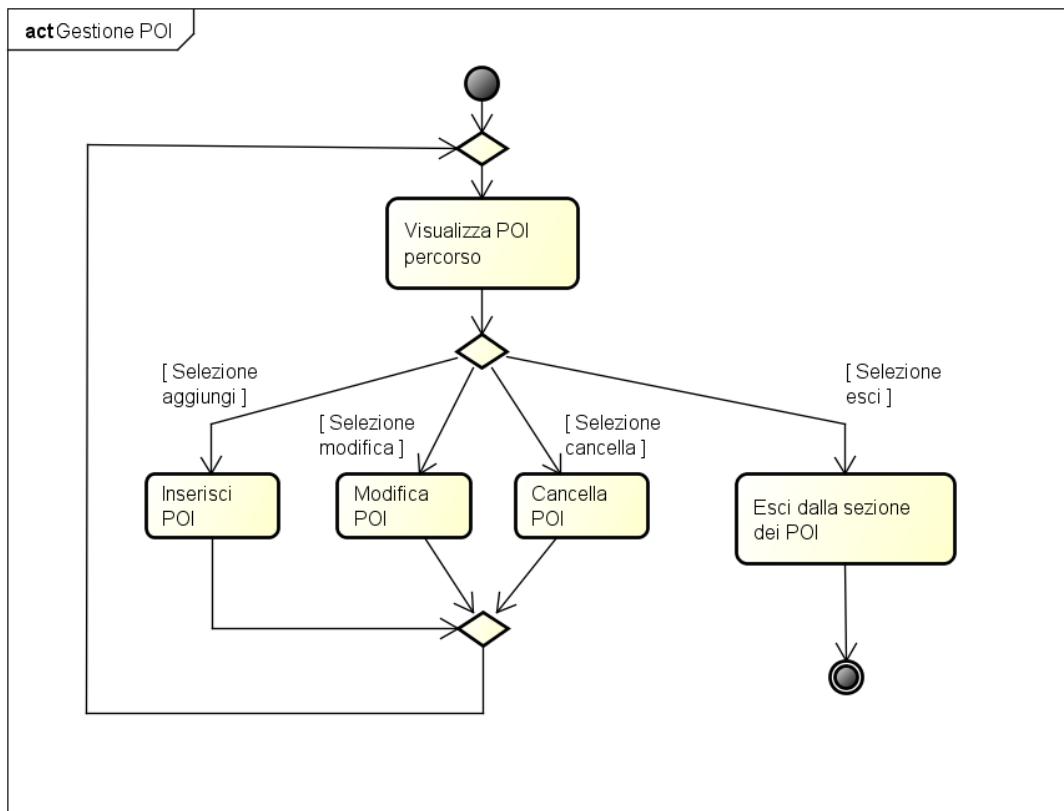


Figura 74: Diagramma per la gestione dei POI

All'interno della sezione per la visualizzazione dei POI di un percorso è possibile effettuare diverse azioni come:

- inserimento di un nuovo POI
- modifica di un POI
- cancellazione di un POI

Dopo aver effettuato un'azione l'utente può decidere se effettuarne un'altra o uscire dalla sezione POI.

6 Design Pattern

6.1 Design Pattern architetturali

6.1.1 MVC, Model-View-Controller

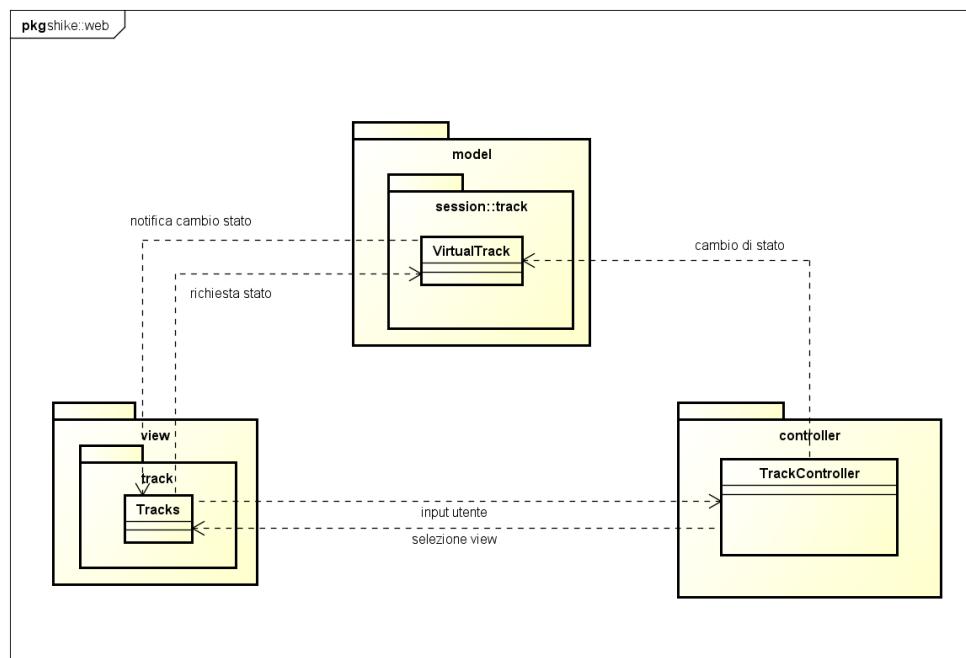


Figura 75: Diagramma del Design Pattern Model View Controller

- **Scopo dell'utilizzo:** è stato scelto il pattern MVC perché favorisce la separazione del modello di dati, dalla parte di logica e vista. Questo modello riduce l'accoppiamento tra le componenti aumentando la manutenibilità del codice e il suo riuso.
- **Contesto di utilizzo:** è utilizzato nell'architettura della parte Web.

6.1.2 MVP, Model-View-Presenter

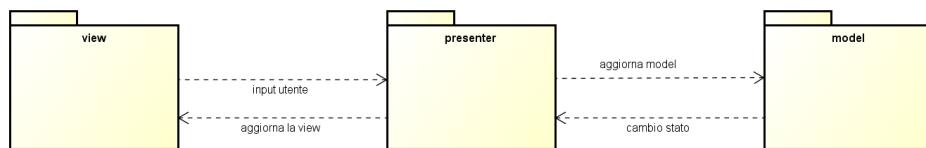


Figura 76: Diagramma del Design Pattern Model View Presenter

- **Scopo dell'utilizzo:** è stato scelto il pattern MVP perché favorisce la separazione del modello di dati, dalla parte di logica e vista; quest'ultima essendo infatti una semplice vista passiva si occupa solamente di invocare i metodi del presenter lasciando ad esso il

compito di gestire la creazione della View e il reperimento dei dati dal Model. Questo modello riduce l'accoppiamento tra le componenti aumentando la manutenibilità del codice e il suo riuso.

- **Contesto di utilizzo:** è utilizzato nell'architettura sviluppata per il *device Android*.

6.1.3 DAO, Data Access Object

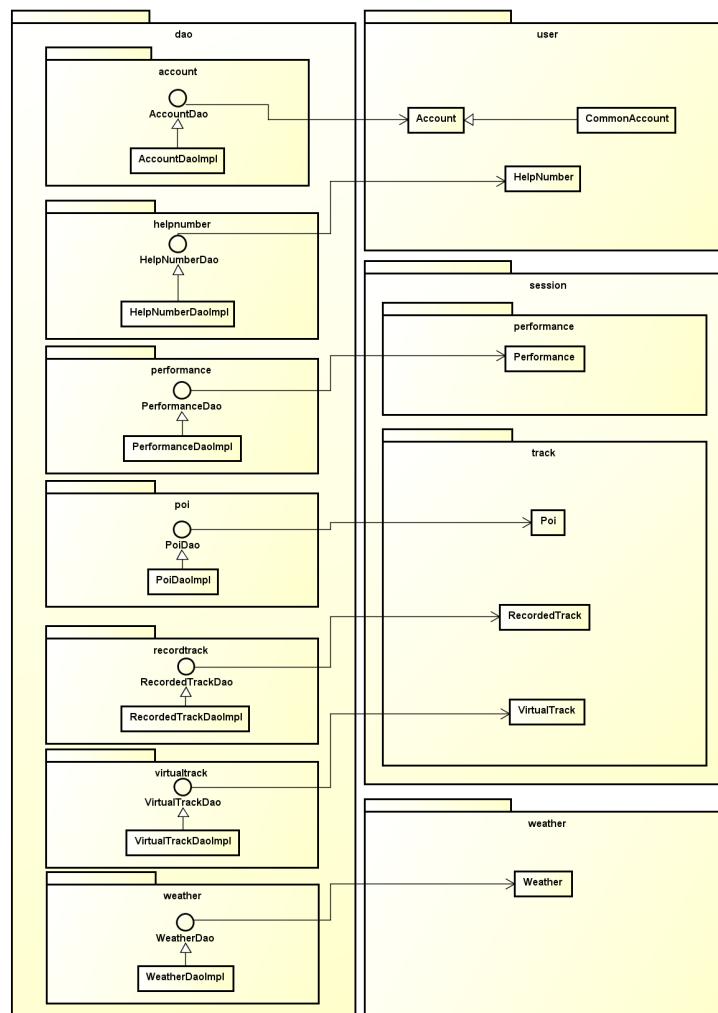


Figura 77: Diagramma del Design Pattern Data Access Object

- **Scopo dell'utilizzo:** l'intento del pattern *DAO (Data Access Object)* è di disaccoppiare la logica di business dalla logica di accesso al database. Questo si ottiene spostando la logica di accesso ai dati dai componenti di business ad una classe DAO rendendo le componenti della prima indipendenti dalla natura del dispositivo di persistenza. Questo approccio garantisce che un eventuale cambiamento del dispositivo di persistenza non comporti modifiche sui componenti di business. Inoltre legando la logica di accesso a un database ad una particolare istanza di un oggetto DAO, si favorisce la manutenibilità del sistema.

- **Contesto di utilizzo:** è utilizzato dall'architettura dell'applicazione web per implementare la serializzazione delle classi del model nella base dati.

6.2 Design Pattern creazionali

6.2.1 Singleton

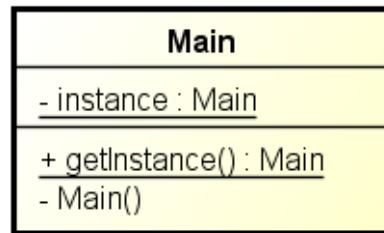


Figura 78: Diagramma del Design Pattern Singleton

- **Scopo dell'utilizzo:** il *pattern Singleton* assicura che una classe abbia una sola istanza e fornisce un punto d'accesso globale a tale istanza.
- **Contesto di utilizzo:** il pattern è utilizzato per avere un punto di accesso unico alle informazioni utenti contenute all'interno del *device Android*.

6.3 Design Pattern comportamentali

6.3.1 Strategy

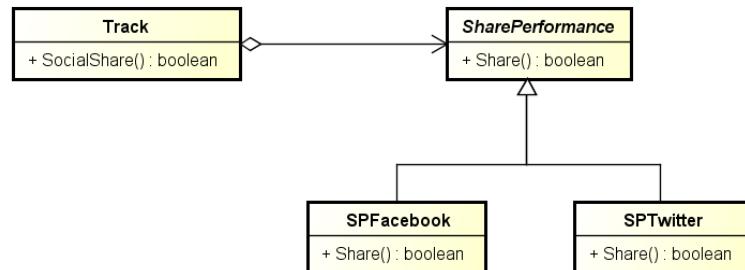


Figura 79: Diagramma del Design Pattern Strategy

- **Scopo dell'utilizzo:** il *pattern Strategy* definisce una famiglia di algoritmi, encapsulati e resi intercambiabili. *Strategy* permette agli algoritmi di variare indipendentemente dai client che ne fanno uso.
- **Contesto di utilizzo:** è utilizzato nell'applicazione web per implementare le funzionalità di condivisione con i *social*.

Ogni *social network* implementa un propria interfaccia API per la condivisione dei contenuti. Tramite il *pattern Strategy* si crea ha un interfaccia comune per differenti comportamenti.

7 Stime di fattibilità e di bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente a fornire una stima sulla fattibilità e di bisogno di risorse.

L'analisi dell'architettura ha messo in evidenza che le tecnologie scelte risultano sufficientemente adeguate per la realizzazione del prodotto. Riescono quindi a soddisfare le esigenze progettuali. La maggior parte delle tecnologie e degli strumenti sono nuove e poco conosciute per tutti i componenti del gruppo. Il team si impegnerà quindi ad approfondire personalmente e tramite il materiale fornito dall'Amministratore le relative capacità di utilizzo.

8 Tracciamento

8.1 Tabella delle associazioni Requisiti-Componenti

Requisito	Descrizione	Componenti associate
RTFO 1	Devono essere fornite indicazioni sul percorso	shike::app::view::session shike::app::view::home shike::app::model::dao::track shike::app::view::session
RAFO 1.2	L'applicazione deve indicare la direzione per rientrare nel percorso nel caso in cui l'utente sia uscito	
RAFO 1.3	L'applicazione deve fornire l'elenco con nome e distanza dall'utente dei percorsi disponibili	shike::app::view::home
RAFF 1.3.1	L'applicazione deve poter visualizzare un percorso	shike::app::view::session shike::app::view::home shike::app::model::dao::track shike::app::model::service
RAFF 1.3.1.1	L'applicazione deve poter visualizzare il tracciato	shike::app::model::session::track
RAFO 1.3.1.2	L'applicazione deve poter visualizzare la lunghezza del percorso	shike::app::view::home
RAFD 1.3.1.3	L'applicazione deve poter visualizzare la difficoltà del percorso	shike::app::view::home
RAFD 1.3.1.4	L'applicazione deve poter visualizzare la durata stimata del percorso	shike::app::view::home
RAFO 1.3.1.5	L'applicazione deve poter visualizzare il meteo del percorso se sono disponibili informazioni aggiornate	shike::app::model::weather shike::app::view::weather shike::app::model::dao::weather shike::app::model::service
RAFO 1.3.1.6	L'applicazione deve tenere traccia dei punti del percorso in cui l'utente è passato	shike::app::model::dao::performance
RAFO 1.4	L'applicazione deve fornire informazioni sui <u>POI</u> presenti nelle vicinanze del percorso	shike::app::model::session::track shike::app::view::poi shike::app::model::dao::poi shike::app::model::service
RAFO 1.4.1	L'applicazione deve indicare i rifugi	shike::app::model::session::track shike::app::view::poi shike::app::model::dao::poi
RAFO 1.4.2	L'applicazione deve indicare le fonti d'acqua	shike::app::model::session::track shike::app::view::poi shike::app::model::dao::poi
RAFD 1.4.3	L'applicazione deve indicare i bivacchi	shike::app::model::session::track shike::app::view::poi
RAFD 1.4.4	L'applicazione deve indicare le cime	shike::app::model::session::track shike::app::view::poi

Requisito	Descrizione	Componenti associate
RAFO 1.4.5	L'applicazione deve indicare i punti panoramici	shike::app::model::session::track shike::app::model::dao::poi
RWFO 1.7	La piattaforma deve fornire informazioni utili sul percorso	shike::web::controller
RWFO 1.7.2	La piattaforma deve dare indicazioni meteo locali	shike::app::model::weather
RWFO 1.7.4	La piattaforma deve indicare la difficoltà del percorso	shike::web::controller
RTFD 1.8	Deve essere fornito il numero del percorso	shike::app::model::session::track
RWFF 10	La piattaforma deve permettere la visualizzazione dei percorsi su mappe	shike::web::controller
RAFO 11	L'applicazione deve visualizzare numeri di soccorso utili all'utente in caso di emergenza	shike::app::model::user shike::app::model::dao::helpnumber shike::app::view::helpnumber shike::app::model::service
RAFO 12.4	L'applicazione deve permettere di avviare una nuova sessione	shike::app::view::session
RAFO 13	L'applicazione deve poter essere collegata ad un <i>account</i> del sito web	shike::app::model::dao::account
RWFO 13.1	Il sito deve generare un codice alfanumerico per permettere il collegamento dell' <i>account</i> utente al dispositivo.	shike::web::controller
RAFO 13.2	L'applicazione deve poter salvare i dati di collegamento all' <i>account</i> utente in modo persistente.	shike::app::model::dao::db shike::app::model::dao::account shike::app::model::service
RAFO 13.3	L'applicazione deve poter sincronizzare le informazioni del dispositivo con la parte web	shike::app::presenter shike::app::model::sync shike::app::helper
RWFO 13.5	Le informazioni della parte web deve poter essere sincronizzate sull'applicazione del dispositivo	shike::web::controller
RWFO 14	La piattaforma deve poter gestire gli utenti	shike::web::controller
RWFO 14.1	La piattaforma deve permettere la registrazione degli utenti	shike::web::controller
RWFO 14.1.1	La piattaforma deve richiedere l'indirizzo email dell'utente	shike::web::controller
RWFO 14.1.2	La piattaforma deve richiedere una password all'utente	shike::web::controller
RWFO 14.2	La piattaforma deve permettere l'autenticazione degli utenti registrati	shike::web::controller
RWFO 14.3	La piattaforma deve permettere agli utenti registrati di recuperare la propria password	shike::web::controller

Requisito	Descrizione	Componenti associate
RWFF 15	La piattaforma deve permettere all'utente di condividere i propri percorsi con i relativi tempi, per competere con altri utenti	shike::web::controller shike::web::helper::sharing
RWFF 15.1	Possibilità di condividere su Facebook	shike::web::controller shike::web::helper::sharing
RWFF 15.2	Possibilità di condividere su Twitter	shike::web::controller shike::web::helper::sharing
RAFO 16	L'applicazione deve salvare i dati di ogni escursione	shike::app::model::session::performance shike::app::model::dao::db shike::app::model::dao::performance shike::app::model::service
RAFO 16.1	L'applicazione deve poter salvare un nuovo percorso	shike::app::model::session::track shike::app::model::dao::db shike::app::model::dao::performance shike::app::model::service
RWFD 17	La piattaforma web deve permettere all'utente di gestire i propri numeri di soccorso	shike::web::model::dao::helpnumber shike::web::controller
RWFD 17.1	La piattaforma web deve permettere di visualizzare l'elenco dei numeri di soccorso memorizzati	shike::web::model::dao::helpnumber shike::web::controller
RWFD 17.2	La piattaforma web deve permettere all'utente di inserire un nuovo numero di soccorso	shike::web::model::dao::helpnumber shike::web::controller
RWFD 17.3	La piattaforma web deve permettere la modifica di un numero di soccorso presente	shike::web::model::dao::helpnumber shike::web::controller
RWFD 17.4	La piattaforma web deve permettere la cancellazione di un numero di soccorso presente	shike::web::controller
RWFO 18	L'utente deve poter visualizzare le statistiche sulle sue prestazioni	shike::web::model::session::performance shike::web::view::track
RWFO 19	L'utente deve poter gestire i dati del suo <i>account</i> .	shike::web::model::dao::account
RWFO 19.1	L'utente può modificare le sue credenziali	shike::web::model::dao::account
RWFF 19.1.1	L'utente può modificare il suo indirizzo email	shike::web::model::dao::account
RWFO 19.1.2	L'utente può modificare la sua password	shike::web::model::dao::account
RWFO 19.2	L'utente può gestire un dispositivo connesso all' <i>account</i> .	shike::web::controller
RWFO 19.2.1	L'utente può associare un unico dispositivo al suo <i>account</i> .	shike::web::controller
RWFO 19.2.2	L'utente può disassociare il dispositivo connesso al suo <i>account</i> .	shike::web::controller

Requisito	Descrizione	Componenti associate
RWFO 20	La piattaforma deve permettere la gestione dei percorsi	shike::web::controller
RWFO 20.1	Gli utenti della piattaforma devono poter aggiungere un nuovo percorso	shike::web::controller
RWFO 20.2	L'amministratore della piattaforma deve poter rimuovere un percorso	shike::web::controller
RWFO 20.3	L'amministratore della piattaforma deve poter modificare un percorso	shike::web::controller
RWFO 20.4	Gli utenti della piattaforma devono poter visualizzare l'elenco dei percorsi disponibili	shike::web::controller
RWFO 20.4.1	Gli utenti della piattaforma devono poter filtrare la visualizzazione dei percorsi disponibili	shike::web::controller
RWFF 20.4.1.4	Filtrare per autore (vedere solo i percorsi creati da un determinato utente)	shike::web::controller
RWFO 20.5	Gli utenti della piattaforma devono poter visualizzare i dettagli di un percorso	shike::web::controller
RWFO 20.6	Gli utenti della piattaforma devono poter ordinare la visualizzazione dei percorsi disponibili	shike::web::controller
RWFO 20.7	La piattaforma deve permettere la gestione dei <u>POI</u>	shike::web::controller shike::web::model::dao::poi
RWFO 20.7.1	Gli utenti della piattaforma devono poter aggiungere un nuovo <u>POI</u>	shike::web::controller shike::web::model::dao::poi
RWFF 20.7.2	Gli utenti della piattaforma devono poter modificare un <u>POI</u>	shike::web::controller shike::web::model::dao::poi
RWFF 20.7.2.1	L'amministratore deve poter modificare qualsiasi <u>POI</u>	shike::web::controller shike::web::model::dao::poi
RWFF 20.7.2.2	L'utente (non amministratore) deve poter modificare i <u>POI</u> che ha inserito	shike::web::controller shike::web::model::dao::poi
RWFO 20.7.3	Gli utenti della piattaforma devono poter cancellare un <u>POI</u>	shike::web::controller shike::web::model::dao::poi
RWFO 20.7.3.1	L'amministratore della piattaforma deve poter cancellare qualsiasi <u>POI</u>	shike::web::controller shike::web::model::dao::poi
RWFO 20.7.3.2	L'utente della piattaforma (non amministratore) deve poter cancellare i <u>POI</u> da lui creati	shike::web::controller shike::web::model::dao::poi
RWFO 20.7.4	L'amministratore della piattaforma deve poter visualizzare l'elenco dei <u>POI</u>	shike::web::controller shike::web::model::dao::poi

Requisito	Descrizione	Componenti associate
RWFD 20.7.4.1	L'amministratore della piattaforma deve poter filtrare la visualizzazione dei POI	shike::web::controller
RWFD 20.7.4.1.1	Filtrare per tipo POI	shike::web::controller
RWF0 20.7.5	Gli utenti della piattaforma devono poter visualizzare i dettagli di un POI	shike::web::controller shike::web::model::dao::poi
RAFO 21	L'applicazione deve permettere all'utente di scegliere quali dati visualizzare nella <i>dashboard</i>	shike::app::view::session
RWF0 22	La piattaforma deve permettere la visualizzazione dell'elenco dei percorsi effettuati dall'utente	shike::web::controller
RWF0 22.1	La piattaforma deve permettere all'utente di visualizzare i dettagli di un percorso effettuato	shike::web::controller
RWF0 22.2	La piattaforma deve permettere all'utente di poter filtrare l'elenco dei percorsi effettuati	shike::web::controller
RWF0 22.4	La piattaforma deve permettere la pubblicazione di un'escursione selezionata su un social network	shike::web::helper::sharing
RWF0 22.5	La piattaforma deve permettere di selezionare un percorso ed effettuare il download	shike::web::controller
RWF0 22.6	La piattaforma deve permettere la cancellazione di un percorso selezionato	shike::web::controller
RAFO 23	L'applicazione deve avere una <i>dashboard</i> , in cui l'utente può visualizzare in modo rapido le informazioni da lui scelte	shike::app::view::session
RAFO 27	L'applicazione deve avere una schermata di configurazione delle impostazioni	shike::app::view::config
RAFO 28	L'applicazione deve avere un menù dal quale sia possibile accedere a tutte le funzionalità	shike::app::view::home
RWF0 29	La piattaforma web deve avere una pagina per la gestione generale dei dati dell'utente	shike::web::controller
RTFO 3	Devono essere fornite indicazioni sull'utente	shike::app::model::user
RAFF 3.1.1	L'applicazione deve indicare i battiti istantanei	shike::app::model::session::performance
RAFF 3.1.2	L'applicazione deve indicare il valore massimo dei battiti cardiaci della sessione	shike::app::model::session::performance

Requisito	Descrizione	Componenti associate
RAFF 3.1.3	L'applicazione deve indicare il valore medio dei battiti cardiaci della sessione	shike::app::model::session::performance
RAFF 3.2	L'applicazione deve calcolare le zone cardiache dell'utente	shike::app::model::user
RWFD 3.3	Deve essere calcolato un profilo indicativo dell'utente che utilizza l'applicazione	shike::app::model::user
RWFD 3.3.1	Deve essere indicato se l'utente è un escursionista esperto	shike::app::model::user
RWFD 3.3.2	Deve essere indicato se l'utente è un escursionista occasionale	shike::app::model::user
RWFO 30	La piattaforma web deve avere una pagina principale per l'utente	shike::web::controller
RAFO 5	L'applicazione deve fornire una bussola	shike::app::view::session
RAFO 6	L'applicazione deve fornire indicazioni meteo	shike::app::model::weather shike::app::view::weather shike::app::model::dao::weather shike::app::model::service
RAFO 6.1	L'applicazione deve permettere di caricare le informazioni meteo prima di iniziare l'escursione	shike::app::model::weather shike::app::model::dao::weather shike::app::model::service
RAFO 6.5	L'applicazione deve fornire indicazioni sul vento	shike::app::model::weather
RAFO 6.5.1	L'applicazione deve indicare la velocità del vento	shike::app::model::weather
RAFO 6.5.2	L'applicazione deve indicare la direzione del vento	shike::app::model::weather
RAFD 6.7	L'applicazione deve indicare con un <i>alert</i> se ci possono essere condizioni meteorologiche pericolose	shike::app::model::weather
RAFD 6.7.2	L'applicazione deve avvertire se il meteo prevede la presenza di neve nel percorso	shike::app::model::weather
RAFD 6.7.3	L'applicazione deve avvertire se il meteo prevede un temporale	shike::app::model::weather

Tabella 2: Tracciamento Requisiti-Componenti

8.2 Tabella delle associazioni Componenti-Requisiti

Componente	Requisiti associati
shike::app::presenter	RAFO 13.3
shike::app::model::weather	RAFO 1.3.1.5 RWFO 1.7.2 RAFO 6 RAFO 6.1 RAFO 6.5 RAFO 6.5.1 RAFO 6.5.2 RAFD 6.7 RAFD 6.7.2 RAFD 6.7.3
shike::app::model::session::track	RAFF 1.3.1.1 RAFO 1.4 RAFO 1.4.1 RAFO 1.4.2 RAFD 1.4.3 RAFD 1.4.4 RAFO 1.4.5 RTFD 1.8 RAFO 16.1
shike::app::model::session::performance	RAFO 16 RAFF 3.1.1 RAFF 3.1.2 RAFF 3.1.3
shike::app::model::user	RAFO 11 RTFO 3 RAFF 3.2 RWFD 3.3 RWFD 3.3.1 RWFD 3.3.2

Componente	Requisiti associati
shike::web::controller	RWFO 1.7 RWFO 1.7.4 RWFF 10 RWFO 13.1 RWFO 13.5 RWFO 14 RWFO 14.1 RWFO 14.1.1 RWFO 14.1.2 RWFO 14.2 RWFO 14.3 RWFF 15 RWFF 15.1 RWFF 15.2 RWFD 17 RWFD 17.1 RWFD 17.2 RWFD 17.3 RWFD 17.4 RWFO 19.2 RWFO 19.2.1 RWFO 19.2.2 RWFO 20 RWFO 20.1 RWFO 20.2 RWFO 20.3 RWFO 20.4 RWFO 20.4.1 RWFF 20.4.1.4 RWFO 20.5 RWFO 20.6 RWFO 20.7 RWFO 20.7.1 RWFF 20.7.2 RWFF 20.7.2.1 RWFF 20.7.2.2 RWFO 20.7.3 RWFO 20.7.3.1 RWFO 20.7.3.2 RWFO 20.7.4 RWFD 20.7.4.1 RWFD 20.7.4.1.1 RWFO 20.7.5 RWFO 22 RWFO 22.1 RWFO 22.2 RWFO 22.5 RWFO 22.6 RWFO 29 RWFO 30

Componente	Requisiti associati
shike::web::model::session::performance	RWFO 18
shike::web::helper::sharing	RWFF 15 RWFF 15.1 RWFF 15.2 RWFO 22.4
shike::web::model::dao::account	RWFO 19 RWFO 19.1 RWFF 19.1.1 RWFO 19.1.2
shike::web::model::dao::helpnumber	RWFD 17 RWFD 17.1 RWFD 17.2 RWFD 17.3
shike::web::model::dao::poi	RWFO 20.7 RWFO 20.7.1 RWFF 20.7.2 RWFF 20.7.2.1 RWFF 20.7.2.2 RWFO 20.7.3 RWFO 20.7.3.1 RWFO 20.7.3.2 RWFO 20.7.4 RWFO 20.7.5
shike::app::model::dao::account	RAFO 13 RAFO 13.2
shike::app::model::dao::helpnumber	RAFO 11
shike::app::model::dao::performance	RAFO 1.3.1.6 RAFO 16 RAFO 16.1
shike::app::model::dao::poi	RAFO 1.4 RAFO 1.4.1 RAFO 1.4.2 RAFO 1.4.5
shike::app::model::dao::track	RTFO 1 RAFF 1.3.1
shike::app::model::dao::weather	RAFO 1.3.1.5 RAFO 6 RAFO 6.1
shike::web::view::track	RWFO 18
shike::app::helper	RAFO 13.3

Componente	Requisiti associati
shike::app::model::service	RAFF 1.3.1 RAFO 1.3.1.5 RAFO 1.4 RAFO 11 RAFO 13.2 RAFO 16 RAFO 16.1 RAFO 6 RAFO 6.1
shike::app::model::sync	RAFO 13.3
shike::app::model::dao::db	RAFO 13.2 RAFO 16 RAFO 16.1
shike::app::view::config	RAFO 27
shike::app::view::helpnumber	RAFO 11
shike::app::view::home	RTFO 1 RAFO 1.3 RAFF 1.3.1 RAFO 1.3.1.2 RAFD 1.3.1.3 RAFD 1.3.1.4 RAFO 28
shike::app::view::poi	RAFO 1.4 RAFO 1.4.1 RAFO 1.4.2 RAFD 1.4.3 RAFD 1.4.4
shike::app::view::session	RTFO 1 RAFO 1.2 RAFF 1.3.1 RAFO 12.4 RAFO 21 RAFO 23 RAFO 5
shike::app::view::weather	RAFO 1.3.1.5 RAFO 6

Tabella 3: Tracciamento Componenti-Requisiti

A Descrizione generale Design Pattern

A.1 Design Pattern architetturali

A.1.1 MVC, Model-View-Controller

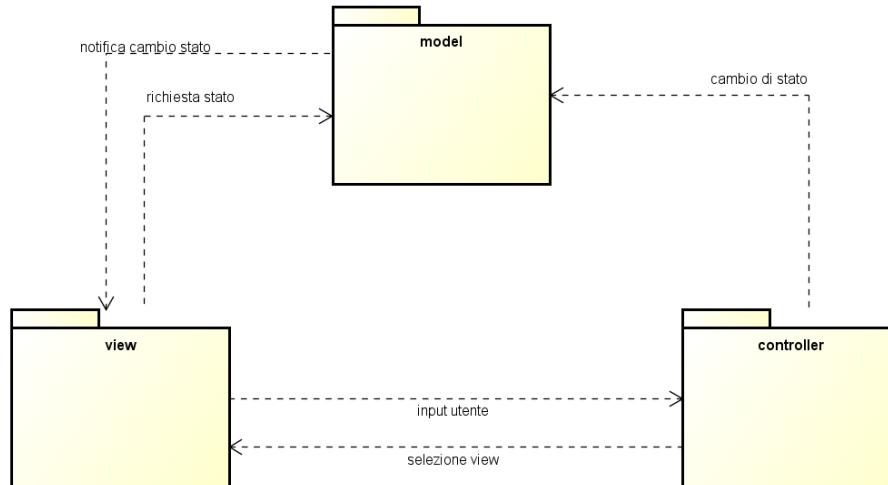


Figura 80: Diagramma generico del Design Pattern MVC

L'architettura utilizzata nel progetto riprende il design pattern MVC:

- **Descrizione:** il design pattern MVC permette una completa disgiunzione tra le funzionalità di vista e di modello dei dati. Si individuano tre componenti:
 1. **Model** rappresenta la logica di memorizzazione e recupero di dati utilizzati nel sistema;
 2. **View** visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti esterni;
 3. **Controller** riceve i comandi dell'utente (in genere attraverso la View) e li attua modificando lo stato degli altri due componenti.
- **Motivazione:** tale design pattern permette la separazione tra la modellazione del dominio, la presentazione, e le azioni basate sugli input degli utenti, all'interno di tre entità separate. Questo schema implica anche la tradizionale separazione fra la logica applicativa, a carico del Controller e del Model, e l'interfaccia utente, a carico della View. Per il progetto si è deciso di implementare la strategia “push-model” in modo da avere una vista più disaccoppiata possibile dal modello.
- **Contesto applicativo:** dopo l'incontro con la ditta si è scelto di implementare il design pattern MVC per entrambi le parti del progetto. Per l'applicazione Android si utilizzerà il framework nativo senza nessun SDK specializzato in quanto quello del proponente non risulta disponibile. L'utilizzo del MVC in questo caso segue le indicazioni, fornite dalla ditta stessa, di usare Fragment come viste e di inizializzare i controller all'interno di un'unica Activity. A seguito dell'incontro è emersa la totale inesistenza della struttura “Cloud”, abbiamo comunque deciso di attenerci il più possibile ai requisiti iniziali e quindi per lo sviluppo della parte Web sarà utilizzato il Framework Spring che implementa il modello MVC nativamente.

A.1.2 DAO, Data Access Object

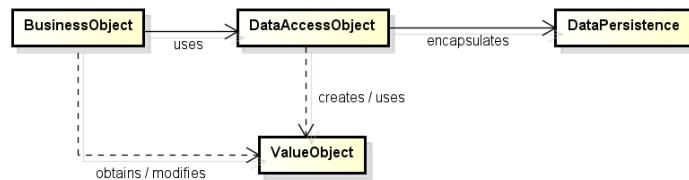


Figura 81: Diagramma generico del Design Pattern Data Access Object

- **Descrizione:** l'idea del pattern DAO si basa sulla possibilità di concentrare il codice per l'accesso al sistema di persistenza in una classe che si occupa di gestire la logica di accesso a questo.
- **Motivazione:** tale design pattern permette di disaccoppiare la logica di business dalla logica di accesso ai dati. Questo si ottiene spostando la logica di accesso ai dati dalle componenti di business ad una classe DAO, rendendo le componenti della prima indipendenti dalla natura del dispositivo di persistenza. Questo approccio garantisce che un eventuale cambiamento del dispositivo di persistenza non comporti modifiche sulle componenti di business.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. si desidera stratificare e isolare l'accesso ad una tabella dalla parte della logica di business;
 2. si desidera nascondere i dettagli implementativi della base di dati, permettendo di cambiare tipo di database senza dover modificare troppo codice.

A.2 Design Pattern creazionali

A.2.1 Singleton

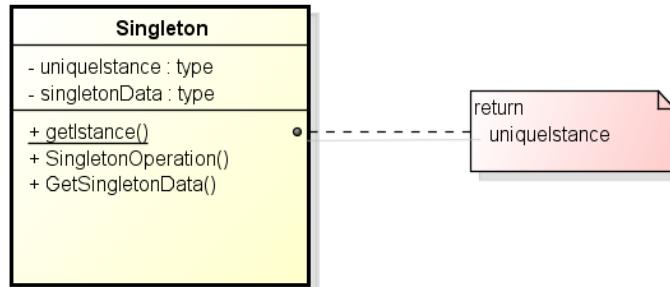


Figura 82: Diagramma generico del Design Pattern Singleton

- **Descrizione:** il design pattern Singleton assicura che una classe (Singleton) abbia una unica istanza e che questa sia globalmente accessibile in un punto ben noto.

- **Motivazione:** garantire che di una determinata classe venga creata una e una sola istanza, fornendo un punto di accesso globale a tale istanza. Si garantisce così un risparmio di risorse fisiche poiché la creazione di istanze dell'oggetto in questione è rimandata fino al momento dell'effettivo utilizzo dell'oggetto in questione.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. si desidera la garanzia che nel sistema vi sia una sola istanza di oggetti di un determinato tipo, e che tali oggetti debbano essere accessibili da un punto di accesso ben preciso;
 2. si desidera che non venga delegato ad altri il controllo di unicità di un oggetto;
 3. si desidera che più oggetti condividano un unico pool di dati.

A.3 Design Pattern comportamentali

A.3.1 Strategy

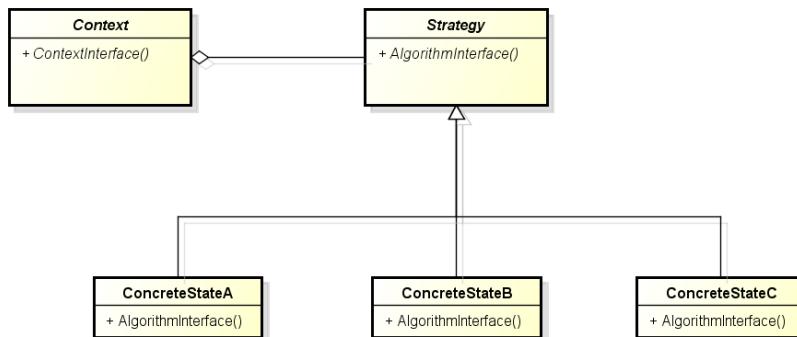


Figura 83: Diagramma generico del Design Pattern Strategy

- **Descrizione:** il design pattern Strategy incapsula una famiglia di algoritmi in una famiglia (Strategy) di classi (`ConcreteStrategyA`, `ConcreteStrategyB`, etc.) rendendoli intercambiabili in base alla situazione (Context). Prevede che gli algoritmi siano intercambiabili tra loro (in base ad una qualche condizione) in modo trasparente al client che ne fa uso. Si applica in situazioni dove esistono diverse strategie per fare una cosa ma l'utente (o l'oggetto) ne sceglie una solamente.
- **Motivazione:** definire una famiglia di algoritmi, incapsularli e renderli intercambiabili in maniera trasparente rispetto all'uso da parte del client.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. classi collegate differiscono solo per il loro comportamento;
 2. gli algoritmi utilizzano dati non a conoscenza del client;
 3. una classe definisce differenti comportamenti, espressi come *statement* condizionali multipli nelle operazioni (il problema si risolve in maniera simile al caso del pattern State).