

MashUp

Software Engineering Group

info@mashup-unipd.it

Informazioni Documento

Nome documento	<i>Definizione di Prodotto</i>
Versione	<i>v4.0.0</i>
Data redazione	2015-04-01
Redattori	Ceccon Lorenzo Santacatterina Luca Tesser Paolo
Verificatori	Faccin Nicola Roetta Marco
Approvazione	Carnovalini Filippo
Lista distribuzione	<i>MashUp</i> <i>Prof. Tullio Vardanega</i> <i>Prof. Riccardo Cardin</i> <i>Dott. David Santucci - Zing Srl</i>
Uso	Esterno

Sommario

Questo documento vuole definire l'architettura in dettaglio del prodotto che verrà sviluppato dal team *MashUp*

Diario Revisioni

Versione	Data	Modifica	Autore & Ruolo
v4.0.0	2015-06-12	<i>Approvazione documento</i>	Faccin Nicola <i>Responsabile di Progetto</i>
v3.1.0	2015-06-11	<i>Verifica modifiche lato server e client</i>	Santacatterina Luca <i>Verificatore</i>
v3.0.6	2015-06-10	<i>Aggiornati riferimenti informativi e normativi</i>	Cusinato Giacomo <i>Progettista</i>
v3.0.5	2015-06-10	<i>Aggiornata descrizione e diagrammi classi package server::processor</i>	Cusinato Giacomo <i>Progettista</i>
v3.0.4	2015-06-09	<i>Aggiunta descrizione ai metodi statici delle classi del package server::miner</i>	Cusinato Giacomo <i>Progettista</i>
v3.0.3	2015-06-09	<i>Sistemazione correzioni relativi ai meccanismi di routing del client</i>	Carnovalini Filippo <i>Progettista</i>
v3.0.2	2015-06-09	<i>Sistemazione correzioni relativi alla generazione del token e dell'invio della password</i>	Carnovalini Filippo <i>Progettista</i>
v3.0.1	2015-06-08	<i>Sistemazione history ordinata sulla base della versione</i>	Cusinato Giacomo <i>Progettista</i>
v3.0.0	2015-05-28	<i>Approvazione documento</i>	Roetta Marco <i>Responsabile di Progetto</i>
v2.1.0	2015-05-28	<i>Verifica modifiche lato server e client</i>	Cusinato Giacomo <i>Responsabile di Progetto</i>
v2.0.3	2015-05-26	<i>Separazione ViewTypeModel</i>	Carnovalini Filippo <i>Responsabile di Progetto</i>
v2.0.2	2015-05-25	<i>Specificazione delle classi di generazione dei grafici con l'uso di promise</i>	Carnovalini Filippo <i>Responsabile di Progetto</i>
v2.0.1	2015-05-22	<i>Aggiunta descrizione classe in processor::commands::social</i>	Santacatterina Luca <i>Progettista</i>
v2.0.0	2015-05-22	<i>Approvazione documento</i>	Carnovalini Filippo <i>Responsabile di Progetto</i>
v1.4.1	2015-05-18	<i>Verificati capitoli classe client::model::data 3.2.3</i>	Faccin Nicola <i>Verificatore</i>
v1.3.1	2015-04-16	<i>Integrazione diagramma, metodi e attributi sulla classe client::model::data 3.2.3</i>	Tesser Paolo <i>Progettista</i>
v1.3.0	2015-05-14	<i>Verificata integrale capitoli server</i>	Faccin Nicola <i>Verificatore</i>
v1.2.0	2015-05-07	<i>Verificati capitoli client server::miner 3.3.16 e server::endpoints 3.3.20</i>	Roetta Marco <i>Verificatore</i>
v1.1.1	2015-05-06	<i>Integrazione diagramma, metodi ed attributi server::miner 3.3.16 e server::endpoints 3.3.20</i>	Santacatterina Luca <i>Progettista</i>
v1.1.0	2015-05-05	<i>Verificata integrale capitoli client</i>	Roetta Marco <i>Verificatore</i>

v1.0.11	2015-05-03	<i>Modificato il diagramma di sequenza del client con login utente e aggiunto al capitolo 4.1 descrizione di autenticazione</i>	Tesser Paolo <i>Progettista</i>
v1.0.10	2015-05-01	<i>Inserimento diagramma di sequenza server nel capitolo 4.2 con relative descrizione fasi di autenticazione</i>	Ceccon Lorenzo <i>Progettista</i>
v1.0.9	2015-05-01	<i>Inserimento diagramma di sequenza client nel capitolo 4.1</i>	Tesser Paolo <i>Progettista</i>
v1.0.8	2015-04-27	<i>Aggiornamento tracciamento</i>	Santacatterina Luca <i>Progettista</i>
v1.0.7	2015-04-26	<i>Modifica riferimenti informativi. Aggiunta Google Platform</i>	Santacatterina Luca <i>Progettista</i>
v1.0.6	2015-04-25	<i>Sistemazione errori ortografici e commenti codice sorgente documenti</i>	Tesser Paolo <i>Progettista</i>
v1.0.5	2015-04-22	<i>Stesura descrizione metodi ed attributi della classe client::view 3.2.5</i>	Santacatterina Luca <i>Progettista</i>
v1.0.4	2015-04-20	<i>Stesura descrizione metodi ed attributi della classe client::model::data 3.2.3 e client::model::services 3.2.4</i>	Tesser Paolo <i>Progettista</i>
v1.0.3	2015-04-19	<i>Stesura descrizione attributi client::controller 3.2.9</i>	Tesser Paolo <i>Progettista</i>
v1.0.2	2015-04-18	<i>Stesura descrizione metodi ed attributi della classe server::processor 3.3.10 e server::endpoints 3.3.20</i>	Ceccon Lorenzo <i>Progettista</i>
v1.0.1	2015-04-16	<i>Stesura descrizione metodi ed attributi della classe server::db 3.3.2 e server::miner 3.3.16</i>	Santacatterina Luca <i>Progettista</i>
v1.0.0	2015-04-13	<i>Approvazione documento</i>	Cusinato Giacomo <i>Responsabile di Progetto</i>
v0.3.0	2015-04-12	<i>Verifica sezione 3.2.9 client::controller</i>	Tesser Paolo <i>Verificatore</i>
v0.2.2	2015-04-12	<i>Inserita sezione tracciamento e riportato dalla base di dati la tabella</i>	Santacatterina Luca <i>Progettista</i>
v0.2.1	2015-04-12	<i>Revisione e sistemazione diagrammi sezione 3.2.9 client::controller</i>	Ceccon Lorenzo <i>Progettista</i>
v0.2.0	2015-04-12	<i>Verifica integrale parte back-end</i>	Carnovalini Filippo <i>Verificatore</i>
v0.1.0	2015-04-12	<i>Verifica integrale parte front-end</i>	Tesser Paolo <i>Verificatore</i>
v0.0.9	2015-04-10	<i>Aggiornati contenuti classi sez. Client e Server - Attributi e Metodi</i>	Roetta Marco <i>Progettista</i>
v0.0.8	2015-04-09	<i>Inserimento descrizione metodi e attributi sezione server::miner 3.3.16</i>	Santacatterina Luca <i>Progettista</i>
v0.0.7	2015-04-08	<i>Inserimento descrizione metodi e attributi sezione server::db 3.3.2</i>	Faccin Nicola <i>Progettista</i>

v0.0.6	2015-04-06	<i>Stesura sezione client::controller 3.2.9 andando ad aggiungere i servizi di AngularJS utilizzati, i metodi e gli attributi dei controller relativi ai requisiti obbligatori</i>	Ceccon Lorenzo <i>Progettista</i>
v0.0.5	2015-04-05	<i>Aggiornati contenuti classi sez. Client - Attributi e Metodi</i>	Roetta Marco <i>Progettista</i>
v0.0.4	2015-04-04	<i>Aggiornati contenuti classi sez. Server - Attributi e Metodi</i>	Faccin Nicola <i>Progettista</i>
v0.0.3	2015-04-02	<i>Stesura sezione 3.2.5 client::view andando ad aggiungere le direttive di AngularJS utilizzate per i template HTML relativi ai requisiti obbligatori</i>	Ceccon Lorenzo <i>Progettista</i>
v0.0.2	2015-04-02	<i>Importato parte del contenuto dal documento Specifica Tecnica v3.0.0</i>	Santacatterina Luca <i>Progettista</i>
v0.0.1	2015-04-01	<i>Creata scheletro del documento</i>	Santacatterina Luca <i>Progettista</i>

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	Standard di progetto	3
2.1	Progettazione architetturale	3
2.2	Documentazione del codice	3
2.3	Denominazione di entità e relazioni	3
2.4	Codifica	3
2.5	Strumenti di lavoro	3
3	Specifiche Componenti	4
3.1	Note per la lettura	4
3.2	Client (Front-end)	5
3.2.1	client	6
3.2.2	client::model	7
3.2.3	client::model::data	8
3.2.3.1	Classi	8
3.2.4	client::model::services	14
3.2.4.1	Classi	14
3.2.5	client::view	27
3.2.6	client::view::public	29
3.2.6.1	Classi	29
3.2.7	client::view::user	31
3.2.7.1	Classi	31
3.2.8	client::view::admin	39
3.2.8.1	Classi	39
3.2.9	client::controller	45
3.2.10	client::controller::public	45
3.2.10.1	Classi	45
3.2.11	client::controller::user	50
3.2.11.1	Classi	51
3.2.12	client::controller::admin	68
3.2.12.1	Classi	69
3.3	Server (Back-end)	78
3.3.1	server	78
3.3.2	server::db	80
3.3.3	server::db::raw_data	80
3.3.3.1	Classi	81
3.3.4	server::db::raw_data::fb	83
3.3.4.1	Classi	83
3.3.5	server::db::raw_data::tw	86
3.3.5.1	Classi	88
3.3.6	server::db::raw_data::ig	91
3.3.6.1	Classi	92

3.3.7	server::db::app_data	95
3.3.8	server::db::app_data::user	95
3.3.8.1	Classi	96
3.3.9	server::db::app_data::recipe	97
3.3.9.1	Classi	97
3.3.10	server::processor	102
3.3.10.1	Classi	102
3.3.11	server::processor::commands	104
3.3.11.1	Classi	105
3.3.12	server::processor::commands::user	105
3.3.12.1	Classi	106
3.3.13	server::processor::commands::recipe	109
3.3.13.1	Classi	109
3.3.14	server::processor::commands::requests	112
3.3.14.1	Classi	112
3.3.15	server::processor::commands::social	115
3.3.15.1	Classi	115
3.3.16	server::miner	123
3.3.16.1	Classi	123
3.3.17	server::miner::fb	129
3.3.17.1	Classi	129
3.3.18	server::miner::tw	135
3.3.18.1	Classi	136
3.3.19	server::miner::ig	141
3.3.19.1	Classi	141
3.3.20	server::endpoints	147
3.3.21	server::endpoints::api	147
3.3.22	server::endpoints::api::public	148
3.3.22.1	Classi	148
3.3.23	server::endpoints::api::private	152
3.3.23.1	Classi	152
3.3.24	server::endpoints::resp	156
3.3.25	server::endpoints::resp::public	156
3.3.25.1	Classi	157
3.3.26	server::endpoints::resp::public::fb	159
3.3.26.1	Classi	159
3.3.27	server::endpoints::resp::public::tw	164
3.3.27.1	Classi	164
3.3.28	server::endpoints::resp::public::ig	168
3.3.28.1	Classi	168
3.3.29	server::endpoints::resp::private	172
3.3.29.1	Classi	173
3.4	Database	176
4	Diagrammi di sequenza	177
4.1	Client	178
4.1.1	Login	178
4.1.2	Visualizzazione delle Recipe	179
4.1.3	Generazione dei grafici	180
4.2	Server	182
4.2.1	Cron task	182

4.2.2	Miner Facebook	183
4.2.3	Login API	183
4.2.4	Recipe List API	186
5	Tracciamento	188
5.1	Classi-Requisiti	188
5.2	Requisiti-Componenti	202

Elenco delle figure

1	Package - client	6
2	Package - client::model	7
3	Package - client::model::data	8
4	Classe - client::model::data::ViewTypeModel	8
5	Classe - client::model::data::ApiDocsModel	9
6	Classe - client::model::data::UserModel	10
7	Classe - client::model::data::RecipeModel	11
8	Classe - client::model::data::RecipeRequestModel	12
9	Classe - client::model::data::RecipeInsertModule	12
10	Classe - client::model::data::MetricModel	13
11	Classe - client::model::data::CompareModel	14
12	Package - client::model	15
13	Classe - client::model::services::RadarChartCreator	16
14	Classe - client::model::services::RadarChartCreator	17
15	Classe - client::model::services::RadarChartCreator	19
16	Classe - client::model::services::UserService	20
17	Classe - client::model::services::RadarChartCreator	21
18	Classe - client::model::services::ChartCreator	21
19	Classe - client::model::services::LineChartCreator	22
20	Classe - client::model::services::BarChartCreator	23
21	Classe - client::model::services::PieChartCreator	23
22	Classe - client::model::services::MapChartCreator	24
23	Classe - client::model::services::RadarChartCreator	24
24	Classe - client::model::services::RadarChartCreator	25
25	Package - client::view	27
26	Package - client::view::public	29
27	Package - client::view::user	32
28	Package - client::view::admin	40
29	Package - client::controller	46
30	Package - client::controller::public	47
31	Classe - client::controller::public::LoginCtrl	48
32	Classe - client::controller::public::RegisterCtrl	48
33	Classe - client::controller::public::ApiDocsCtrl	49
34	Package - client::controller::user	50
35	Classe - client::controller::user::RecipeCtrl	54
36	Classe - client::controller::user::MetricsCtrl	55
37	Classe - client::controller::user::ChartsCtrl	56
38	Classe - client::controller::user::CompareCtrl	58
39	Classe - client::controller::user::RecipeRequestCtrl	60
40	Classe - client::controller::user::FavouritesCtrl	63
41	Classe - client::controller::user::TokenConfigCtrl	65
42	Classe - client::controller::user::SettingsCtrl	66
43	Classe - client::controller::user::SettingsCtrl	68
44	Package - client::controller::admin	68
45	Classe - client::controller::admin::RequestListCtrl	71
46	Classe - client::controller::admin::InsertRecipeCtrl	72
47	Classe - client::controller::admin::RatingsCtrl	75
48	Classe - client::controller::admin::UserConfigCtrl	76
49	Package - server	78

50	Package - server::db	80
51	Package - server::db::raw_data	81
52	Package - server::db::raw_data::fb	82
53	Classe - server::db::raw_data::fb::AbsFbRawData	83
54	Classe - server::db::raw_data::fb::RawFbPage	83
55	Classe - server::db::raw_data::fb::RawFbPageTrend	84
56	Classe - server::db::raw_data::fb::RawFbEvent	84
57	Classe - server::db::raw_data::fb::RawFbEventTrend	85
58	Classe - server::db::raw_data::fb::RawFbPostTrend	86
59	Package - server::db::raw_data::tw	87
60	Classe - server::db::raw_data::tw::AbsTwRawData	88
61	Classe - server::db::raw_data::tw::RawTwUser	88
62	Classe - server::db::raw_data::tw::RawTwUserTrend	89
63	Classe - server::db::raw_data::tw::RawTwUserTweet	89
64	Classe - server::db::raw_data::tw::RawTwHashtagTweet	90
65	Package - server::db::raw_data::ig	91
66	Classe - server::db::raw_data::ig::AbsIgRawData	92
67	Classe - server::db::raw_data::ig::RawIgUser	92
68	Classe - server::db::raw_data::ig::RawIgUserTrend	93
69	Classe - server::db::raw_data::ig::RawIgHashtagTrend	94
70	Classe - server::db::raw_data::ig::RawIgMedia	94
71	Package - server::db::app_data	95
72	Classe - server::db::app_data::user::UserModel	96
73	Classe - server::db::app_data::user::FavouritesModel	97
74	Classe - server::db::app_data::recipe::AbsRecipeModel	98
75	Classe - server::db::app_data::recipe::RecipeModel	99
76	Classe - server::db::app_data::recipe::RatingModel	99
77	Classe - server::db::app_data::recipe::AbsSocialMetricModel	100
78	Package - server::processor	102
79	Classe - server::processor::RequestHandler	102
80	Classe - server::processor::CommandDispatcher	103
81	Package - server::processor::commands	104
82	Classe - server::processor:: ICommand	105
83	Package - server::processor::commands::user	105
84	Package - server::processor::commands::recipe	110
85	Package - server::processor::commands::requests	112
86	Package - server::processor::commands::social	115
87	Package - server::miner	123
88	Classe - server::miner::MinerScheduler	124
89	Classe - server::miner::AbsCounter	124
90	Classe - server::miner::AbsFetcher	125
91	Classe - server::miner::SocialAuth	126
92	Classe - server::miner::GeocodeHelper	127
93	Package - server::miner::fb	128
94	Classe - server::miner::fb::AbsFbFetcher	129
95	Classe - server::miner::fb::FbPageFetcher	130
96	Classe - server::miner::fb::FbEventFetcher	131
97	Classe - server::miner::fb::AbsFbCounter	132
98	Classe - server::miner::fb::PostsCounter	133
99	Classe - server::miner::fb::CommentsCounter	134

100	Package - server::miner::tw	135
101	Classe - server::miner::tw::AbsTwFetcher	136
102	Classe - server::miner::tw::TwHashtagFetcher	136
103	Classe - server::miner::tw::TwUserFetcher	137
104	Classe - server::miner::tw::AbsTwCounter	138
105	Classe - server::miner::tw::UserTweetCounter	139
106	Classe - server::miner::tw::HashtagTweetCounter	140
107	Package - server::miner::ig	141
108	Classe - server::miner::ig::AbsIgFetcher	142
109	Classe - server::miner::ig::IgUserFetcher	142
110	Classe - server::miner::ig::IgHashtagFetcher	143
111	Classe - server::miner::ig::AbsIgCounter	144
112	Classe - server::miner::ig::MediaCounter	145
113	Package - server::endpoints	147
114	Package - server::endpoints::api	148
115	Package - server::endpoints::api::public	149
116	Classe - server::endpoints::api::public::RecipeApi	149
117	Classe - server::endpoints::api::public::FbMetricsApi	150
118	Classe - server::endpoints::api::public::TwMetricsApi	150
119	Classe - server::endpoints::api::public::IgMetricsApi	151
120	Package - server::endpoints::api::private	152
121	Classe - server::endpoints::api::private::RecipePrivateApi	153
122	Classe - server::endpoints::api::private::UserApi	153
123	Classe - server::endpoints::api::private::RecipeRequestApi	154
124	Classe - server::endpoints::api::private::FavouritesApi	155
125	Classe - server::endpoints::resp::private::FavouriteResponse	155
126	Package - server::endpoints::resp	156
127	Package - server::endpoints::resp::public	157
128	Classe - server::endpoints::resp::public::RecipeResponse	158
129	Package - server::endpoints::resp::public::fb	159
130	Classe - server::endpoints::resp::public::fb::PageResponse	159
131	Classe - server::endpoints::resp::public::fb::PageTrendResponse	160
132	Classe - server::endpoints::resp::public::fb::EventResponse	161
133	Classe - server::endpoints::resp::public::fb::EventTrendResponse	162
134	Classe - server::endpoints::resp::public::fb::PostResponse	163
135	Package - server::endpoints::resp::public::tw	164
136	Classe - server::endpoints::resp::public::tw::UserResponse	164
137	Classe - server::endpoints::resp::public::tw::UserTrendResponse	165
138	Classe - server::endpoints::resp::public::tw::HashtagResponse	166
139	Classe - server::endpoints::resp::public::tw::TweetResponse	167
140	Package - server::endpoints::resp::public::ig	168
141	Classe - server::endpoints::resp::public::ig::UserResponse	168
142	Classe - server::endpoints::resp::public::ig::UserTrendResponse	169
143	Classe - server::endpoints::resp::public::ig::HashtagResponse	170
144	Classe - server::endpoints::resp::public::ig::HashtagTrendResponse	171
145	Classe - server::endpoints::resp::public::ig::MediaResponse	171
146	Package - server::endpoints::resp::private	173
147	Classe - server::endpoints::resp::private::RecipeReqResponse	173
148	Classe - server::endpoints::resp::private::UserResponse	174
149	Classe - server::endpoints::resp::private::FavouriteResponse	175

150	Diagramma di sequenza - Login lato client	178
151	Diagramma di sequenza - Visualizzazione delle Recipe	179
152	Diagramma di sequenza - Generazione dei grafici	181
153	Diagramma di sequenza - Cron task	182
154	Diagramma di sequenza - Miner Facebook	184
155	Diagramma di sequenza - Login API	185
156	Diagramma di sequenza - Recipe List API	187

1 Introduzione

1.1 Scopo del documento

Questo documento ha come scopo quello di definire la progettazione di dettaglio per il prodotto BDSMApp. Verrà presentata la struttura e le relazioni tra le componenti a basso livello. Queste serviranno ai *Programmatori* come guida per le attività di codifica.

1.2 Scopo del prodotto

Lo scopo del prodotto è di creare una nuova infrastruttura che permetta di interrogare Big Data recuperati dai social network, quali: Facebook, Twitter, Instagram. L'applicazione sarà composta da due parti:

- consultazione e interrogazione con interfaccia web per utente;
- servizi web REST interrogabili.

1.3 Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio usato nei documenti viene allegato il *Glossario v4.0.0*. Esso ha lo scopo di definire ed analizzare tutti i termini tecnici del progetto e di fugare eventuali ambiguità fornendo un'accurata descrizione. Tutte le occorrenze di tali termini nei documenti verranno contrassegnate con una “G” a pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Analisi dei Requisiti:** *Analisi dei Requisiti v6.0.0*
- **Norme di Progetto:** *Norme di Progetto v6.0.0*
- **Specifiche Tecniche:** *Specifiche Tecniche v3.0.0*

1.4.2 Informativi

- **ControllerAs syntax:**
<http://www.johnpapa.net/angularjss-controller-as-and-the-vm-variable/>
- **Promise AngularJS_G:**
<http://www.webdeveeasy.com/javascript-promises-and-angularjs-q-service/>
- **John Papa style guide:**
<https://github.com/johnpapa/angular-styleguide>
- **ng-auth doc:**
<https://github.com/lynndylanhurley/ng-token-auth>
- **HTTPS su GitHub_G pages:**
<https://konklone.com/post/github-pages-now-sorta-supports-https-so-use-it>
- **Google Datastore:**
<https://cloud.google.com/appengine/docs/python/ndb/>

- **Google Cloud Endpoints:**
<https://cloud.google.com/appengine/docs/python/endpoints/>
- **Google Cron:**
<https://cloud.google.com/appengine/docs/python/config/cron>
- **Api facebook (facebook-sdk):**
<https://github.com/pythonforfacebook/facebook-sdk>
- **Api twitter (tweepy):**
<http://tweepy.readthedocs.org/en/v3.2.0/>
- **Api instagram (python-instagram):**
<https://github.com/Instagram/python-instagram>

2 Standard di progetto

2.1 Progettazione architetturale

Gli standard relativi alla progettazione ad alto livello dell'architettura sono definite nella *Specifica Tecnica v3.0.0*.

2.2 Documentazione del codice

Gli standard relativi alla documentazione del codice sono definiti nelle *Norme di Progetto v6.0.0*.

2.3 Denominazione di entità e relazioni

Entità e relazioni devono avere nomi chiari ed autoesplicativi, preferendo sostantivi per i primi e verbi per i secondi. Per ulteriori informazioni si faccia riferimento alle *Norme di Progetto v6.0.0*.

2.4 Codifica

Gli standard relativi alla codifica sono definiti nelle *Norme di Progetto v6.0.0*.

2.5 Strumenti di lavoro

Gli strumenti di lavoro e le procedure che ogni *Programmatore* deve seguire per un corretto sviluppo del prodotto sono definite nelle *Norme di Progetto v6.0.0*.

3 Specifica Componenti

3.1 Note per la lettura

- Nell'esposizione delle caratteristiche di ciascuna classe si utilizzerà la marcatura “N/A” per identificare l'assenza di attributi, metodi o entrambi nella classe descritta. L'assenza di ambedue però è presente solo in alcune classi identificate nel “client” che non possono venire codificate secondo lo schema tradizionale come quelle presenti nella sezione ?? o nei classi che gestiscono il routing delle pagine;
- Nella descrizione degli attributi e dei metodi per ogni classe, se non specificato, si assuma che l'elemento descritto sia marcato come “private”;
- Nella sezione “server::processor_G::commands” tutte le classi di comando, non hanno un diagramma UML_G associato in quanto presentano tutte la medesima struttura, composta da un solo metodo uguale per tutte, come indicato del diagramma ad inizio sezione.

3.2 Client (Front-end)

Nel descrivere le componenti si parlerà di classi, che però saranno da intendersi nella loro accezione logica. Il linguaggio di programmazione impiegato infatti non possiede la struttura di classe, ma solo di oggetto, sarà quindi compito del codificatore ripor-tare le classi di seguito identificate nelle strutture fornite dallo specifico linguaggio. Al fine di rendere maggiormente comprensibile la descrizione testuale di alcune classi abbiamo deciso di presentare ora alcuni dei servizi che utilizzeremo in parte delle classi presenti che ci vengono offerti direttamente dal framework_G AngularJS_G o da qualche modulo esterno utilizzato.

- **\$scope:** rappresenta un contesto di esecuzione dove sono definiti e poi utilizzati dati necessari all'aggiornamento della view_G e alle elaborazioni dei controller. Per questo motivo viene spesso utilizzato come argomento delle funzioni nel controller. Esso è responsabile del two-way data binding.
Nonostante quanto detto però utilizziamo nella specifica questa notazione perché più familiare con il framework_G scelto in particolare quando definiamo gli **Attributi associati allo \$scope** e i **Metodi associati allo \$scope**. La reale implementazione utilizza la sintassi esposta nel template_G relativo ai controller esposto nell'appendice delle *Norme di Progetto v6.0.0*. Questo ci consente di non dover utilizzare l'oggetto \$scope in maniera esplicita e permette di mantenere il codice più pulito, efficiente e manutenibile;
- **\$http:** gestisce le richieste del front-end per il back-end ed inoltre gestisce le risposte di quest'ultimo utilizzando il protocollo HTTP. Questo servizio viene utilizzato nei servizi definiti nel package_G **client::model::services**;
- **\$stateProvider:** utilizzato per implementare la navigazione tra le varie pagine del nostro sito. Viene utilizzato in ogni classe di routing nel controller per il reindirizzamento alla pagina adeguata e per poter assegnare un controller personalizzato ad ogni pagina nel caso fosse necessario. Questo servizio ci viene offerto dal modulo **ui-router**;
- **\$stateParams:** utilizzato per recuperare dei parametri passati tra una pagina e l'altra grazie all'utilizzo del modulo **ui-router**;
- **\$location:** permette di manipolare la URL del browser;
- **\$on:** rappresenta un ascoltatore di eventi che è in ascolto di un segnale emesso dal broadcast;

3.2.1 client

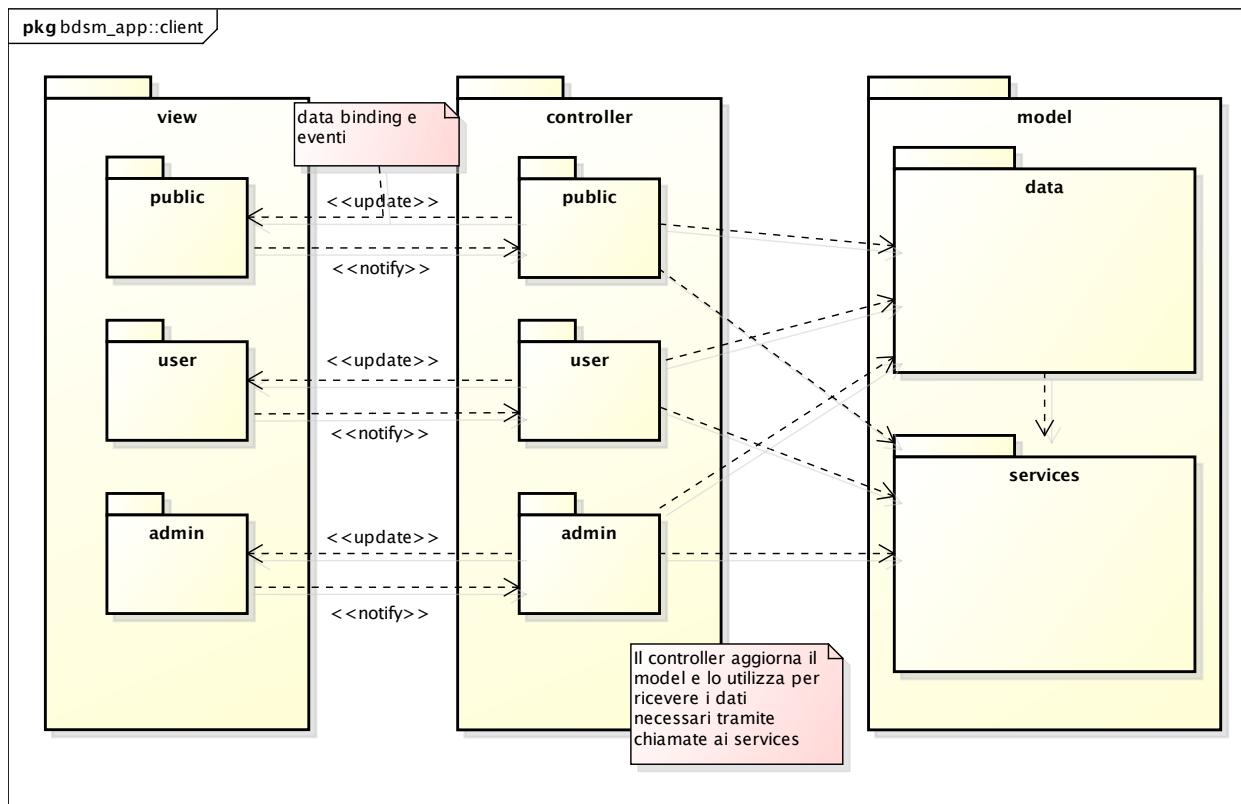


Figura 1: Package - client

- **Descrizione:** è il package_G che racchiude tutte le parti del front-end. È quindi l'insieme dei componenti che viene eseguito nel browser degli utenti normali e degli amministratori, fornendo loro un'interfaccia grafica per interagire con il sistema;
- **Package_G contenuti:**
 - client::model
 - client::view_G
 - client::controller

3.2.2 client::model

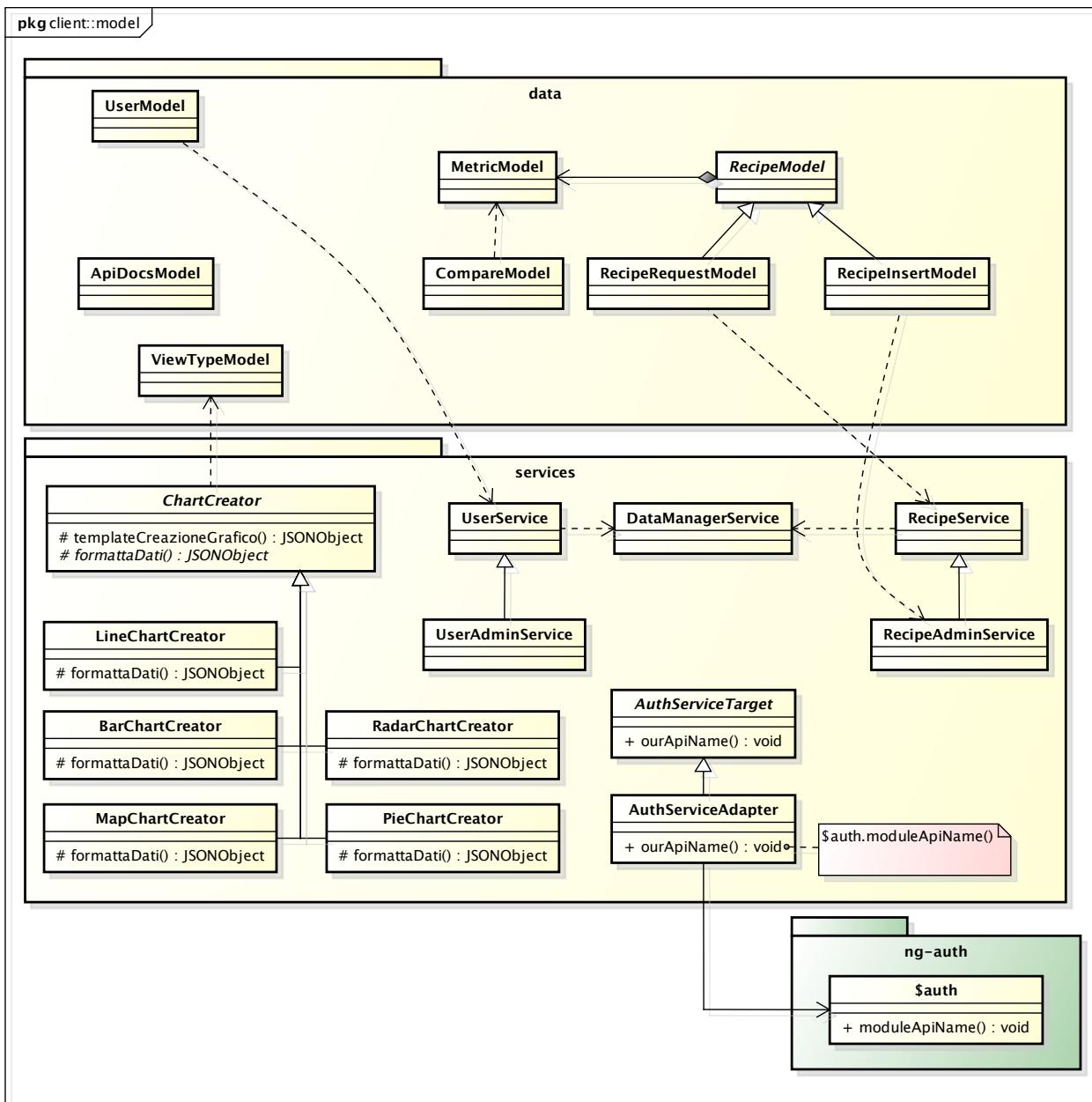


Figura 2: Package - client::model

- **Descrizione:** è il package_G che contiene sia le classi dei modelli di dati utilizzati dal client sia i servizi che mettono in comunicazione esso con il server attraverso i diversi servizi REST_G;
- **Padre:** client;
- **Package_G contenuti:**
 - client::model::data
 - client::model::services
- **Interazione con altri componenti:**

- client::controller

3.2.3 client::model::data

Le classi di questo package_G sono state progettate, ma si prevede che non verranno codificate poiché verrà sfruttato lo stile duck-typing_G della gestione dei tipi di JavaScript_G. Solo le classi **client::model::data::ViewTypeModel** e **client::model::data::ApiDocsModel** saranno implementate in quanto si è scelto di tenere i dati direttamente nel modello non ritenendo necessario avere un collegamento al server per questo tipo di dati. Per tutte le altre classi quindi verranno solo forniti gli attributi che ogni oggetto che vuole rappresentare quel tipo dovrà avere.

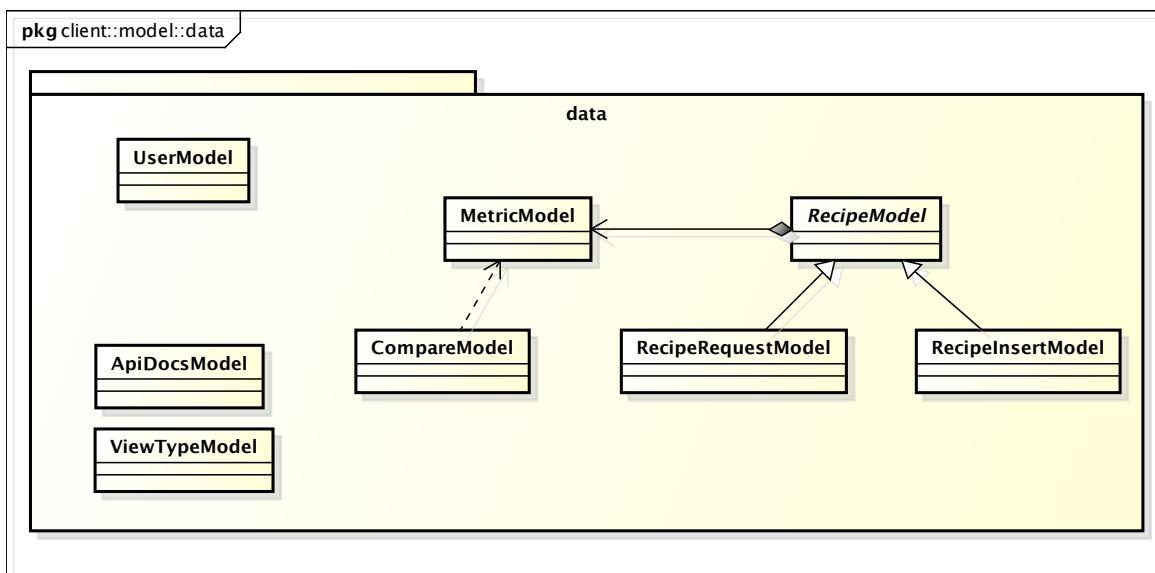


Figura 3: Package - client::model::data

- **Descrizione:** è il package_G che contiene le classi che rappresentano i modelli dei dati utilizzati dal client;
- **Padre:** client::model
- **Interazione con altri componenti:**
 - client::model::services
 - client::controller

3.2.3.1 Classi

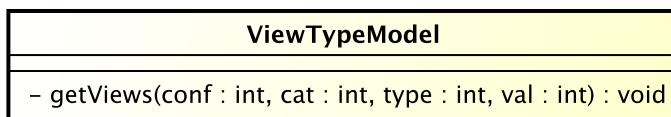


Figura 4: Classe - client::model::data::ViewTypeModel

3.2.3.1.1 client::model::data::ViewTypeModel

- **Descrizione:** è la classe che rappresenta i tipi di View_G che vengono generati a seconda delle metriche;
- **Utilizzo:** viene utilizzata dal controller delle View_G (ChartsCtrl) per capire quali grafici generare;
- **Relazioni con altre classi:**
 - client::controller::user::ChartsCtrl
- **Attributi:** N/A
- **Metodi:**
 - – `getViews(conf, cat, type, val, viewType)`

Descrizione: metodo che restituisce un array di funzioni che verranno applicate dalle classi figlie di ChartCreator per ottenere i vari tipi di grafico da generare (views) in base alla categoria_G e al tipo della metrica_G, e in base al fatto che si tratti di un grafico di confronto o meno. L'argomento conf (booleano) determina se bisogna generare un confronto, cat e type contengono la categoria e il tipo della metrica, mentre val il suo valore. Nel caso venga specificata la stringa viewType viene restituita solo la funzione relativa alla stringa passata. Si noti che questa classe è in realtà suddivisa in una classe base e le sei sottoclassi viewTypeFbSingleModel, viewTypeFbConfModel, viewTypeTwSingleModel, viewTypeTwConfModel, viewTypeIgSingleModel, viewTypeIgConfModel (view_G singole e confronti per ogni social), in quanto la classe sarebbe diventata troppo grossa per linee di codice e SLOC. Logicamente però il sistema vede queste classi come la sola classe ViewTypeModel, ragion per cui non sono state descritte singolarmente queste classi.
- **Servizi AngularJS_G utilizzati:** N/A

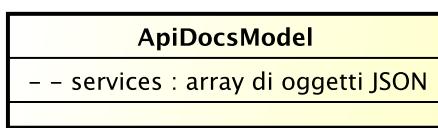


Figura 5: Classe - client::model::data::ApiDocsModel

3.2.3.1.2 client::model::data::ApiDocsModel

- **Descrizione:** è la classe che rappresenta la struttura della documentazione di una API_G;
- **Utilizzo:** viene utilizzata dai controller ApiDocsController e TokenConfigCtrl per generare la documentazione dei servizi pubblici che vengono messi a disposizione per l'interrogazione;
- **Relazioni con altre classi:**
 - client::controller::public::ApiDocsCtrl
 - client::controller::user::TokenConfigCtrl

- **Attributi:**

- `- services : array di oggetti JSONG`

Descrizione: questo attributo contiene la descrizione dei vari servizi REST_G pubblici dell'applicazione e ne costituisce la documentazione.
 Il formato dei JSON_G che contiene è il seguente:

```
{
    req: "ig/users/{user_id}",
    type: 'GET',
    desc: 'Get all the data associated with an Instagram user'
}
```

- **Metodi:** N/A

- **Servizi AngularJS_G utilizzati:** N/A

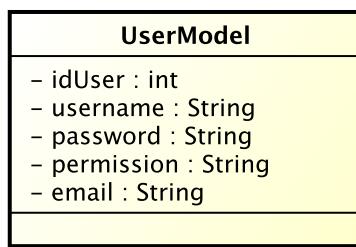


Figura 6: Classe - client::model::data::UserModel

3.2.3.1.3 client::model::data::UserModel

- **Descrizione:** è la classe che rappresenta la struttura dati relativa agli utenti;
- **Utilizzo:** viene utilizzata da diversi servizi e controller per accedere agli attributi associati all'utente;

- **Relazioni con altre classi:**

- client::model::services::AuthService
- client::model::services::UserService
- client::model::services::UserAdminService
- client::controller::admin::RecipeConfigCtrl
- client::controller::admin::InsertRecipeCtrl
- client::controller::user::RecipeRequestCtrl
- client::controller::user::SettingsCtrl
- client::controller::user::TokenConfigCtrl

- **Attributi:**

- `- idUser : numero`

Descrizione: identificativo dell'utente all'interno del sistema

- `- username : stringa`

Descrizione: username dell'utente

- `- password : stringa`

Descrizione: chiave di accesso dell'utente.

- – `permission : stringa`

Descrizione: descrive se l'utente è un utente normale o amministratore.

- – `email : stringa`

Descrizione: l'indirizzo email dell'utente.

- **Metodi:** N/A

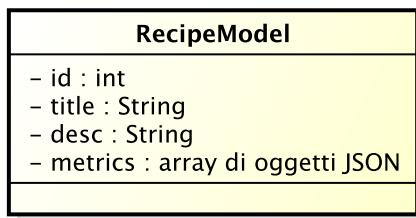


Figura 7: Classe - client::model::data::RecipeModel

3.2.3.1.4 client::model::data::RecipeModel

- **Descrizione:** è la classe che rappresenta la struttura base dei dati relativa ad una Recipe_G;

- **Utilizzo:** viene utilizzata come classe astratta da RecipeRequestModel e RecipeInsertModel che andranno ad estenderla in maniera opportuna a seconda delle diverse esigenze;

- **Relazioni con altre classi:**

- client::model::data::MetricModel
- client::model::data::RecipeRequestModel
- client::model::data::RecipeInsertModel

- **Attributi:**

- – `id : intero`

Descrizione: identificativo della Recipe_G all'interno del sistema

- – `title : stringa`

Descrizione: nome della Recipe_G

- – `desc : stringa`

Descrizione: descrizione del contenuto della Recipe_G.

- – `metrics : array di oggetti JSONG nella forma descritta da MetricModel`

Descrizione: la lista delle metriche che fanno parte della recipe_G.

- **Metodi:** N/A

RecipeRequestModel
- title : String - desc : String - metrics : array di oggetti JSON - user_id : int

Figura 8: Classe - client::model::data::RecipeRequestModel

3.2.3.1.5 client::model::data::RecipeRequestModel

- **Descrizione:** è la classe che rappresenta la struttura dei dati relativa ad una richiesta di una nuova Recipe_G;
- **Utilizzo:** viene utilizzata per creare l'oggetto che dovrà essere inviato qualora l'utente volesse richiedere l'inserimento di una nuova Recipe_G;
- **Classi ereditate:** client::model::data::RecipeModel;
- **Relazioni con altre classi:**
 - client::model::services::RecipeService
 - client::controller::RecipeRequestCtrl
- **Attributi:**
 - `- title : stringa`
Descrizione: nome della Recipe_G
 - `- desc : stringa`
Descrizione: descrizione del contenuto della Recipe_G.
 - `- metrics : array di oggetti JSON, nella forma descritta da MetricModel`
Descrizione: la lista delle metriche che fanno parte della recipe_G.
 - `- user_id : intero`
Descrizione: identificativo dell'utente che invia la richiesta.
- **Metodi:** N/A

RecipeInsertModel
- desc:String : String - title : String - metrics : array di oggetti JSON - admin_id : int

Figura 9: Classe - client::model::data::RecipeInsertModule

3.2.3.1.6 client::model::data::RecipeInsertModel

- **Descrizione:** è la classe che rappresenta la struttura dei dati relativa ad un inserimento di una nuova Recipe_G;
- **Utilizzo:** viene utilizzata per creare l'oggetto che dovrà essere inviato qualora l'amministratore volesse andare ad inserire una nuova Recipe_G;

- **Classi ereditate:** client::model::data::RecipeModel;
- **Relazioni con altre classi:**
 - client::controller::RecipeConfigCtrl
 - client::controller::InsertRecipeCtrl
- **Attributi:**
 - `- title : stringa`
Descrizione: nome della Recipe_G
 - `- desc : stringa`
Descrizione: descrizione del contenuto della Recipe_G.
 - `- metrics : array di oggetti JSON, nella forma descritta da MetricModel`
Descrizione: la lista delle metriche che fanno parte della recipe_G.
 - `- admin_id : intero`
Descrizione: identificativo dell'amministratore che aggiunge la richiesta.
- **Metodi:** N/A

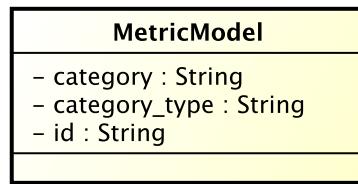


Figura 10: Classe - client::model::data::MetricModel

3.2.3.1.7 client::model::data::MetricModel

- **Descrizione:** è la classe che rappresenta la struttura dei dati relativa ad un metrica_G di una Recipe_G;
- **Utilizzo:** viene utilizzata per creare l'oggetto che sarà contenuto almeno una volta nel modello dei dati relativo alle Recipe_G;
- **Relazioni con altre classi:**
 - client::model::data::RecipeModel
 - client::controller::MetricsCtrl
 - client::controller::RecipeConfigCtrl
 - client::controller::InsertRecipeCtrl
 - client::controller::RecipeRequestCtrl
- **Attributi:**
 - `- category : stringa`
Descrizione: social network al quale la metrica_G si riferisce
 - `- category_type : stringa`

Descrizione: tipo di metrica_G.

- – `id : stringa`

Descrizione: valore della metrica_G.

- **Metodi:** N/A



Figura 11: Classe - client::model::data::CompareModel

3.2.3.1.8 client::model::data::CompareModel

- **Descrizione:** è la classe che rappresenta la struttura dei dati relativa ad un confronto tra diverse metriche;

- **Utilizzo:** viene utilizzata per creare l'oggetto che sarà contenuto almeno una volta nel modello dei dati relativo alle Recipe_G;

- **Relazioni con altre classi:**

- client::model::services::RecipeService
- client::controller::user::CompareCtrl

- **Attributi:**

- – `metrics : array di oggetti JSONG nella forma descritta da MetricModel`

Descrizione: la lista delle metriche che fanno parte del confronto.

- **Metodi:** N/A

3.2.4 client::model::services

- **Descrizione:** è il package_G che contiene le classi che sviluppano il core business dell'applicazione: in particolare gestiscono il recupero dei dati, siano essi salvati in locale o da recuperare chiamando il server, la creazione dei grafici e l'autenticazione. Tutte le classi contenute in questo package implementano il pattern *Singleton*;

- **Padre:** client::model

- **Interazione con altri componenti:**

- client::model::data
- client::controller

3.2.4.1 Classi

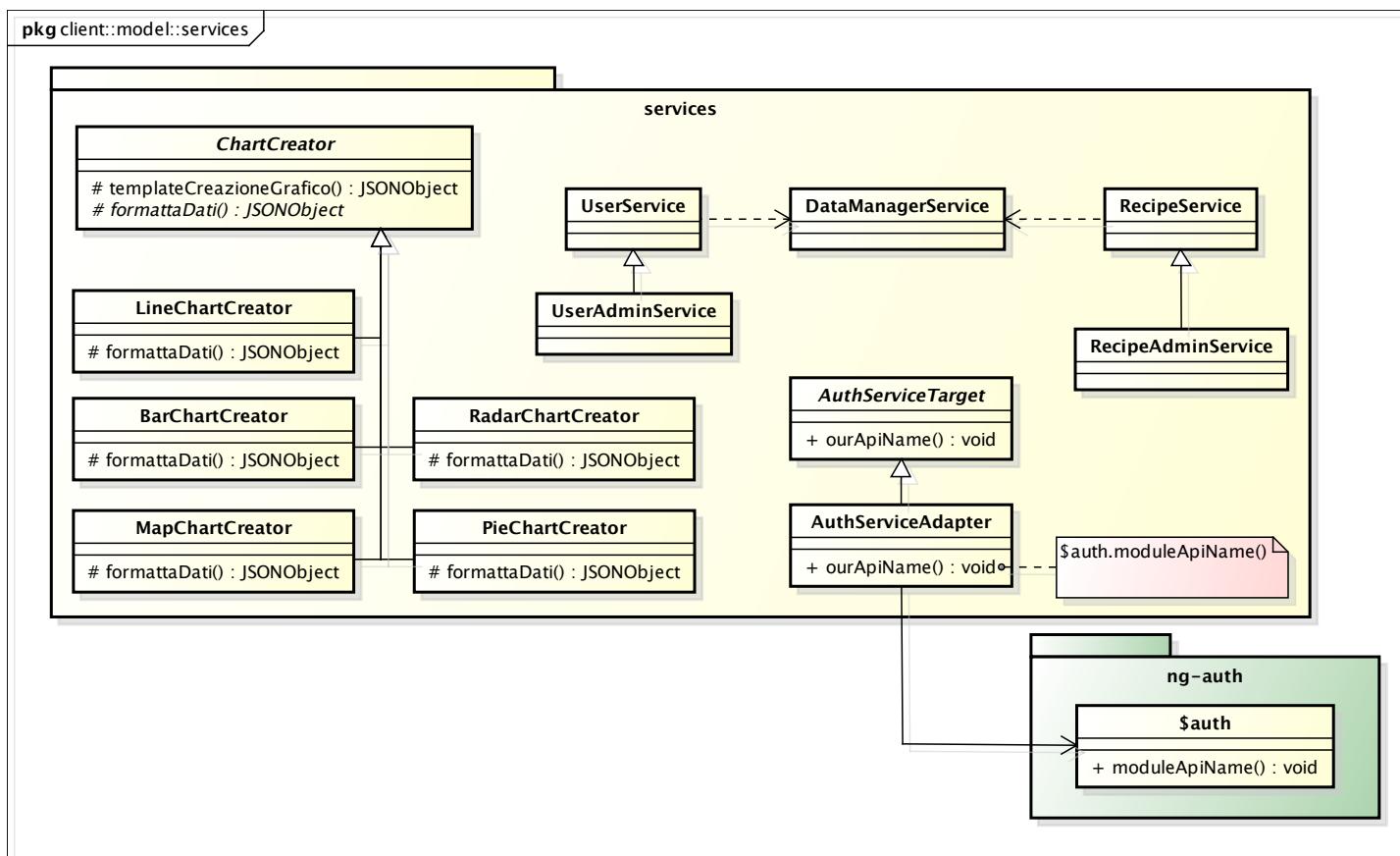


Figura 12: Package - client::model

3.2.4.1.1 client::model::services::AuthServiceTarget

- **Descrizione:** è la classe di interfaccia per gestire le chiamate al modulo esterno usato per gestire l'autenticazione secondo il pattern *Adapter* ;
- **Utilizzo:** Questa classe non è stata implementata in quanto non esistono interfacce in JavaScript_G, e grazie alla Dependency Injection è stato possibile iniettare direttamente la classe AuthServiceAdapter dove è stato necessario.

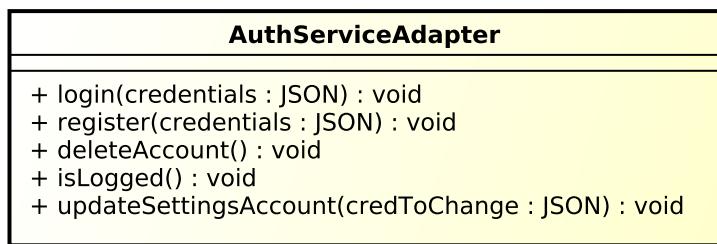


Figura 13: Classe - client::model::services::RadarChartCreator

3.2.4.1.2 client::model::services::AuthServiceAdapter

- **Descrizione:** è la classe *Adapter* che permette di invocare i metodi del modulo di autenticazione esterno invocando i metodi come presentati in AuthServiceTarget;
- **Utilizzo:** i metodi di questa classe saranno invocati quando un utente effettua la registrazione al servizio o effettua un login/logout. Questa classe è stata implementata col nome più semplice di AuthService in quanto non è necessario distinguerla dalla classe Target (non implementata);

- **Relazioni con altre classi:**

- client::controller::public::LoginCtrl
- client::controller::public::RegisterCtrl
- client::controller::user::LogoutCtrl
- ng-auth::\$auth

- **Attributi:** N/A

- **Metodi:**

- **+ login(credentials)**

Descrizione: metodo per effettuare il login. Invia al server il JSON_G credentials, nella forma:

```

{
  email: "info@mashup-unipd.it",
  pwd: 'password'
}
  
```

Le credenziali non vengono criptate in quanto questa operazione non porta nessun vantaggio se fatta lato client. Questo tipo di sicurezza è garantita solo tramite l'utilizzo del protocollo **HTTPS (HyperText Transfer Protocol over Secure Socket Layer)**. I Google Cloud Endpoints che utilizziamo implementano questo protocollo ed

esso viene implementato anche sul nostro sito web attraverso le tecniche descritte nell'articolo seguente: <https://konkclone.com/post/github-pages-now-sorta-supports-https-so-use-it>

– + `register(credentials)`

Descrizione: metodo per effettuare la registrazione. Invia al server il JSON_G credentials, nella forma:

```
{
    username: "MashUp",
    email: "info@mashup-unipd.it",
    pwd: 'password'
}
```

– + `deleteAccount()`

Descrizione: metodo per eliminare l'account che invia la richiesta al server.

– + `isLoggedIn()`

Descrizione: metodo che restituisce un booleano true se l'utente è autenticato al servizio.

– + `updateSettingsAccount(credToChange)`

Descrizione: metodo per cambiare le proprie credenziali. Invia al server un JSON_G nella forma seguente:

```
{
    username: "MashUp",
    email: "info@mashup-unipd.it",
    pwd: 'password'
}
```

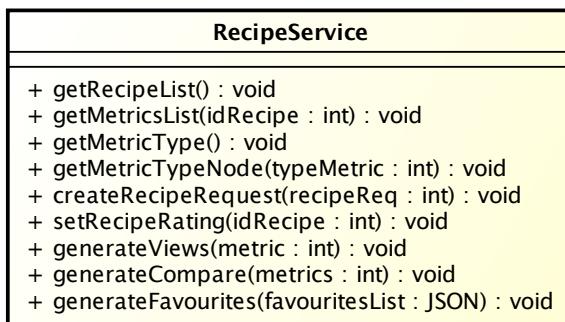


Figura 14: Classe - client::model::services::RadarChartCreator

3.2.4.1.3 client::model::services::RecipeService

- **Descrizione:** è la classe che racchiude i metodi per reperire i dati relativi alle Recipe_G, siano essi salvati in locale o da recuperare dal server, inoltre gestisce l'invio di richieste di Recipe al server;
- **Utilizzo:** i suoi metodi vengono invocati ogni volta che sia necessario avere dei dati relativi a una o più Recipe_G per un utente non amministratore, o se un utente non amministratore desidera inviare una richiesta di Recipe;
- **Relazioni con altre classi:**

- client::model::data::RecipeRequestModel
- client::model::services::ManagerDataExpirationDate
- client::controller::user::RecipeCtrl
- client::controller::user::MetricsCtrl
- client::controller::user::ChartsCtrl
- client::controller::user::CompareCtrl

- **Attributi:** N/A

- **Metodi:**

- + `getRecipeList()`

Descrizione: metodo per ottenere la lista delle recipe_G disponibili.

- + `getMetricsList(idRecipe)`

Descrizione: metodo per ottenere la lista delle metriche dato l'id di una recipe_G.

- + `getMetricType()`

Descrizione: metodo che restituisce la lista delle categorie presenti (i vari social).

- + `getMetricTypeNode(typeMetric)`

Descrizione: metodo che la lista dei tipi di metriche data la categoria_G.

- + `createRecipeRequest(recipeReq)`

Descrizione: metodo per inviare una nuova richiesta di ricetta. Invia un JSON_G nella forma descritta dalla classe RecipeRequestModel.

- + `setRecipeRating(idRecipe)`

Descrizione: metodo per impostare un rating a una recipe_G e inviare il voto al server.

- + `generateViews(metric,typeView)`

Descrizione: metodo che data una metrica_G (nel formato descritto da MetricModel) restituisce un promessa di un array di promesse contenenti l'html per generare i grafici associati a quella metrica. Riceve opzionalmente il nome di una funzione tratta da ViewTypeModel per generare un solo grafico specifico.

- + `generateCompare(metrics,typeView)`

Descrizione: metodo che dato un array di metriche (nel formato descritto da MetricModel) restituisce un promessa di un array di promesse contenenti l'html per generare i grafici di confronto di quelle metriche. Riceve opzionalmente il nome di una funzione tratta da ViewTypeModel per generare un solo grafico specifico.

- + `generateFavourites(favouritesList)`

Descrizione: metodo che dato un array di favoriti restituisce una promessa di array di promesse contenenti l'html per generare i grafici segnati come favoriti. Riceve in input la lista dei favoriti come essa viene restituita dal server.

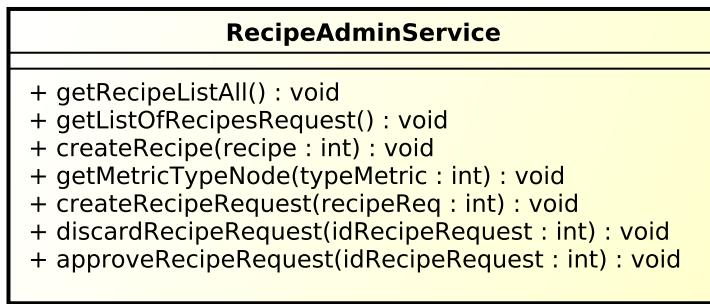


Figura 15: Classe - client::model::services::RadarChartCreator

3.2.4.1.4 client::model::services::RecipeAdminService

- **Descrizione:** è la classe che oltre a gestire ciò che è gestito dalla classe padre si occupa dell'inserimento di nuove recipe_G nel server e dell'eliminazione di Recipe_G già presenti;
- **Utilizzo:** i suoi metodi vengono invocati quando un utente amministratore desidera agire su dati di Recipe_G da pagine non accessibili a un utente non amministratore;
- **Classi ereditate:**
 - client::model::services::RecipeService
- **Relazioni con altre classi:**
 - client::model::data::RecipeInsertModel
 - client::model::services::ManagerDataExpirationDate
 - client::controller::admin::RecipeConfigCtrl
 - client::controller::admin::InsertRecipeCtrl
- **Attributi:** N/A
- **Metodi:**
 - `+ getRecipeListAll()`
Descrizione: metodo per ottenere la lista delle recipe_G disponibili con tutti i dati ad esse associate (inclusi i ratings).
 - `+ getListOfRecipesRequest()`
Descrizione: metodo per ottenere la lista delle richieste di recipe_G.
 - `+ createRecipe(recipeG)`
Descrizione: metodo per aggiungere una ricetta al sistema (l'argomento è nella forma descritta da RecipeModel).
 - `+ getMetricTypeNode(typeMetric)`
Descrizione: metodo che la lista dei tipi di metriche data la categoria_G.
 - `+ createRecipeRequest(recipeReq)`
Descrizione: metodo per inviare una nuova richiesta di ricetta. Invia un JSON_G nella forma descritta dalla classe RecipeRequestModel.
 - `+ discardRecipeRequest(idRecipeRequest)`

Descrizione: metodo che elimina una richiesta di Recipe_G (dato il suo id).

– + `approveRecipeRequest(idRecipeRequest)`

Descrizione: metodo che genera una Recipe_G come descritta da una richiesta di recipe_G (di cui è dato l'id), eliminando poi quella richiesta.

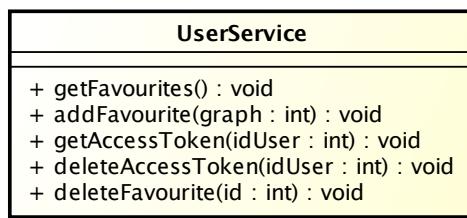


Figura 16: Classe - client::model::services::UserService

3.2.4.1.5 client::model::services::UserService

- **Descrizione:** è la classe che gestisce il recupero e la modifica di dati utente relativi a sé stessi;

- **Utilizzo:** i suoi metodi vengono invocati quando un utente accede alla pagina dei propri dati personali;

- **Relazioni con altre classi:**

- client::model::data::UserModel
- client::model::services::ManagerDataExpirationDate
- client::controller::user::SettingsCtrl

- **Attributi:** N/A

- **Metodi:**

- + `getFavourites()`

Descrizione: metodo che restituisce la lista dei grafici favoriti.

- + `deleteFavourite(id)`

Descrizione: metodo che richiede la rimozione di un grafico dai favoriti.

Riceve l'id del favorito da eliminare.

- + `addFavourite(metrics, info)`

Descrizione: metodo per aggiungere ai favoriti un grafico. Riceve un array di MetricModel, e il nome di una funzione tratta da viewType-Model

- + `getAccessToken(idUser)`

Descrizione: metodo per generare un token di accesso alle api_G. Prende come argomento l'id dell'utente per cui generare il token. Questo metodo chiama il servizio **DataManagerService** per effettuare il collegamento con il **server**, il quale genererà il token secondo lo schema previsto e lo invierà in risposta al client associandolo alle informazioni dell'utente;

- + `deleteAccessToken(idUser)`

Descrizione: metodo che elimina il token generato per l'utente identificato da idUser.



Figura 17: Classe - client::model::services::RadarChartCreator

3.2.4.1.6 client::model::services::UserAdminService

- **Descrizione:** è la classe che oltre a gestire ciò che è gestito dalla sua classe padre si occupa del recupero e della modifica di dati di utenti, inclusa l'eliminazione di un utente e il cambiare i suoi permessi;
- **Utilizzo:** i suoi metodi vengono invocati quando un utente amministratore agisce su dati relativi ad utenti da una pagina non accessibile agli utenti non amministratori;
- **Classi ereditate:**
 - client::model::services::UserService
- **Relazioni con altre classi:**
 - client::model::data::UserModel
 - client::model::services::ManagerDataExpirationDate
 - client::controller::user::UserConfigCtrl
- **Attributi:** N/A
- **Metodi:**
 - **+ getListOfUsers()**

Descrizione: metodo per ottenere la lista degli utenti.

- **+ editUserPermissions(idUser)**

Descrizione: metodo per cambiare i permessi di un utente.

- **+ deleteUserAccount(idUser)**

Descrizione: metodo per eliminare un utente.

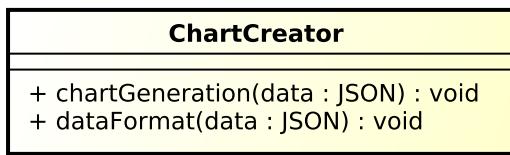


Figura 18: Classe - client::model::services::ChartCreator

3.2.4.1.7 client::model::services::ChartCreator

- **Descrizione:** è una classe astratta che descrive il codice comune delle classi che generano un grafico, secondo il pattern *template G method*;

- **Utilizzo:** i suoi metodi vengono invocati dalle classi figlie nella creazione di grafici;
- **Relazioni con altre classi:**
 - client::model::data::ViewTypeModel
 - client::model::services::LineChartCreator
 - client::model::services::BarChartCreator
 - client::model::services::MapChartCreator
 - client::model::services::PieChartCreator
 - client::model::services::RadarChartCreator
 - client::controller::user::ChartsCtrl
 - client::controller::user::CompareCtrl
- **Attributi:** N/A
- **Metodi:**
 - + `chartGeneration(data, info)`

Descrizione: metodo che definisce le operazioni comuni di generazione di un grafico. Riceve com argomento una promessa di risultati di una chiamata REST_G (data) e una funzione estratta da ViewTypeModel. Restituisce una promessa del codice html di un grafico.

 - + `dataFormat(data)`

Descrizione: metodo per portare i dati nel formato corretto per lo specifico tipo di grafico. Riceve una funzione estratta da ViewTypeModel.

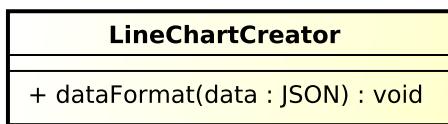


Figura 19: Classe - client::model::services::LineChartCreator

3.2.4.1.8 client::model::services::LineChartCreator

- **Descrizione:** la classe contiene i metodi per la creazione di un Line Chart;
- **Utilizzo:** i suoi metodi vengono invocati quando si vuole creare un Line Chart;
- **Classi ereditate:**
 - client::model::services::ChartCreator
- **Relazioni con altre classi:**
 - client::controller::user::ChartsCtrl
 - client::controller::user::CompareCtrl
- **Attributi:** N/A
- **Metodi:**

- + `dataFormat(data)`

Descrizione: metodo per portare i dati nel formato corretto per un line chart.
 Riceve un JSON_G nella forma descritta per il metodo chartGeneration.

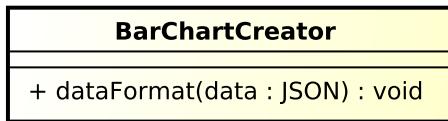


Figura 20: Classe - client::model::services::BarChartCreator

3.2.4.1.9 client::model::services::BarChartCreator

- **Descrizione:** la classe contiene i metodi per la creazione di un Bar Chart;
- **Utilizzo:** i suoi metodi vengono invocati quando si vuole creare un Bar Chart;
- **Classi ereditate:**
 - client::model::services::ChartCreator
- **Relazioni con altre classi:**
 - client::controller::user::ChartsCtrl
 - client::controller::user::CompareCtrl
- **Attributi:** N/A
- **Metodi:**
 - + `dataFormat(data)`

Descrizione: metodo per portare i dati nel formato corretto per un bar chart.
 Riceve un JSON_G nella forma descritta per il metodo chartGeneration.

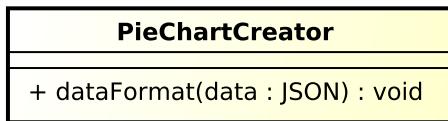


Figura 21: Classe - client::model::services::PieChartCreator

3.2.4.1.10 client::model::services::PieChartCreator

- **Descrizione:** la classe contiene i metodi per la creazione di un Pie Chart;
- **Utilizzo:** i suoi metodi vengono invocati quando si vuole creare un Pie Chart;
- **Classi ereditate:**
 - client::model::services::ChartCreator
- **Relazioni con altre classi:**
 - client::controller::user::ChartsCtrl
 - client::controller::user::CompareCtrl

- **Attributi:** N/A
- **Metodi:**
- + `dataFormat(data)`

Descrizione: metodo per portare i dati nel formato corretto per un pie chart.
 Riceve un JSON_G nella forma descritta per il metodo chartGeneration.

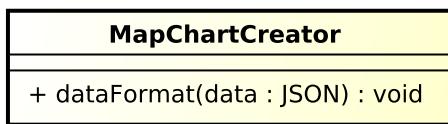


Figura 22: Classe - client::model::services::MapChartCreator

3.2.4.1.11 client::model::services::MapChartCreator

- **Descrizione:** la classe contiene i metodi per la creazione di un Map Chart;
- **Utilizzo:** i suoi metodi vengono invocati quando si vuole creare un Map Chart;
- **Classi ereditate:**
 - client::model::services::ChartCreator
- **Relazioni con altre classi:**
 - client::controller::user::ChartsCtrl
 - client::controller::user::CompareCtrl
- **Attributi:** N/A
- **Metodi:**
- + `dataFormat(data)`

Descrizione: metodo per portare i dati nel formato corretto per un map chart. Riceve un JSON_G nella forma descritta per il metodo chartGeneration.

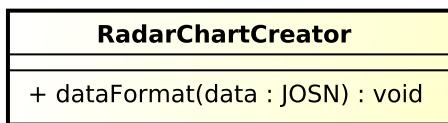


Figura 23: Classe - client::model::services::RadarChartCreator

3.2.4.1.12 client::model::services::RadarChartCreator

- **Descrizione:** la classe contiene i metodi per la creazione di un Radar Chart;
- **Utilizzo:** i suoi metodi vengono invocati quando si vuole creare un Radar Chart;
- **Classi ereditate:**
 - client::model::services::ChartCreator

- **Relazioni con altre classi:**

- client::controller::user::ChartsCtrl
- client::controller::user::CompareCtrl

- **Attributi:** N/A

- **Metodi:**

- + `dataFormat(data)`

Descrizione: metodo per portare i dati nel formato corretto per un radar chart. Riceve un JSON_G nella forma descritta per il metodo chartGeneration.

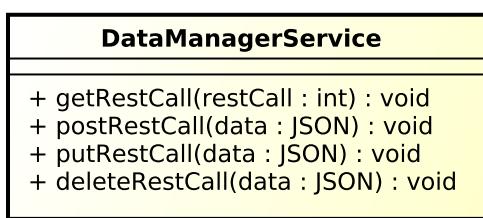


Figura 24: Classe - client::model::services::RadarChartCreator

3.2.4.1.13 client::model::services::DataManagerService

- **Descrizione:** questa classe gestisce l'inoltro di chiamate REST_G al server. Se si tratta di chiamate Get controlla se i dati che vengono richiesti sono già presenti in locale o se devono essere richiesti al server. In particolare, li richiede al server solo se questi sono assenti o se sono l'ultima volta che sono stati chiesti al server è passata da troppo tempo;

- **Utilizzo:** i suoi metodi vengono invocati dalle classi che desiderano fare richieste al server;

- **Relazioni con altre classi:**

- client::model::services::RecipeService
- client::model::services::RecipeAdminService
- client::model::services::UserService
- client::model::services::UserAdminService

- **Attributi:** N/A

- **Metodi:**

- + `getRestCall(restCall)`

Descrizione: metodo che controlla se i risultati di una chiamata rest_G passata per argomento sono presenti in locale, e se non lo sono chiama il server restituendo comunque i dati richiesti.

- + `postRestCall(data)`

Descrizione: metodo per inviare chiamate Rest di tipo post al Server. Prima di aggiornare i dati sul Server aggiorna prima i dati eventualmente presenti in locale.

– + `putRestCall(data)`

Descrizione: metodo per inviare chiamate Rest di tipo put al Server. Prima di aggiornare i dati sul Server aggiorna prima i dati eventualmente presenti in locale.

– + `deleteRestCall(data)`

Descrizione: metodo per inviare chiamate Rest di tipo delete al Server. Prima di aggiornare i dati sul Server aggiorna prima i dati eventualmente presenti in locale.

3.2.5 client::view

In questa sezione e in tutte quelle inerenti al package_G view_G, il termine classe e pagina HTML, saranno sinonimi.

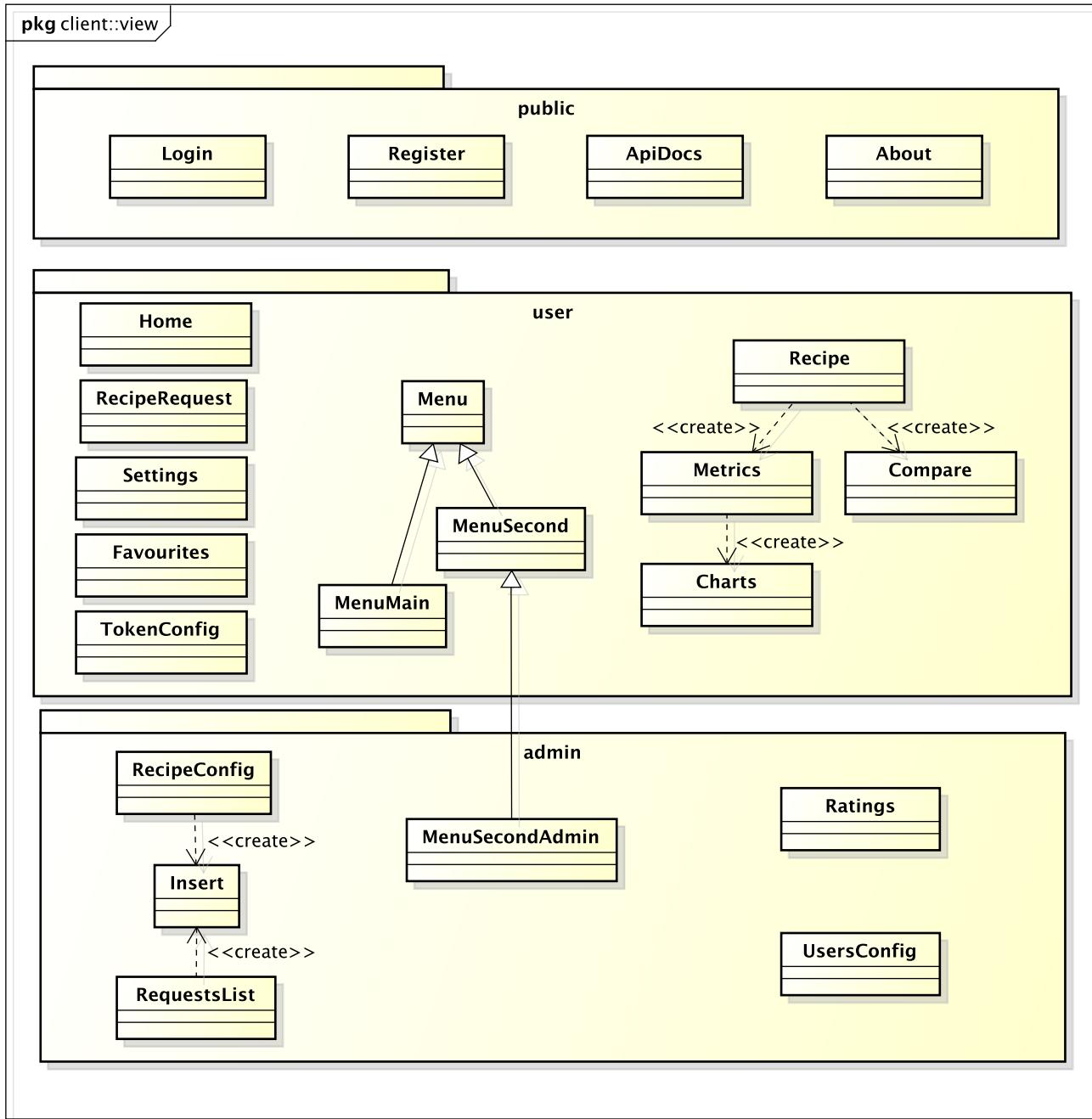


Figura 25: Package - client::view

- **Descrizione:** è il package_G che contiene tutte le classi che costituiscono la view_G del client. Ogni classe è equivalente a un template_G di pagina HTML che sarà presentato all'utente, quando richiesto, tramite un sistema di routing. All'interno di esso sono presenti altri componenti che separano le pagine HTML offerte, a seconda dei permessi che possiede un utente;

- **Padre:** client

- **Package_G contenuti:**

- client::view_G::public
- client::view_G::user
- client::view_G::admin

- **Interazione con altri componenti:**

- client::controller

3.2.6 client::view::public

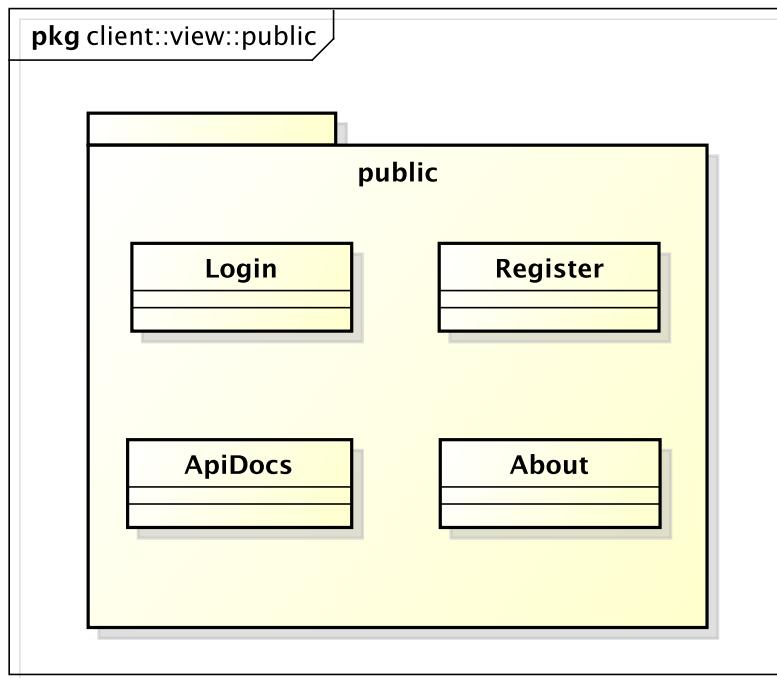


Figura 26: Package - client::view::public

- **Descrizione:** è il package_G che contiene tutte le pagine HTML che possono essere offerte quando l'utente deve ancora effettuare l'accesso al sistema;
- **Padre:** client::view_G
- **Interazione con altri componenti:**
 - client::controller::public

3.2.6.1 Classi

3.2.6.1.1 client::view::public::Login

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione dell'interfaccia di autenticazione;
- **Utilizzo:** viene utilizzata per generare la pagina HTML di autenticazione;
- **Relazioni con altre classi:**
 - client::controller::public::PublicRoute
 - client::controller::public::LoginCtrl
- **Direttive AngularJS_G:**
 - **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di Login;
 - **ng-submit:** rileva una scelta di invio dati dell'utente e attiva una funzione, in questo caso **login(credentials)** definita in LoginCtrl;

- **ng-model**: permette di creare un legame tra le variabili presenti nel controller e i campi di tipo input dell'HTML, realizzando così il two-way data binding.
- **ng-disabled**: permette di bloccare un determinato campo fintanto che una o più condizioni non vengono soddisfatte. Viene utilizzata per bloccare il pulsante di login qualora non fossero stati completati correttamente i campi presenti.

3.2.6.1.2 client::view::public::About

- **Descrizione**: la classe rappresenta un template_G HTML per la visualizzazione della pagina di informazioni sull'applicazione;
- **Utilizzo**: viene usata per generare la pagina HTML di About;
- **Relazioni con altre classi**:
 - client::controller::public::PublicRoute
- **Direttive AngularJS_G**: non sono state utilizzate direttive in quanto è una pagina statica senza bisogno di interazioni con l'utente.

3.2.6.1.3 client::view::public::Register

- **Descrizione**: la classe rappresenta un template_G HTML per la visualizzazione dell'interfaccia di registrazione al servizio;
- **Utilizzo**: viene utilizzata per generare la pagina HTML di registrazione;
- **Relazioni con altre classi**:
 - client::controller::public::PublicRoute
 - client::controller::public::RegisterCtrl
- **Direttive AngularJS_G**:
 - **ng-controller**: permette l'attivazione del controller associato al template_G HTML relativo alla pagina di registrazione;
 - **ng-submit**: rileva una scelta di invio dati dell'utente e attiva una funzione, in questo caso **register(credentials)** definita in RegisterCtrl;
 - **ng-disabled**: permette di bloccare un determinato campo fintanto che una o più condizioni non vengono soddisfatte. Viene utilizzata per bloccare il pulsante di login qualora non fossero stati completati correttamente i campi presenti;
 - **ng-model**: permette di creare un legame tra le variabili presenti nel controller e i campi di tipo input dell'HTML, realizzando così il two-way data binding;
 - **ng-message**: permette di visualizzare un determinato messaggio che viene associato ad un particolare errore scaturito nel form_G di registrazione;
 - **ng-if**: permette di visualizzare un determinato elemento, e il contenuto al suo interno, qualora la condizione specificata risultasse vera. Viene utilizzata per nascondere i messaggi d'errore e renderli visibili qualora l'errore si verificasse;

- **ng-class:** permette di associare una determinata classe, a un elemento del template_G, qualora la condizione specificata risultasse vera. Questa viene utilizzata per mostrare un colore rosso d'errore sui campi che non soddisfano le specifiche richieste dal form_G.

3.2.6.1.4 client::view::public::ApiDocs

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione della pagina contenente la documentazione dei servizi REST_G offerti;
- **Utilizzo:** viene utilizzata per generare la pagina HTML contenente la documentazione dei servizi offerti;
- **Relazioni con altre classi:**
 - client::controller::public::PublicRoute
 - client::controller::public::ApiDocsCtrl
- **Direttive AngularJS_G:**
 - **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione delle API_G disponibili a un utente che ha ottenuto un token di accesso per usarle;
 - **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le API_G e la loro descrizione;
 - **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{} {}`. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**.

3.2.7 client::view::user

- **Descrizione:** è il package_G che contiene tutte le pagine HTML che ha a disposizione l'utente che ha effettuato l'accesso al sistema, sia che sia un utente normale sia che sia un amministratore.
Per semplicità e chiarezza non è stato evidenziato nel grafico il fatto che ogni classe di questo package_G possiede un'istanza di MenuMain e di MenuSecond.
- **Padre:** client::view_G
- **Interazione con altri componenti:**
 - client::controller::user

3.2.7.1 Classi

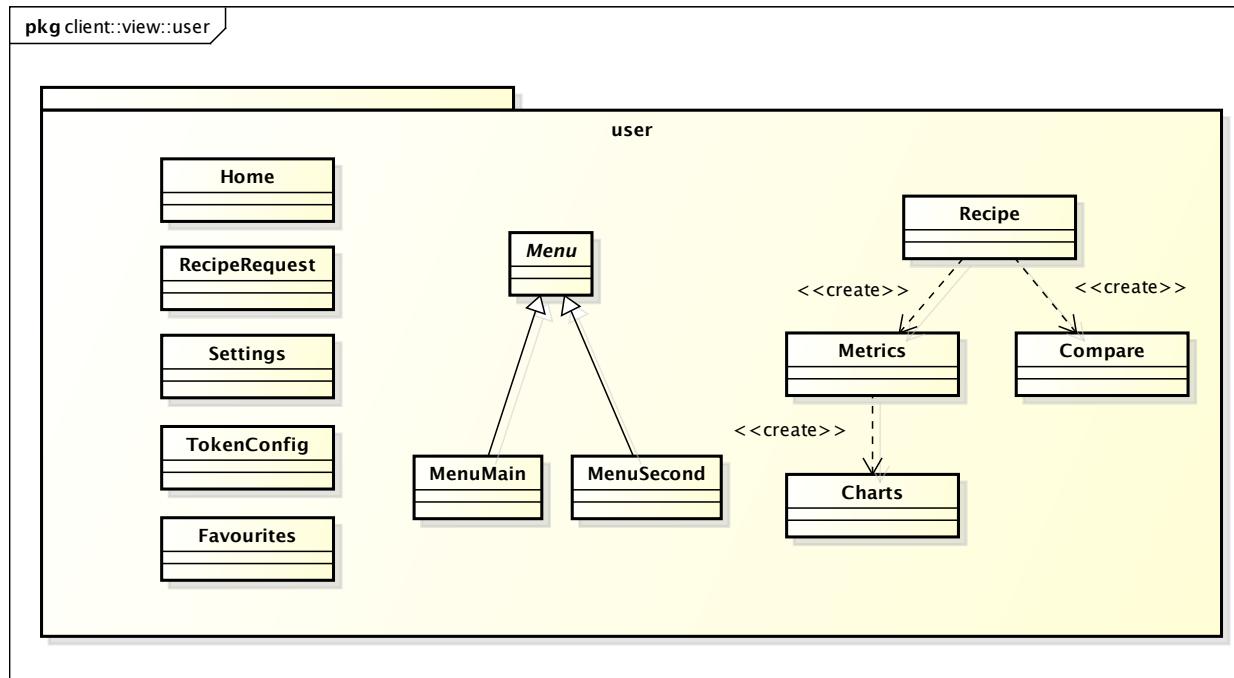


Figura 27: Package - client::view::user

3.2.7.1.1 client::view::user::Home

- **Descrizione:** la classe rappresenta un template_G HTML contenente lo scheletro delle pagine destinate a uno User e a un Admin;
- **Utilizzo:** viene utilizzata per generare lo scheletro della pagina HTML utilizzata come base da altre pagine;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::user::MenuSecond
 - client::controller::public::HomeRoute
- **Direttive AngularJS_G:**
 - **ui-view_G**: direttiva utilizzata dal modulo **ui-router** che permette il caricamento del template_G HTML corretto a seconda di un determinato url o di uno stato attivato tramite la direttiva **ui-sref**.

3.2.7.1.2 client::view::user::Recipe

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione della pagina di visione delle recipe_G a disposizione;
- **Utilizzo:** viene usata per generare la pagina HTML di visione delle recipe_G;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::user::MenuSecond

- client::view_G::user::Metrics
- client::view_G::user::Compare
- client::controller::user::RecipeRoute
- client::controller::user::RecipeCtrl

- **Direttive AngularJS_G:**

- **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione delle recipe_G presenti nel sistema;
- **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le recipe_G;
- **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore {{ }}. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**.
- **ui-sref:** permette di effettuare un reindirizzamento in una determinata pagina, con la possibilità di passare dei parametri. Viene utilizzata dai pulsanti **Show metrics** e **Compare metrics** per andare alle pagine Metrics e Compare a seconda delle necessità dell'utente.

3.2.7.1.3 client::view::user::Metrics

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione della pagina di visione delle metriche a disposizione per una selezionata recipe_G;
- **Utilizzo:** viene utilizzata per generare la pagina HTML di visione delle metriche quando viene selezionata una Recipe_G;

- **Relazioni con altre classi:**

- client::view_G::user::MenuMain
- client::view_G::user::MenuSecond
- client::view_G::user::Recipe_G
- client::view_G::user::Charts
- client::controller::user::RecipeRoute
- client::controller::user::MetricsCtrl

- **Direttive AngularJS_G:**

- **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione delle metriche di una determinata recipe_G;
- **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le metriche relative a una determinata recipe_G;

- **ng-bind**: permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{} {}`. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**.
- **ui-sref**: permette di effettuare un reindirizzamento in una determinata pagina, con la possibilità di passare dei parametri. Viene utilizzata dai pulsanti **Show charts** per andare alla pagina che visualizza tutti i grafici associati a una determinata metrica_G.

3.2.7.1.4 client::view::user::Charts

- **Descrizione**: la classe rappresenta un template_G HTML per la visualizzazione della pagina di visione delle view_G associate ad una metrica_G selezionata;
- **Utilizzo**: viene usata per generare la pagina HTML di visione delle view_G quando viene selezionata una metrica_G;
- **Relazioni con altre classi**:
 - client::view_G::user::MenuMain
 - client::view_G::user::MenuSecond
 - client::view_G::user::Metrics
 - client::controller::user::RecipeRoute
 - client::controller::user::ChartsCtrl
- **Direttive AngularJS_G**:
 - **ng-controller**: permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione di grafici generati da una metrica_G o da confronto;
 - **ng-repeat**: permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per visualizzare tutti i grafici generati;
 - **ng-bind**: permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{} {}`. Viene utilizzata per stampare la descrizione dei grafici che si stanno iterando tramite la direttiva **ng-repeat**.
 - **ng-bind-html**: permette di rimpiazzare il contenuto di un elemento con del codice HTML che non viene interpretato come stringa ma come codice della pagina. Viene usato per il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{} {}`. Viene utilizzata per visualizzare i grafici che si stanno iterando tramite la direttiva **ng-repeat**.

3.2.7.1.5 client::view::user::Compare

- **Descrizione**: la classe rappresenta un template_G HTML per la visualizzazione dell'interfaccia di confronto tra diverse metriche di una recipe_G;

- **Utilizzo:** viene utilizzata per generare la pagina HTML di confronto tra metriche;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::user::MenuSecond
 - client::view_G::user::Recipe_G
 - client::controller::user::RecipeRoute
 - client::controller::user::CompareCtrl
- **Direttive AngularJS_G:**
 - **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di che permette di confronta più metriche di una stessa Recipe_G;
 - **ng-submit:** permette di rilevare una scelta di invio dati dell'utente e attiva una funzione, in questo caso **generate()** definita in CompareCtrl;
 - **ng-click:** permette di rilevare una scelta dell'utente e attiva una funzione, in questo caso **addMetric()** definita in CompareCtrl;
 - **ng-change:** permette di rilevare un cambiamento su un determinato elemento di una form_G, in questo caso aggiorna i valori di alcuni array a seconda del tipo di cambiamento che viene effettuato;
 - **ng-if:** permette di visualizzare un determinato elemento, e il contenuto al suo interno, qualora la condizione specificata risultasse vera. Viene utilizzata per nascondere i messaggi d'errore e renderli visibili qualora l'errore si verificasse;
 - **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le metriche aggiunte alla richiesta di confronto;
 - **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{}{}`. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**.

3.2.7.1.6 client::view::user::RecipeRequest

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione dell'interfaccia di richiesta di una nuova recipe_G;
- **Utilizzo:** viene utilizzata per generare la pagina HTML di richiesta di una nuova recipe_G;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::user::MenuSecond
 - client::controller::user::RecipeRequestRoute
 - client::controller::user::RecipeRequestCtrl

- **Direttive AngularJS_G:**

- **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione del form_G di richiesta di inserimento di una nuova recipe_G;
- **ng-submit:** permette di rilevare una scelta di invio dati dell'utente e attiva una funzione, in questo caso `insertRecipeRequest()` definita in RecipeRequestCtrl;
- **ng-click:** permette di rilevare una scelta dell'utente e attiva una funzione, in questo caso `addMetric()` definita in RecipeRequestCtrl.
- **ng-disabled:** permette di bloccare un determinato campo fintanto che una o più condizioni non vengono soddisfatte. Viene utilizzata per bloccare il pulsante di inserimento della richiesta qualora non fossero stati completati correttamente i campi presenti;
- **ng-model:** permette di creare un legame tra le variabili presenti nel controller e i campi di tipo input dell'HTML, realizzando così il two-way data binding;
- **ng-if:** permette di visualizzare un determinato elemento, e il contenuto al suo interno, qualora la condizione specificata risultasse vera. Viene utilizzata per nascondere i messaggi d'errore e renderli visibili qualora l'errore si verificasse;
- **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le metriche aggiunte alla richiesta di inserimento di una recipe_G;
- **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{}{}`. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**.

3.2.7.1.7 client::view::user::Favourites

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione della pagina contenente le view_G preferite;
- **Utilizzo:** viene utilizzata per generare la pagina HTML delle view_G preferite;
- **Relazioni con altre classi:**

- client::view_G::user::MenuMain
- client::view_G::user::MenuSecond
- client::controller::user::FavouritesRoute
- client::controller::user::FavouritesCtrl

- **Direttive AngularJS_G:**

- **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione dei grafici preferiti;

- **ng-repeat**: permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per visualizzare tutti i grafici favoriti;
- **ng-bind**: permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{{ }}`. Viene utilizzata per stampare la descrizione dei grafici che si stanno iterando tramite la direttiva **ng-repeat**.
- **ng-bind-html**: permette di rimpiazzare il contenuto di un elemento con del codice HTML che non viene interpretato come stringa ma come codice della pagina. Viene usato per il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{{ }}`. Viene utilizzata per visualizzare i grafici che si stanno iterando tramite la direttiva **ng-repeat**.

3.2.7.1.8 client::view::user::TokenConfig

- **Descrizione**: la classe rappresenta un template_G HTML per la visualizzazione dell'interfaccia di gestione dei token per l'utilizzo dei servizi REST_G;
- **Utilizzo**: viene utilizzata per generare la pagina HTML di gestione dei token per i servizi REST_G;
- **Relazioni con altre classi**:
 - client::view_G::user::MenuMain
 - client::view_G::user::MenuSecond
 - client::controller::user::TokenConfigRoute
 - client::controller::user::TokenConfigCtrl
- **Direttive AngularJS_G**:
 - **ng-controller**: permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione del form_G di richiesta/cancellazione di un token di accesso ai servizi REST_G;
 - **ng-if**: permette di visualizzare un determinato elemento, e il contenuto al suo interno, qualora la condizione specificata risultasse vera. Viene utilizzata per alternare i pulsanti di richiesta/cancellazione del token di accesso, a seconda se esso sia già stato generato o meno;
 - **ng-bind**: permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{{ }}`. Viene utilizzata per stampare il valore del token generato.

3.2.7.1.9 client::view::user::Settings

- **Descrizione**: la classe rappresenta un template_G HTML per la visualizzazione dell'interfaccia di gestione delle opzioni;
- **Utilizzo**: viene utilizzata per generare la pagina HTML delle opzioni;
- **Relazioni con altre classi**:

- client::view_G::user::MenuMain
- client::view_G::user::MenuSecond
- client::controller::user::SettingsRoute
- client::controller::user::SettingsCtrl

- **Direttive AngularJS_G:**

- **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione del form_G delle impostazioni personali del profilo utente;
- **ng-init:** permette di inizializzare una variabile definita all'interno del template_G con un determinato valore. Viene utilizzata per impostare le variabili **editUserName** e **editMail** inizialmente a **true**;
- **ng-model:** permette di creare un legame tra le variabili presenti nel controller e i campi di tipo input dell'HTML, realizzando così il two-way data binding;
- **ng-submit:** permette di rilevare una scelta di invio dati dell'utente e attiva una funzione, in questo caso **saveEdit()** definita in SettingsCtrl;
- **ng-click:** permette di rilevare una scelta dell'utente e attiva una funzione o cambia il valore di qualche variabile dichiarata all'interno del template_G, in questo caso cambia la variabile **editUserName** o **editMail** nel valore **editUserName** o **editMail**;
- **ng-disabled:** permette di bloccare un determinato campo fintanto che una o più condizioni non vengono soddisfatte. Viene utilizzata per bloccare il pulsante di salvataggio delle modifiche o per bloccare i campi di modifica della password e della mail, attivabile solo premendo su un apposito pulsante.

3.2.7.1.10 client::view::user::Menu

- **Descrizione:** questa classe astratta rappresenta un template_G HTML per la creazione di un menù;
- **Utilizzo:** non viene mai istanziata nel sistema, viene ereditata dalla classi che istanziano dei menù;
- **Relazioni con altre classi:**

- client::view_G::user::MenuMain
- client::view_G::user::MenuSecond
- client::controller::user::MenuCtrl

3.2.7.1.11 client::view::user::MenuMain

- **Descrizione:** la classe rappresenta un template_G HTML per la creazione del menù principale;
- **Utilizzo:** viene usata nel generare ogni pagina usata da un utente autenticato per generare il menù principale in essa contenuta;
- **Classi ereditate:**

- client::view_G::user::Menu

- **Relazioni con altre classi:**

- client::view_G::user::Menu
- client::controller::user::MenuMainCtrl

- **Direttive AngularJS_G:**

- **ui-sref:** permette di effettuare un reindirizzamento in una determinata pagina, con la possibilità di passare dei parametri. Viene utilizzata dai pulsanti presenti nel menu principale per navigare in alcune delle pagine del sistema;

3.2.7.1.12 client::view::user::MenuSecond

- **Descrizione:** la classe rappresenta un template_G HTML per la creazione del menù secondario di un utente non amministratore;

- **Utilizzo:** viene usata nel generare ogni pagina di un utente autenticato non amministratore per generare il menù secondario;

- **Classi ereditate:**

- client::view_G::user::Menu

- **Relazioni con altre classi:**

- client::view_G::user::Menu
- client::view_G::admin::MenuSecondAdmin
- client::controller::user::MenuSecondCtrl

- **Direttive AngularJS_G:**

- **ui-sref:** permette di effettuare un reindirizzamento in una determinata pagina, con la possibilità di passare dei parametri. Viene utilizzata dai pulsanti presenti nel menu secondario per navigare in alcune delle pagine del sistema;

3.2.8 client::view::admin

- **Descrizione:** è il package_G che contiene tutte le pagine HTML che ha a disposizione l'utente che ha i privilegi di amministratore;

Per semplicità e chiarezza non è stato evidenziato nel grafico il fatto che ogni classe di questo package_G possiede un'istanza di MenuMain e di MenuSecondAdmin.

- **Padre:** client::view_G

- **Interazione con altri componenti:**

- client::controller::admin

3.2.8.1 Classi

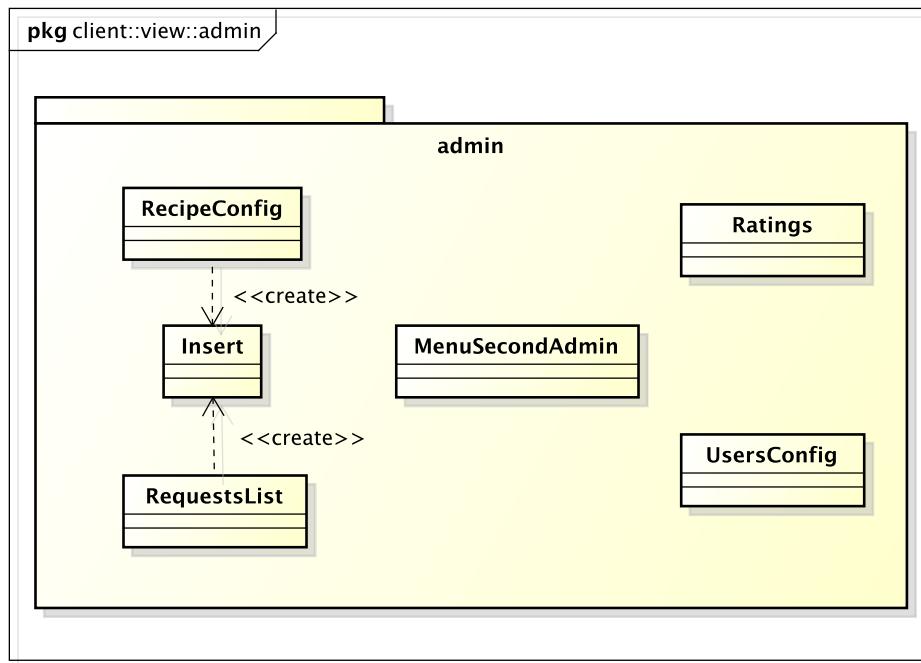


Figura 28: Package - client::view::admin

3.2.8.1.1 client::view::admin::RecipeConfig

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione dell’interfaccia di gestione delle Recipe_G;
- **Utilizzo:** viene utilizzata per generare la pagina HTML della gestione delle Recipe_G;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::admin::MenuSecondAdmin
 - client::view_G::admin::Insert
 - client::controller::admin::RecipeConfigRoute
 - client::controller::admin::RecipeConfigCtrl
- **Direttive AngularJS_G:**
 - **ng-controller:** permette l’attivazione del controller associato al template_G HTML relativo alla pagina di inserimento di nuove recipe_G;

3.2.8.1.2 client::view::admin::RequestList

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione della pagina di visione delle richieste di Recipe_G;
- **Utilizzo:** viene utilizzata per generare la pagina HTML della lista delle richieste di Recipe_G;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain

- client::view_G::admin::MenuSecondAdmin
- client::view_G::admin::Insert
- client::controller::admin::RequestListRoute
- client::controller::admin::RequestListCtrl

- **Direttive AngularJS_G:**

- **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione delle richieste di nuove recipe_G;
- **ng-click:** permette di rilevare una scelta dell'utente e attiva una funzione o cambia il valore di qualche variabile dichiarata all'interno del template_G, in questo viene utilizzato da due pulsanti che attivano le relative funzioni **discardRequest(idRequestRecipe)** e **approvedRequest(idRequestRecipe)**;
- **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all'interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le richieste di nuove recipe_G;
- **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall'espressione definita. Questa direttiva viene preferita all'uso dell'operatore `{} {}`. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**;
- **ui-sref:** permette di effettuare un reindirizzamento in una determinata pagina, con la possibilità di passare dei parametri. Viene utilizzata dal pulsante **See Details** che manda alla visualizzazione dei dettagli di una richiesta particolare.

3.2.8.1.3 client::view::admin::Insert

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione dell'interfaccia di inserimento di una nuova Recipe_G;
- **Utilizzo:** viene utilizzata per generare la pagina HTML di inserimento di una nuova recipe_G;

- **Relazioni con altre classi:**

- client::view_G::user::MenuMain
- client::view_G::admin::MenuSecondAdmin
- client::view_G::admin::RecipeConfig
- client::view_G::admin::RequestList
- client::controller::admin::RecipeConfigRoute
- client::controller::admin::InsertRecipeCtrl

- **Direttive AngularJS_G:**

- **ng-controller:** permette l'attivazione del controller associato al template_G HTML relativo al form_G di inserimento di nuova recipe_G;
- **ng-submit:** rileva una scelta di invio dati dell'utente e attiva una funzione, in questo caso **insertRecipe()** definita in RecipeCtrl;

- **ng-click:** permette di rilevare una scelta dell’utente e attiva una funzione o cambia il valore di qualche variabile dichiarata all’interno del template_G, in questo viene utilizzato dal pulsante che aggiunge un metrica_G tramite la funzione **addMetric(cat, typeCat, value);**
- **ng-show:** permette di visualizzare o meno del contenuti HTML a seconda se una determina condizione sia verificata. Viene utilizzato per nascondere/visualizzare gli errori di inserimento dei campi dati o qualche informazione di successo delle operazioni previste;
- **ng-model:** permette di creare un legame tra le variabili presenti nel controller e i campi di tipo input dell’HTML, realizzando così il two-way data binding;
- **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all’interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le metriche che si vogliono aggiungere alla Recipe_G;
- **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall’espressione definita. Questa direttiva viene preferita all’uso dell’operatore {{ }}. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**;

3.2.8.1.4 client::view::admin::Ratings

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione della pagina dei voti assegnati alle varie Recipe_G;
- **Utilizzo:** viene utilizzata per generare la pagina HTML di visione dei voti delle Recipe_G;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::admin::MenuSecondAdmin
 - client::controller::admin::RatingsRoute
 - client::controller::admin::RatingsCtrl
- **Direttive AngularJS_G:**
 - **ng-controller:** permette l’attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione delle valutazione delle recipe_G;
 - **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all’interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutte le recipe_G con affianco la loro valutazione;
 - **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall’espressione definita. Questa direttiva viene preferita all’uso dell’operatore {{ }}. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**;

3.2.8.1.5 client::view::admin::UsersConfig

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione dell’interfaccia di gestione degli utenti;
- **Utilizzo:** viene utilizzata per generare la pagina HTML di gestione degli utenti;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::admin::MenuSecondAdmin
 - client::controller::admin::UsersConfigRoute
 - client::controller::admin::UsersConfigCtrl
- **Direttive AngularJS_G:**
 - **ng-controller:** permette l’attivazione del controller associato al template_G HTML relativo alla pagina di visualizzazione della lista degli utenti presenti nel sistema;
 - **ng-click:** permette di rilevare una scelta dell’utente e attiva una funzione o cambia il valore di qualche variabile dichiarata all’interno del template_G, in questo viene utilizzato dal pulsante che elimina un utente tramite la funzione **deleteAccount(idUser)**;
 - **ng-repeat:** permette di iterare ogni elemento di una lista e di poterlo utilizzare all’interno del template_G HTML grazie alla direttiva **ng-bind** potendone ricavare dati o eseguire metodi su di esso. Viene utilizzata per stampare la lista di tutti gli utenti del sistema;
 - **ng-bind:** permette di rimpiazzare il contenuto di un elemento con il valore dato dall’espressione definita. Questa direttiva viene preferita all’uso dell’operatore `{}{}`. Viene utilizzata per stampare il valore degli elementi che si stanno iterando tramite la direttiva **ng-repeat**;

3.2.8.1.6 client::view::admin::MenuSecondAdmin

- **Descrizione:** la classe rappresenta un template_G HTML per la creazione del menù secondario di un utente amministratore;
- **Utilizzo:** viene usata nel generare ogni pagina di un utente autenticato amministratore per generare il menù secondario;
- **Classi ereditate:**
 - client::view_G::user::Menu
 - client::view_G::user::MenuSecond
- **Relazioni con altre classi:**
 - client::view_G::user::Menu
 - client::view_G::user::MenuSecond
 - client::controller::admin::MenuSecondAdminCtrl
- **Direttive AngularJS_G:**

- **ui-sref:** permette di effettuare un reindirizzamento in una determinata pagina, con la possibilità di passare dei parametri. Viene utilizzata dai pulsanti presenti nel menu secondario per navigare in alcune delle pagine del sistema riservate all'amministratore;

3.2.9 client::controller

Le classi definite per la gestione dell’indirizzamento delle pagine HTML dell’applicazione, e cioè quelle terminanti con **Route**, rappresentano le configurazioni necessarie al sistema per indirizzare l’utente nella pagina corretta a seconda di che voce seleziona nei menu messi a disposizione e associare a quella pagina il controller necessario alla sua gestione.

Questo implica che non possiedono metodi o attributi, ma al massimo utilizzano servizi per attuare il loro compito che vengono descritti nella nota introduttiva in 3.2. Tutti questi moduli di routing vengono iniettati nell’applicazione tramite il meccanismo di Dependency Injection offerto dal framework che avvia il meccanismo appena viene caricato il sito.

La zona quindi dedicata a quei valori sarà marcata come specificato in 3.1 e non verrà fornita nessuna immagine, la codifica dovrà quindi seguire le linee guida e i template descritti nel documento *Norme di Progetto v6.0.0*.

- **Descrizione:** è il package_G che contiene le componenti che contengono i controller dell’applicazione e che incapsulano le funzionalità di two-way data binding tra le View_G e il Model. In esse è contenuta la logica applicativa riguardante le diverse pagine HTML;
- **Padre:** client;
- **Package_G contenuti:**
 - client::controller::public
 - client::controller::user
 - client::controller::admin
- **Interazione con altri componenti:**
 - client::model
 - client::view_G

3.2.10 client::controller::public

- **Descrizione:** è il package_G che contiene le classi che controllano le decisioni di che pagine HTML mostrare all’utente che non si è ancora autenticato e di quali operazioni poter effettuare su di esse;
- **Padre:** client::controller
- **Interazione con altri componenti:**
 - client::model::public
 - client::view_G::public

3.2.10.1 Classi

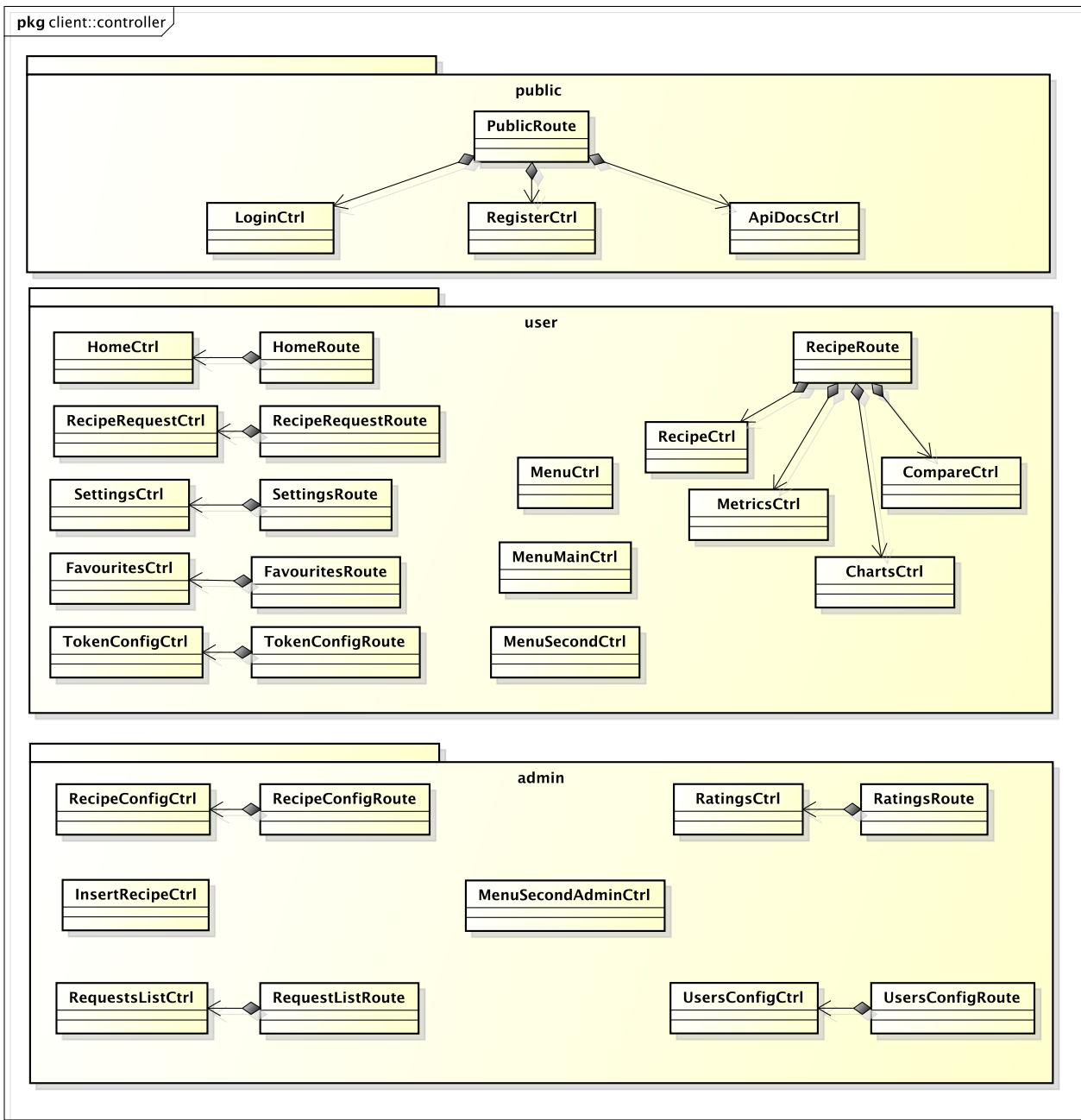


Figura 29: Package - `client::controller`

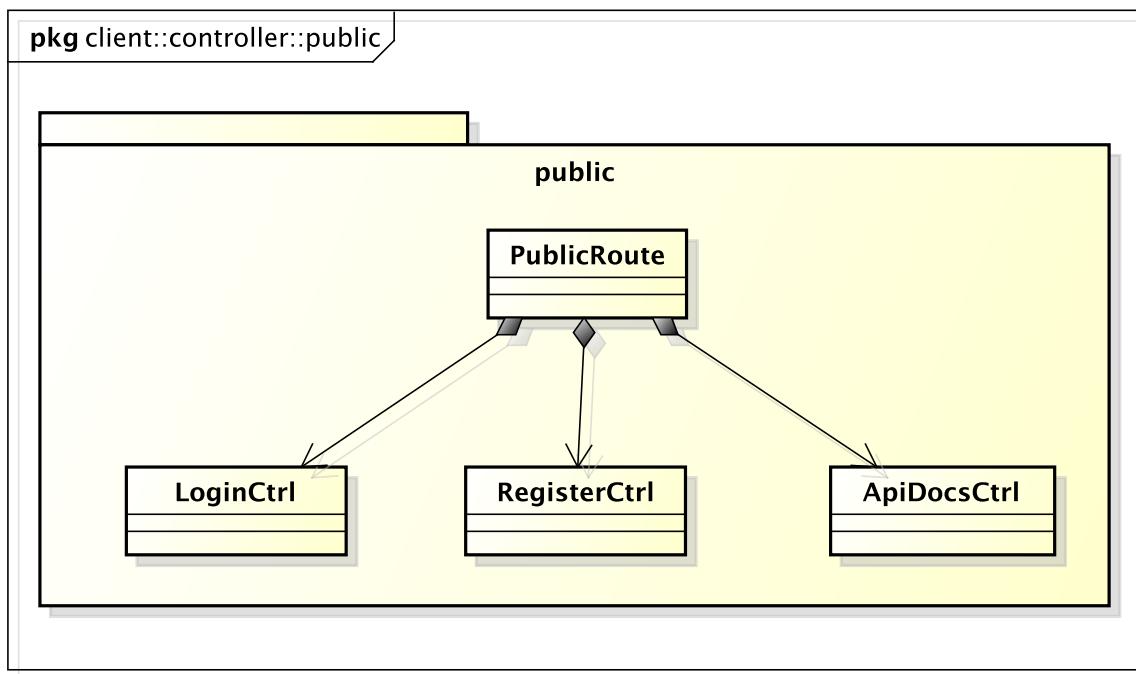


Figura 30: Package - client::controller::public

3.2.10.1.1 client::controller::public::PublicRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante le pagine che può visualizzare un utente non autenticato, quali quella di autenticazione, di registrazione o di visione della documentazione dei servizi REST_G offerti;
- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML adeguato in caso si voglia o autenticare o registrare al sistema, o visualizzare i servizi REST_G offerti, ed assegnare al template scelto il controller opportuno per gestire i diversi compiti. Nel caso l'utente fosse già autenticato, la classe provvederà a reindirizzarlo all'URL associato alla Home;
- **Relazioni con altre classi:**
 - client::view_G::public::Login
 - client::view_G::public::Register
 - client::view_G::public::ApiDocs
 - client::controller::public::LoginCtrl
 - client::controller::public::RegisterCtrl
 - client::controller::public::ApiDocsCtrl
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

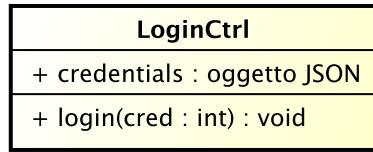


Figura 31: Classe - client::controller::public::LoginCtrl

3.2.10.1.2 client::controller::public::LoginCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante l'autenticazione al sistema;
- **Utilizzo:** viene utilizzata per gestire le operazioni di autenticazione al sistema qualora un utente non autenticato decidesse di effettuare l'accesso. Richiamando quindi i servizi che mettono in comunicazione il client con il server, verifica le credenziali ed effettua un reindirizzamento alla Home qualora fossero valide;
- **Relazioni con altre classi:**
 - client::model::services::AuthService
 - client::view_G::public::Login
 - client::controller::public::PublicRoute
- **Attributi associati allo \$scope:**
 - + `credentials` : oggetto JSON_G

Descrizione: oggetto nel formato JSON_G che contiene il valore iniziale dei campi mail e password che inizialmente saranno vuoti. Il formato dell'oggetto JSON è il seguente:

```
{
  email: 'info@mashup-unipd.it',
  pwd: 'password'
}
```

- **Metodi associati allo \$scope e privati:**

- + `login(cred)`

Descrizione: gli vengono passati i dati dalla view_G e viene invocato il metodo login della classe AuthService.

- **Servizi AngularJS_G utilizzati:** N/A

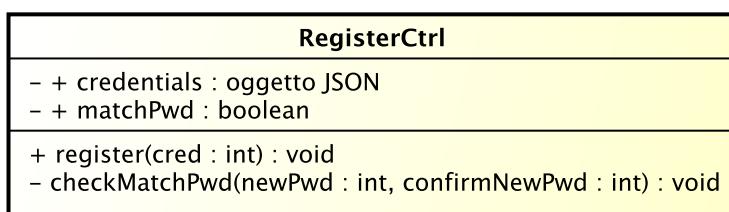


Figura 32: Classe - client::controller::public::RegisterCtrl

3.2.10.1.3 client::controller::public::RegisterCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la registrazione al sistema;
- **Utilizzo:** viene utilizzata per gestire le operazioni di registrazione al sistema qualora un utente che non si è ancora registrato decidesse di effettuare l'accesso. Richiamando quindi i servizi che mettono in comunicazione il client con il server, verifica le credenziali ed effettua un reindirizzamento alla pagina di Login qualora la registrazione fosse avvenuta con successo;
- **Relazioni con altre classi:**
 - client::model::services::AuthService
 - client::model::data::UserMode
 - client::view_G::public::Register
 - client::controller::public::PublicRoute
- **Attributi associati allo \$scope:**
 - + `credentials` : oggetto JSON_G

Descrizione: oggetto nel formato JSON_G che contiene il valore iniziale dei campi username, mail, password e password da confermare che inizialmente saranno vuoti. Il formato dell'oggetto JSON è il seguente:

```

{
    username: 'MashUp',
    email: 'info@mashup.unipd.it',
    pwd: 'gruppo',
    confirmPwd: 'gruppo'
}
  
```

 - + `matchPwd` : bool

Descrizione: booleano inizializzato a false che cambia valore se il valore della nuova password e quello di conferma non combaciano.
- **Metodi associati allo \$scope e privati:**
 - + `register(cred)`

Descrizione: gli vengono passati i dati dalla view_G e viene invocato il metodo register della classe AuthService.

 - - `checkMatchPwd(newPwd, confirmNewPwd)`

Descrizione: gli vengono passati i dati dalla view_G e viene invocato il metodo register della classe AuthService.
- **Servizi AngularJS_G utilizzati:** N/A

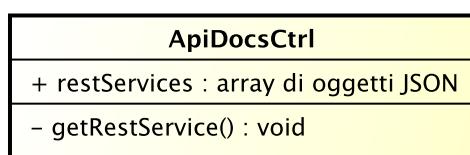


Figura 33: Classe - client::controller::public::ApiDocsCtrl

3.2.10.1.4 client::controller::public::ApiDocsCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina di visualizzazione dei servizi REST_G che il sistema offre;
- **Utilizzo:** viene utilizzata per gestire in maniera automatica la documentazione riguardante i servizi REST_G offerti dal sistema;
- **Relazioni con altre classi:**
 - client::model::ApiDocsModel
 - client::view_G::public::ApiDocs
 - client::controller::public::PublicRoute
- **Attributi associati allo \$scope:**
 - + `restServices : array di oggetti JSONG`

Descrizione: array di oggetti nel formato JSON_G che rappresentano la lista delle API_G pubbliche. Questa variabile viene inizializzata tramite la chiamata immediata alla funzione privata `getRestService`.
- **Metodi associati allo \$scope e privati:**
 - - `getRestService()`

Descrizione: metodo che preleva la lista delle API_G pubbliche tramite una chiamata alla classe `ApiDocsModel`.
- **Servizi AngularJS_G utilizzati:** N/A

3.2.11 client::controller::user

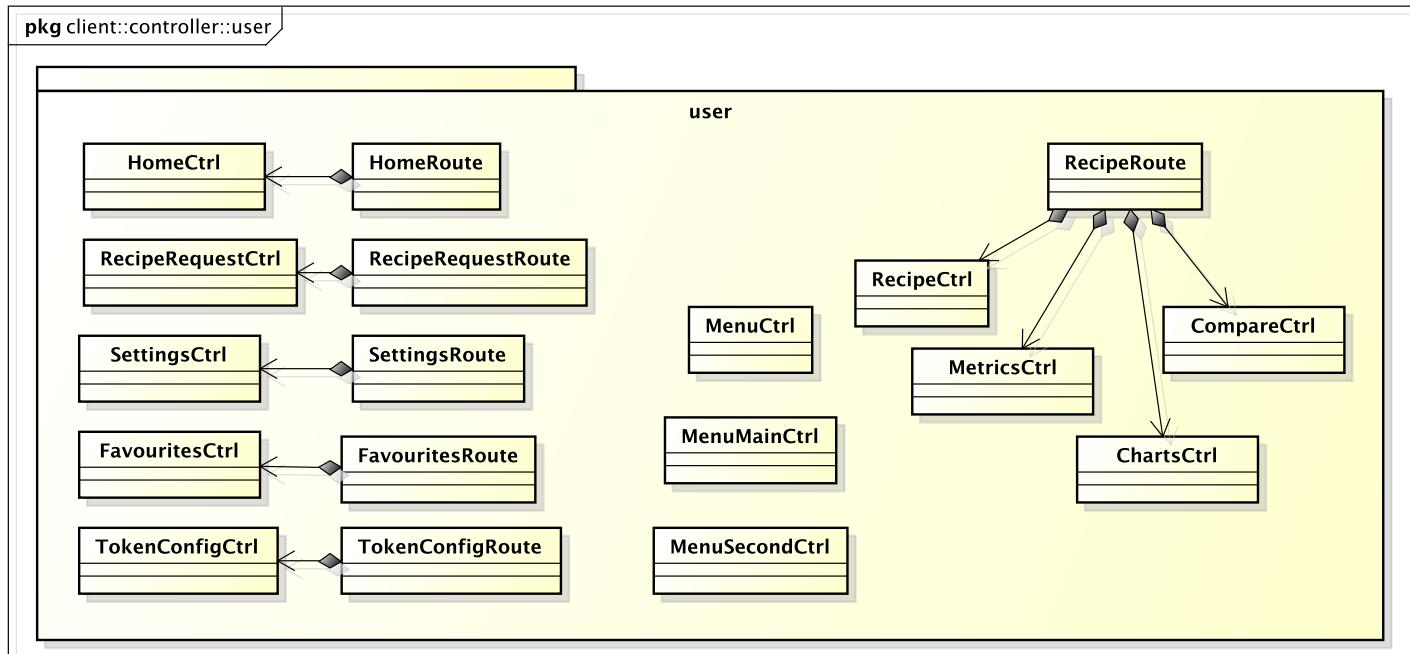


Figura 34: Package - client::controller::user

- **Descrizione:** è il package_G che contiene le classi che controllano le decisioni di che pagine HTML mostrare all’utente autenticato e di quali operazioni poter effettuare su di esse;
- **Padre:** client::controller
- **Interazione con altri componenti:**
 - client::model::user
 - client::view_G::user

3.2.11.1 Classi

3.2.11.1.1 client::controller::user::MenuCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la parte in comune tra i diversi menu presenti nell’applicazione
- **Utilizzo:** viene utilizzata per gestire le operazioni in comune e gli stati tra i diversi menu effettivi che conterranno i vari collegamenti alle diverse pagine;
- **Relazioni con altre classi:**
 - client::view_G::user::Menu
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A

3.2.11.1.2 client::controller::user::MenuMainCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante il menu principale dell’applicazione;
- **Utilizzo:** viene utilizzata per gestire i collegamenti principali ad alcune pagine del sistema ed ad effettuare l’operazione di logout;
- **Classi ereditate:**
 - client::controller::user::MenuCtrl. Questa ereditarietà è solo logica e non viene implementata in nessuna forma.
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::controller::user::LogoutCtrl
 - client::controller::public::HomeRoute
 - client::controller::user::SettingsRoute
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:** N/A

3.2.11.1.3 client::controller::user::MenuSecondCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante il menu secondario dell'applicazione;
- **Utilizzo:** viene utilizzata per gestire i collegamenti secondari alla maggior parte delle pagine del sistema;
- **Classi ereditate:**
 - client::controller::user::MenuCtrl. Questa ereditarietà è solo logica e non viene implementata in nessuna forma.
- **Relazioni con altre classi:**
 - client::view_G::user::MenuSecond
 - client::controller::user::RecipeRoute
 - client::controller::user::RecipeRequestRoute
 - client::controller::user::FavouritesRoute
 - client::controller::user::TokenConfigRoute
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:** N/A

3.2.11.1.4 client::controller::user::HomeRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la Home page dell'utente che ha effettuato l'accesso al sistema;
- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina iniziale dell'applicativo una volta effettuato l'accesso ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;
- **Relazioni con altre classi:**
 - client::view_G::user::Home
 - client::controller::user::HomeCtrl
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

3.2.11.1.5 client::controller::user::HomeCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina principale dell'applicazione, una volta che l'utente ha effettuato l'accesso;
- **Utilizzo:** viene utilizzata per gestire la pagina principale dell'applicazione. Su di essa per il momento non sono state previste particolari operazioni, ma viene lo stesso inserito un controller per scopi futuri;
- **Relazioni con altre classi:**
 - client::view_G::user::Home
 - client::controller::user::HomeRoute
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:** N/A

3.2.11.1.6 client::controller::user::RecipeRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina delle Recipe_G e di quelle che vengono create da essa, quali **Metrics**, **Charts** e **Compare**;
- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina relativa alle Recipe_G presenti nel sistema ed assegnare al template scelto il controller opportuno per gestire i diversi compiti. Da lì l'utente potrà muoversi a seconda delle sue esigenze nelle pagine citate nella *Descrizione*;
- **Relazioni con altre classi:**
 - client::view_G::user::Recipe_G
 - client::view_G::user::Metrics
 - client::view_G::user::Charts
 - client::view_G::user::Compare
 - client::controller::user::RecipeCtrl
 - client::controller::user::MetricsCtrl
 - client::controller::user::ChartsCtrl
 - client::controller::user::CompareCtrl
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

RecipeCtrl
+ listRecipes : array di oggetti JSON
- getListOfRecipes() : void

Figura 35: Classe - client::controller::user::RecipeCtrl

3.2.11.1.7 client::controller::user::RecipeCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante le Recipe_G presenti nel sistema;

- **Utilizzo:** viene utilizzata per gestire la lista delle Recipe_G disponibili che verranno mostrate all'utente e le operazioni che esso ha a disposizione su queste. Queste operazioni saranno sotto forma di pulsanti associati a ciascuna metrica_G che porteranno l'utente o a visualizzare tutte le metriche inerenti alla Recipe scelta o ad effettuare un confronto tra le diverse metriche presenti;

- **Relazioni con altre classi:**

- client::model::services::RecipeService
- client::view_G::user::Recipe_G
- client::controller::view_G::RecipeRoute

- **Attributi associati allo \$scope:**

- + listRecipes : array di oggetti JSON_G

Descrizione: array di oggetti nel formato JSON_G che rappresentano la lista delle recipe_G presenti nel sistema. Questa variabile viene inizializzata tramite la chiamata immediata al metodo privato getListOfRecipes. Il formato degli oggetti JSON contenuti nell'array è il seguente:

```
{
    id: '42',
    title: 'Titolo della Recipe\gloss{}',
    desc: 'Descrizione della Recipe\gloss{}',
}
```

- **Metodi associati allo \$scope e privati:**

- - getListOfRecipes()

Descrizione: metodo che preleva la lista delle recipe_G tramite una chiamata al metodo getRecipesList della classe RecipeService.

- **Servizi AngularJS_G utilizzati:** N/A

3.2.11.1.8 client::controller::user::MetricsCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante le metriche associate ad una Recipe_G precisa;

- **Utilizzo:** viene utilizzata per gestire la lista delle metriche associate a una Recipe_G, fornendo dei pulsanti per ciascuna che serviranno all'utente per andare a visualizzare i grafici inerenti alla metrica_G selezionata;

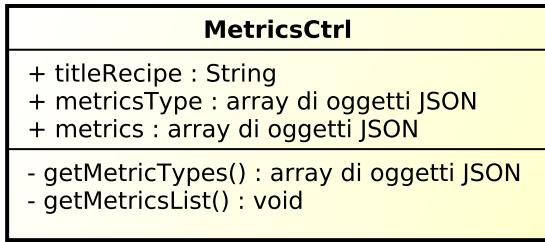


Figura 36: Classe - client::controller::user::MetricsCtrl

- Relazioni con altre classi:

- client::model::services::RecipeService
- client::view_G::user::metrics
- client::controller::user::RecipeRoute

- Attributi associati allo \$scope:

- + titleRecipe : String

Descrizione: parametro che identifica il nome della Recipe_G che contiene le metriche visualizzate dalla pagina. Questo parametro viene inizializzato con il valore preso in input dalla pagina delle Recipe quando si sceglie di visualizzare le metriche per una determinata di esse;

- + metricsType : array di oggetti JSON_G

Descrizione: parametro che contiene la lista delle categorie possibile di metriche. Questo campo viene inizializzato invocando il metodo **getMetricTypes()**;

- + metrics : array di oggetti JSON_G

Descrizione: parametro che contiene la lista delle metriche di una determinata Recipe_G.

- Metodi associati allo \$scope e privati:

- - getMetricTypes() : array di oggetti JSON_G

Descrizione: metodo che recupera il tipo di metriche presenti attraverso la chiamata **getMetricType()** al servizio **RecipeService**. L'array restituito sarà il seguente:

```
[
  {
    key: 'facebook',
    value: 'Facebook'
  },
  {
    key: 'twitter',
    value: 'Twitter'
  },
  {
    key: 'instagram',
    value: 'Instagram'
  }
]
```

– – `getMetricsList() : void`

Descrizione: metodo che viene invocato quando la classe viene attivata.

Questo effettua la chiamata `getMetricsList(idRecipe)` sul servizio **RecipeService** e va a riempire l'attributo **metrics** con un array di oggetti JSON_G del seguente formato:

```
{
  id: 'StarWars.it',
  category: 'facebook',
  category_type: 'page'
}
```

- **Servizi AngularJS_G utilizzati:** N/A

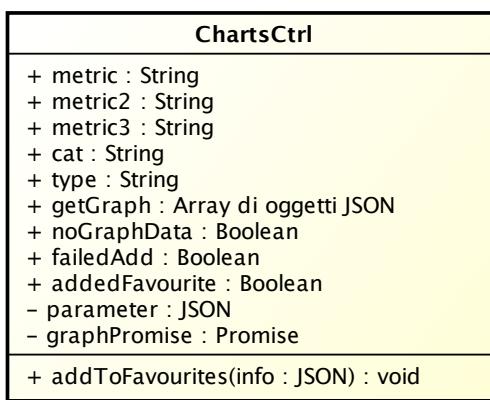


Figura 37: Classe - client::controller::user::ChartsCtrl

3.2.11.1.9 client::controller::user::ChartsCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante i grafici che vengono generati a partire dai dati associati ad una metrica_G precisa;
- **Utilizzo:** viene utilizzata per gestire e generare i grafici a partire dai dati della metrica_G selezionata. La classe richiamerà i diversi metodi presenti nel **model** che a seconda della tipologia della metrica, creeranno i grafici opportuni;

- **Relazioni con altre classi:**

- client::model::services::RecipeService
- client::view_G::user::Charts
- client::controller::user::RecipeRoute

- **Attributi associati allo \$scope:**

– + `metric : String`

Descrizione: parametro che identifica il nome della metrica_G da cui generare i grafici. Se sono presenti altre metriche è una delle metriche da cui generare il confronto.

– + `metric2 : String`

Descrizione: parametro opzionale, se non vuoto identifica una delle metriche con cui effettuare confronto.

– + `metric3 : String`

Descrizione: parametro opzionale, se non vuoto identifica una delle metriche con cui effettuare confronto.

– + `cat : String`

Descrizione: parametro che identifica la categoria_G della/e metrica_G/metriche.

– + `type : String`

Descrizione: parametro che identifica il tipo della/e metrica_G/metriche.

– + `getGraph : Array di oggetti JSONG`

Descrizione: contiene i grafici che devono essere visualizzati in un JSON_G della forma:

```
{
  desc: 'Descrizione del grafico',
  data: codice HTML per generare il grafico
  info: nome della funzione di ViewTypeModel che ha generato il grafico
}
```

– + `noGraphData : Boolean`

Descrizione: indica se non è presente nessun grafico.

– + `addedFavourite : Boolean`

Descrizione: indica se è stato aggiunto un grafico ai favoriti.

– + `failedAdd : Boolean`

Descrizione: indica se è stata fallita un'aggiunta ai favoriti.

– - `parameter : JSONG nella forma di MetricModel`

Descrizione: JSON_G che viene passato al RecipeService per richiedere la generazione di grafici.

– - `graphPromise : Promise`

Descrizione: Promessa dei grafici da visualizzare.

- **Metodi associati allo \$scope e privati:**

– + `addToFavourites(info)`

Descrizione: metodo per chiamare i metodi di UserService per l'aggiunta di un favorito.

- **Servizi AngularJS_G utilizzati:**

– \$stateParams

3.2.11.1.10 client::controller::user::CompareCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la generazione di confronti tra diverse metriche di una Recipe_G;

- **Utilizzo:** viene utilizzata per gestire e generare i grafici a partire dai dati delle metriche selezionate che si vuole confrontare;

- **Relazioni con altre classi:**

CompareCtrl
<pre>+ titleRecipe : String + metricsType : array di oggetti JSON + oneInserted : boolean + maxReached : boolean + categories : array di oggetti JSON + types : array di JSON + type : String + valueMetric : String + metrics : array di oggetti JSON</pre>
<pre>- getMetricTypes() : array di oggetti JSON + addMetric(cat : int, typeCat : int, value : int) : void + updateTypeMetric(category : int) : void + updateMetrics(category : int, type : int) : void</pre>

Figura 38: Classe - client::controller::user::CompareCtrl

- client::model::data::CompareModel
- client::view_G::user::Compare
- client::controller::RecipeRoute

- **Attributi associati allo \$scope:**

- + **titleRecipe** : **String**

Descrizione: parametro che identifica il nome della Recipe_G della quale si vogliono usare le metriche per effettuare un confronto. Questo parametro viene inizializzato con il valore preso in input dalla pagina delle Recipe quando si sceglie di visualizzare le metriche per una determinata di esse;

- + **metricsType** : **array di oggetti JSON_G**

Descrizione: parametro che contiene la lista delle categorie possibili di metriche. Questo campo viene inizializzato invocando il metodo **getMetricTypes()**;

- + **oneInserted** : **bool**

Descrizione: parametro che serve ad indicare quando almeno una metrica_G è stata aggiunta. Questo blocca degli elementi della form_G che non possono più essere modificati. Viene inizializzata a false e cambia il suo valore la prima volta che il metodo **addMetric(cat, typeCat, value)** viene eseguito;

- + **maxReached** : **bool**

Descrizione: parametro che serve ad indicare se si è raggiunto il massimo numero di metriche possibili per un confronto. Viene inizializzato a false e può cambiare il suo valore all'interno del metodo **addMetric(cat, typeCat, value)**;

- + **categories** : **array di oggetti JSON_G**

Descrizione: attributo che contiene le categoria_G di metriche possibili. L'attributo viene inizializzato andando a prelevare i dati tramite il servizio **RecipeService** attraverso la chiamata al metodo **getMetricType()**;

– + `types` : array di JSON_G

Descrizione: attributo che contiene i tipi di categoria_G disponibili per una determinata categoria. L'attributo viene inizializzato quando viene effettuata la scelta di quale categoria si vuole inserire tramite la chiamata al metodo `updateTypeMetric(category)`;

– + `type` : String

Descrizione: attributo che identifica il tipo di metrica_G che si andrà ad inserire. Viene inizializzato con la stringa vuota;

– + `valueMetric` : String

Descrizione: attributo che identifica il valore di una metrica_G che si andrà a richiedere. Viene inizializzato con la stringa vuota;

– + `metrics` : array di oggetti JSON_G

Descrizione: attributo che contiene tutte le metriche aggiunte alla Recipe_G. Viene inizializzato con un array vuoto;

- Metodi associati allo \$scope e privati:

– - `getMetricTypes()` : array di oggetti JSON_G

Descrizione: metodo che recupera il tipo di metriche presenti attraverso la chiamata `getMetricType()` al servizio **RecipeService**. L'array restituito sarà il seguente:

```
[
  {
    key: 'facebook',
    value: 'Facebook'
  },
  {
    key: 'twitter',
    value: 'Twitter'
  },
  {
    key: 'instagram',
    value: 'Instagram'
  }
]
```

– + `addMetric(cat, typeCat, value)` : void

Descrizione: metodo che aggiunge la metrica_G inserita nel form_G in un array che contiene tutte le metriche aggiunte e cioè **metrics**. Prima di inserirla controlla se i campi siano stati inseriti correttamente. Il formato dell'oggetto JSON_G inserito nell'array dovrà essere il seguente:

```
{
  'id': val,
  'category': cat,
  'category_type': typeCat
}
```

– + `updateTypeMetric(category)` : void

Descrizione: metodo che aggiorna l'attributo **types** quando dal form_G viene scelta la categoria_G della metrica_G. Questi valori vengono presi dal servizio **RecipeService** attraverso la chiamata al metodo **getMetricTypeNode(category)**;

– + **updateMetrics(category, type) : void**

Descrizione: metodo che aggiorna i valori disponibili per una determinata selezione di categoria_G e tipo di categoria avvenuta nella form_G. Questo chiama il metodo **getMetricsList(idRecipe)** del servizio **RecipeService** e filtra i risultati ottenuti in base ai valori inseriti. Una volta filtrati gli associa all'attributo **metrics**.

- **Servizi AngularJS_G utilizzati:** N/A

3.2.11.1.11 client::controller::user::RecipeRequestRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina che serve ad un utente per generare delle richieste di nuove Recipe_G da inserire nel sistema;

- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'utente di fare delle richieste per delle nuove Recipe_G ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;

- **Relazioni con altre classi:**

- client::view_G::user::RecipeRequest
- client::controller::user::RecipeRequestCtrl

- **Attributi associati allo \$scope:** N/A

- **Metodi associati allo \$scope e privati:** N/A

- **Servizi AngularJS_G utilizzati:**

- \$stateProvider

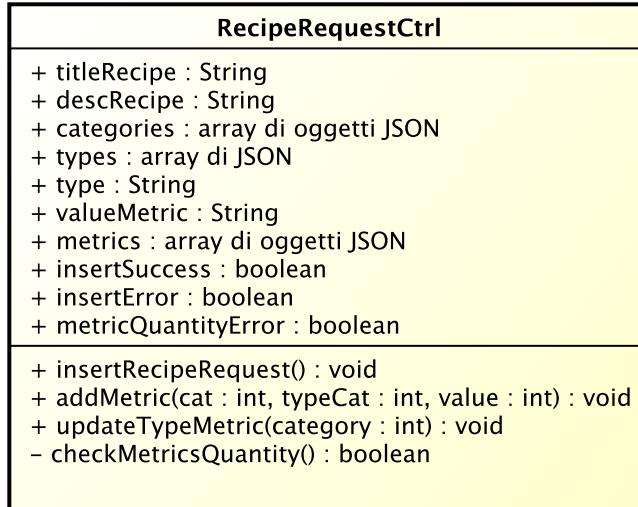


Figura 39: Classe - client::controller::user::RecipeRequestCtrl

3.2.11.1.12 client::controller::user::RecipeRequestCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante il form_G che l'utente deve compilare per effettuare una richiesta per una nuova Recipe_G;
- **Utilizzo:** viene utilizzata per gestire il form_G che l'utente, che desidera fare richiesta per l'inserimento di una nuova Recipe_G, deve compilare. La classe dovrà utilizzare il verificare in una prima istanza, la validità dei campi inseriti, anche se poi sarà il server ha validare effettivamente la richiesta;
- **Relazioni con altre classi:**
 - client::model::data::RecipeRequestModel
 - client::view_G::user::RecipeRequest
 - client::controller::user::RecipeRequestRoute
- **Attributi associati allo \$scope:**
 - + `titleRecipe : String`

Descrizione: attributo che identifica il titolo della Recipe_G che si andrà a richiedere. Viene inizializzato con la stringa vuota. Questo campo è obbligatorio da inserire per poter inserire una nuova Recipe;
 - + `descRecipe : String`

Descrizione: attributo che identifica la descrizione della Recipe_G che si andrà a richiedere. Viene inizializzato con la stringa vuota;
 - + `categories : array di oggetti JSONc`

Descrizione: attributo che contiene le categoria_G di metriche possibili. L'attributo viene inizializzato andando a prelevare i dati tramite il servizio **RecipeService** attraverso la chiamata al metodo **getMetricType()**;
 - + `types : array di JSONc`

Descrizione: attributo che contiene i tipi di categoria_G disponibili per una determinata categoria. L'attributo viene inizializzato quando viene effettuata la scelta di quale categoria si vuole inserire tramite la chiamata al metodo **updateTypeMetric(category)**;
 - + `type : String`

Descrizione: attributo che identifica il tipo di metrica_G che si andrà ad inserire. Viene inizializzato con la stringa vuota;
 - + `valueMetric : String`

Descrizione: attributo che identifica il valore di una metrica_G che si andrà a richiedere. Viene inizializzato con la stringa vuota;
 - + `metrics : array di oggetti JSONc`

Descrizione: attributo che contiene tutte le metriche aggiunte alla Recipe_G. Viene inizializzato con un array vuoto;
 - + `insertSuccess : bool`

Descrizione: attributo che identifica se l'inserimento della richiesta di una nuova Recipe_G è avvenuto con successo. Viene inizializzato a false;

– + `insertError : bool`

Descrizione: attributo che identifica se l'inserimento della richiesta di una nuova Recipe_G è fallito. Viene inizializzato a false;

– + `metricQuantityError : bool`

Descrizione: attributo che identifica se si sono inserite almeno due metriche prima di poter effettuare la richiesta di inserimento di una nuova Recipe_G. Viene inizializzato a false e viene modificato tramite la chiamata al metodo `checkMetricsQuantity()` nel caso la quantità fosse inferiore a quanto descritto.

- **Metodi associati allo \$scope e privati:**

– + `insertRecipeRequest() : void`

Descrizione: metodo che viene invocato quando si vuole inviare una nuova richiesta Recipe_G. Questo prima controlla se i campi obbligatori sono stati inseriti e se il numero di metriche minimo è stato aggiunto attraverso il metodo `checkMetricsQuantity()`. Se tutti i valori sono corretti allora crea un oggetto JSON_G con i dati prelevati dal form_G e chiama il metodo `createRecipe(value)` del servizio **RecipeAdminService**. Il formato dell'oggetto JSON passato dovrà essere il seguente:

```
{
  'title': vm.titleRecipe,
  'desc': vm.descRecipe,
  'user_id': id,
  'metrics': vm.metrics
}
```

– + `addMetric(cat, typeCat, value) : void`

Descrizione: metodo che aggiunge la metrica_G inserita nel form_G in un array che contiene tutte le metriche aggiunte e cioè **metrics**. Prima di inserirla controlla se i campi siano stati inseriti correttamente. Il formato dell'oggetto JSON_G inserito nell'array dovrà essere il seguente:

```
{
  'id': val,
  'category': cat,
  'category_type': typeCat
}
```

– + `updateTypeMetric(category) : void`

Descrizione: metodo che aggiorna l'attributo **types** quando dal form_G viene scelta la categoria_G della metrica_G. Questi valori vengono presi dal servizio **RecipeService** attraverso la chiamata al metodo `getMetricTypeNode(category)`;

– - `checkMetricsQuantity() : bool`

Descrizione: metodo che controlla che nell'array **metrics** ci siano almeno due valori e restituisce true in tal caso.

- **Servizi AngularJS_G utilizzati:** N/A

3.2.11.1.13 client::controller::user::FavouritesRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina che serve ad un utente per guardare tutti i grafici che si è salvato tra i preferiti;
- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'utente di visualizzare i grafici che ha tra i preferiti ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;
- **Relazioni con altre classi:**
 - client::view_G::user::Favourites
 - client::controller::user::FavouritesCtrl
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

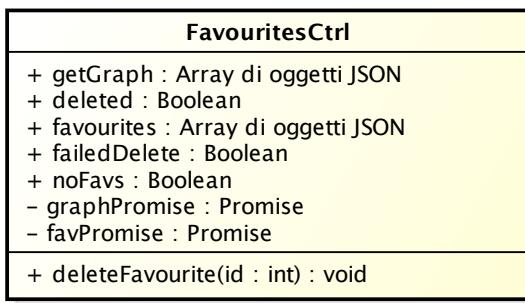


Figura 40: Classe - client::controller::user::FavouritesCtrl

3.2.11.1.14 client::controller::user::FavouritesCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina che mostra tutti i grafici che l'utente ha salvato tra i preferiti;
- **Utilizzo:** viene utilizzata per gestire tutti i grafici che l'utente ha salvato tra i preferiti;
- **Relazioni con altre classi:**
 - client::model::services::RecipeService
 - client::view_G::user::Favourites
 - client::controller::user::FavouritesRoute
- **Attributi associati allo \$scope:**
 - **+ getGraph :** Array di oggetti JSON_G
- **Descrizione:** contiene i grafici che devono essere visualizzati in un JSON_G della forma:

```
{
    desc: 'Descrizione del grafico',
    data: codice HTML per generare il grafico
    info: nome della funzione di ViewTypeModel che ha generato il grafico
}
```

– + favourites : Array di oggetti JSON_G

Descrizione: contiene i favoriti che devono essere visualizzati in un JSON_G della forma:

```
{
    metric: MetricModel,
    id: identificativo del favorito,
    chart_type: nome della funzione di ViewTypeModel che ha generato il grafico
}
```

– + deleted : Boolean

Descrizione: descrive se è stato cancellato un favorito

– + failedDelete : Boolean

Descrizione: descrive se è stata fallita una cancellazione di un favorito

– + noFavs : Boolean

Descrizione: descrive se non è presente nessun favorito

– - graphPromise : Promise

Descrizione: Promessa dei grafici da visualizzare tra i favoriti

– - favPromise : Promise

Descrizione: Promessa dei dati relativi ai favoriti

- Metodi associati allo \$scope e privati:

– + deleteFavourite(id)

Descrizione: metodo per chiamare i metodi di UserService per richiedere la rimozione del favorito segnato come con identificativo id.

- Servizi AngularJS_G utilizzati: N/A

3.2.11.1.15 client::controller::user::TokenConfigRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina che serve ad un utente per generare il token che gli consente di utilizzare i servizi REST_G che l'applicazione offre;

- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'utente di generare il token necessario ad utilizzare i servizi REST_G pubblici ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;

- Relazioni con altre classi:

– client::view_G::user::TokenConfig

– client::controller::user::TokenConfigCtrl

- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

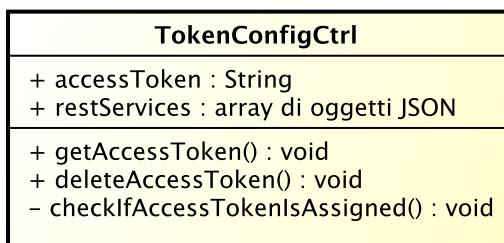


Figura 41: Classe - client::controller:user::TokenConfigCtrl

3.2.11.1.16 client::controller::user::TokenConfigCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina che fornisce la possibilità ad un utente di ottenere un token da utilizzare per interrogare i servizi REST_G offerti;
- **Utilizzo:** viene utilizzata per gestire le operazioni che forniscono all’utente il token necessario ad interrogare i servizi REST_G offerti;
- **Relazioni con altre classi:**
 - client::model::services::UserService
 - client::view_G::user::TokenConfig
 - client::controller::user::TokenConfigRoute

- **Attributi associati allo \$scope:**

– + accessToken : String

Descrizione: attributo che identifica il token di accesso ai servizi REST_G utilizzabili dall’utente. Questo campo viene inizializzato con la stringa vuota se il token non è ancora stato generato, altrimenti riceve il valore corretto;

– + restServices : array di oggetti JSON_G

Descrizione: attributo che identifica la lista dei servizi REST_G disponibili all’utente. Il formato degli oggetti JSON_G contenuti nell’array è il seguente:

```
{
  type: 'GET',
  req: '/recipes',
  desc: 'Get the list of the available recipes'
}
```

- **Metodi associati allo \$scope e privati:**

– + getAccessToken() : void

Descrizione: metodo che chiama il metodo **getAccessToken(idUser)** del servizio **UserService** per recuperare il token e lo assegna all'attributo **accessToken** se la chiamata è andata a buon fine;

- + **deleteAccessToken() : void**

Descrizione: metodo che chiama il metodo **deleteAccessToken(idUser)** del servizio **UserService** per rimuovere il token e assegna all'attributo **accessToken** una stringa vuota;

- - **checkIfAccessTokenIsAssigned() : void**

Descrizione: metodo che attraverso una chiamata al metodo **checkAccessToken(idUser)** del servizio **UserService** salva il valore del token qualora fosse già stato generato e lo assegna all'attributo **accessToken**.

- **Servizi AngularJS_G utilizzati:** N/A

3.2.11.1.17 client::controller::user::SettingsRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina contenente le informazioni associate ad un utente e le modifiche che può effettuare su quei dati;

- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'utente di visualizzare le impostazioni del suo profilo ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;

- **Relazioni con altre classi:**

- client::view_G::user::Settings
- client::controller::user::SettingsCtrl

- **Attributi associati allo \$scope:** N/A

- **Metodi associati allo \$scope e privati:** N/A

- **Servizi AngularJS_G utilizzati:**

- \$stateProvider

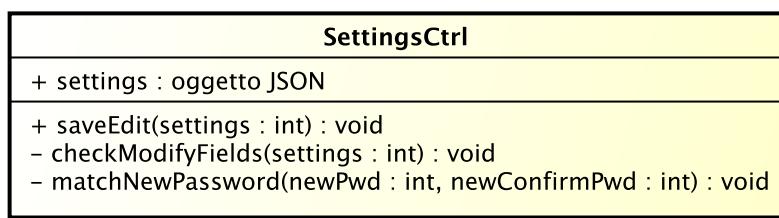


Figura 42: Classe - client::controller::user::SettingsCtrl

3.2.11.1.18 client::controller::user::SettingsCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina che mostra le informazioni dell'utente e la possibilità di modificarle;

- **Utilizzo:** viene utilizzata per gestire la visualizzazione dei dati personali associati all'utente e per le operazioni di modifica;

- **Relazioni con altre classi:**

- client::model::data::UserModel
- client::view_G::user::Settings
- client::controller::user::SettingsRoute

- **Attributi associati allo \$scope:**

- + **settings** : oggetto JSON_G

Descrizione: oggetto nel formato JSON_G che contiene i valori delle impostazioni del profilo utente. I campi username e mail vengono inizializzati quando l'utente accede alla impostazioni, tramite il servizio AuthService. Il formato dell'oggetto JSON è il seguente:

```
{
  username: '',
  email: '',
  oldPassword: '',
  newPassword: ''
  confirmNewPwd: ''
}
```

- **Metodi associati allo \$scope e privati:**

- + **saveEdit(settings)**

Descrizione: metodo che preso in input un oggetto JSON_G del tipo di quello visto per l'attributo **settings** va a chiamare il servizio AuthService per salvare le modifiche effettuate. Tutto questo dopo avere controllato se la password vecchia era valida e se la password nuova combaciava con quella di conferma;

- + **deleteAccount()**

Descrizione: metodo che cancella l'account dell'utente che ne ha richiesto la cancellazione. Questo richiama il metodo **deleteAccount(idUser)** del servizio **AuthService**;

- - **checkModifyFields(settings)**

Descrizione: metodo che controlla quali campi e se sono stati modificati effettivamente e ritorna true se è stata effettuata qualche modifica e quindi ha senso chiamare il servizio per aggiornare i valori. Questo metodo viene utilizzato dentro il metodo saveEdit(settings);

- - **matchNewPassword(newPwd, newConfirmPwd)**

Descrizione: metodo che presi in input due valori, controlla se sono uguali e in tal caso restituisce true.

- **Servizi AngularJS_G utilizzati:** N/A

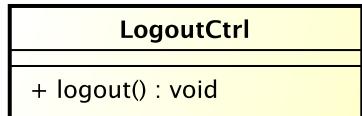


Figura 43: Classe - client::controller::user::SettingsCtrl

3.2.11.1.19 client::controller::user::LogoutCtrl

- **Descrizione:** la classe è il controller che serve a gestire le operazioni di logout di un utente;
- **Utilizzo:** viene utilizzata per distruggere la sessione attuale di un utente, attraverso la chiamata ad opportuni servizi, e il re-indirizzamento alla pagina di Login;
- **Relazioni con altre classi:**
 - client::model::services::AuthService
 - client::view_G::user::MenuMain
 - client::controller::public::PublicRoute
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:**
 - `+ logout()`

Descrizione: metodo che chiama il servizio AuthService utilizzando il metodo logout() per fare uscire l'utente dal sistema;

3.2.12 client::controller::admin

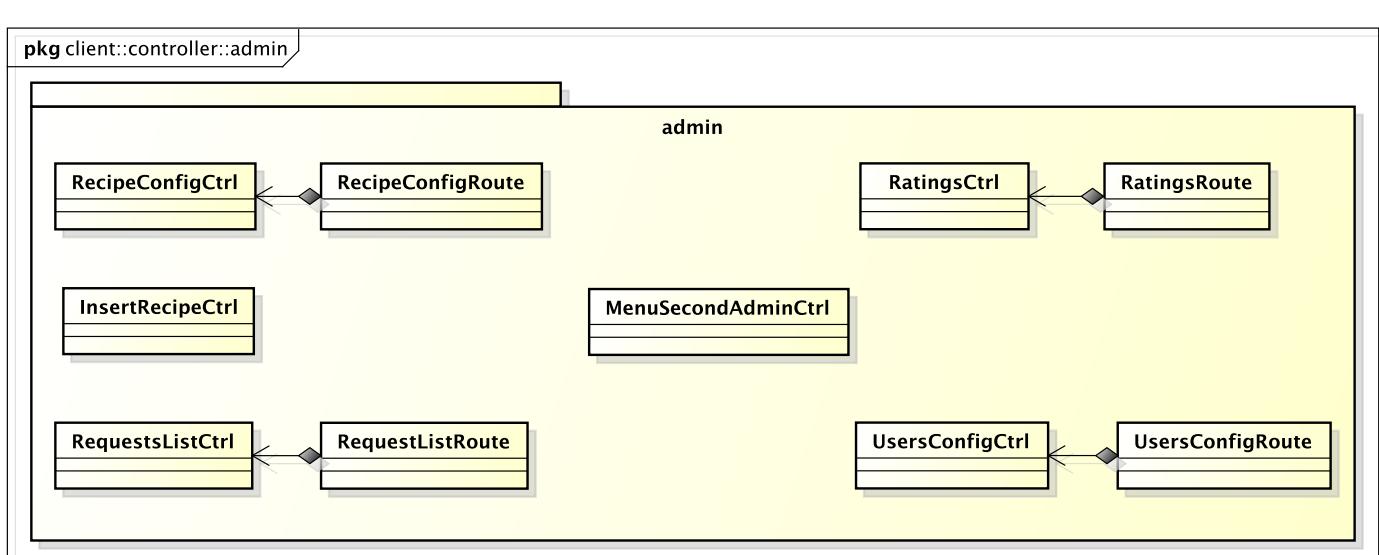


Figura 44: Package - client::controller::admin

- **Descrizione:** è il package_G che contiene le classi che controllano le decisioni di che pagine HTML mostrare all'amministratore e di quali operazioni poter effettuare su di esse;

- **Padre:** client::controller
- **Interazione con altri componenti:**
 - client::model::admin
 - client::view_G::admin

3.2.12.1 Classi

3.2.12.1.1 client::controller::admin::MenuSecondAdminCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante il menu secondario dell'applicazione che ha a disposizione in più l'amministratore;
- **Utilizzo:** viene utilizzata per gestire i collegamenti secondari alle pagine del sistema che l'amministratore ha a disposizione per andare ad effettuare le operazioni aggiuntive a lui consentite;
- **Classi ereditate:**
 - client::controller::user::MenuSecondCtrl. Questa ereditarietà è solo logica e non viene implementata in nessuna forma.
- **Relazioni con altre classi:**
 - client::view_G::admin::MenuSecondAdmin
 - client::controller::admin::RecipeConfigRoute
 - client::controller::admin::RequestListRoute
 - client::controller::admin::RatingsRoute
 - client::controller::admin::UserConfigRoute
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A

3.2.12.1.2 client::controller::admin::RecipeConfigRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina che consente di inserire una nuova Recipe_G nel sistema;
- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'amministratore di andare ad inserire una nuova Recipe_G nel sistema ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;
- **Relazioni con altre classi:**
 - client::view_G::admin::RecipeConfig
 - client::controller::admin::RecipeConfigCtrl
- **Attributi associati allo \$scope:** N/A

- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

3.2.12.1.3 client::controller::admin::RecipeConfigCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina che consente ad un amministratore di andare ad inserire una nuova Recipe_G;
- **Utilizzo:** viene utilizzata per gestire le operazioni che servono ad andare ad inserire una nuova Recipe_G;
- **Relazioni con altre classi:**
 - client::model::data::RecipeInsertModel
 - client::view_G::admin::RecipeConfig
 - client::controller::admin::RecipeConfigRoute
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:** N/A

3.2.12.1.4 client::controller::admin::RequestListRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina che mostra la lista di tutte le richieste di nuove Recipe_G;
- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'amministratore di visualizzare la lista di tutte le richieste di nuove Recipe_G proposte dagli utenti ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;
- **Relazioni con altre classi:**
 - client::view_G::admin::RecipeList
 - client::controller::admin::RecipeListCtrl
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

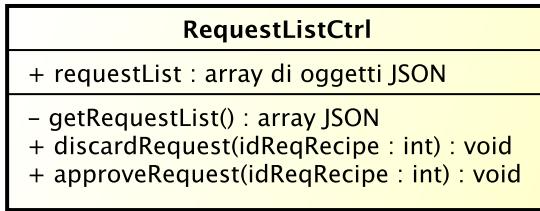


Figura 45: Classe - client::controller::admin::RequestListCtrl

3.2.12.1.5 client::controller::admin::RequestListCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina che consente ad un amministratore di visualizzare tutte le richieste di nuove Recipe_G inoltrate dagli utenti;
- **Utilizzo:** viene utilizzata per gestire le operazioni che servono ad andare a visualizzare la lista di tutte le richieste di nuove Recipe_G. Una volta scelta la richiesta da esaminare, l'amministratore potrà cliccare su un apposito pulsante per visualizzarne i dettagli;
- **Relazioni con altre classi:**
 - client::model::services::RecipeAdminService
 - client::view_G::admin::RecipeList
 - client::controller::admin::RecipeListRoute
- **Attributi associati allo \$scope:**
 - + requestList : array di oggetti JSON_G

Descrizione: contiene la lista delle richieste di nuove Recipe_G proveniente dagli utenti non amministratori. Il formato degli oggetti JSON_G contenuti nell'array è il seguente:

```
{
    idRequestRecipe: '42',
    titleRecipe: 'Titolo della Recipe\gloss{}',
    descRecipe: 'Descrizione della Recipe\gloss{}',
    emailUser: 'info@mashup-unipd.it'
}
```

- **Metodi associati allo \$scope e privati:**

- - getRequestList() : array JSON_G

Descrizione: metodo che attraverso l'invocazione del metodo **getListOfRecipesRequest()** sul servizio **recipeAdminService** riceve una promessa alla chiamata effettiva per recuperare i dati dal backend. Quando questa sarà soddisfatta, l'attributo **requestList** verrà popolato con i dati ricevuti.

- + discardRequest(idReqRecipe) : void

Descrizione: metodo che attraverso l'invocazione del metodo **discardRecipeRequest(idReqRecipe)** sul servizio **recipeAdminService** riceve una promessa alla chiamata effettiva per andare a rifiutare la richiesta per quella recipe_G. Quando questa sarà soddisfatta, verrà visualizzato un messaggio di conferma.

– + approveRequest(idReqRecipe) : void

Descrizione: metodo che attraverso l'invocazione del metodo `approveRecipeRequest(idReqRecipe)` sul servizio `recipeAdminService` riceve una promessa alla chiamata effettiva per andare ad accettare e inserire la richiesta per quella recipe_G . Quando questa sarà soddisfatta, verrà visualizzato un messaggio di conferma.

- **Servizi AngularJS_G utilizzati:** N/A

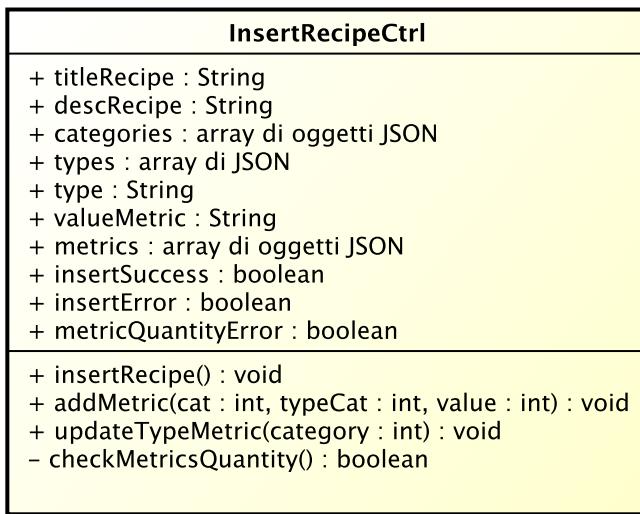


Figura 46: Classe - client::controller::admin::InsertRecipeCtrl

3.2.12.1.6 client::controller::admin::InsertRecipeCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante il form_G contenente i campi necessari all'inserimento di una nuova Recipe_G;
- **Utilizzo:** viene utilizzata per gestire il form_G che serve per inserire una nuova Recipe_G nel sistema. Può essere utilizzato sia per il form di un inserimento ex novo sia per gestire i dettagli delle richieste di nuove Recipe;
- **Relazioni con altre classi:**
 - client::model::data::RecipeInsertModel
 - client::view_G::admin::Insert
 - client::controller::admin::RecipeConfigRoute
- **Attributi associati allo \$scope:**
 - + titleRecipe : String

Descrizione: attributo che identifica il titolo della Recipe_G che si andrà ad inserire. Viene inizializzato con la stringa vuota. Questo campo è obbligatorio da inserire per poter inserire una nuova Recipe;

 - + descRecipe : String

Descrizione: attributo che identifica la descrizione della Recipe_G che si andrà ad inserire. Viene inizializzato con la stringa vuota;

– + `categories : array di oggetti JSONG`

Descrizione: attributo che contiene le categoria_G di metriche possibili. L'attributo viene inizializzato andando a prelevare i dati tramite il servizio **RecipeService** attraverso la chiamata al metodo **getMetricType()**;

– + `types : array di JSONG`

Descrizione: attributo che contiene i tipi di categoria_G disponibili per una determinata categoria. L'attributo viene inizializzato quando viene effettuata la scelta di quale categoria si vuole inserire tramite la chiamata al metodo **updateTypeMetric(category)**;

– + `type : String`

Descrizione: attributo che identifica il tipo di metrica_G che si andrà ad inserire. Viene inizializzato con la stringa vuota;

– + `valueMetric : String`

Descrizione: attributo che identifica il valore di una metrica_G che si andrà ad inserire. Viene inizializzato con la stringa vuota;

– + `metrics : array di oggetti JSONG`

Descrizione: attributo che contiene tutte le metriche aggiunte alla Recipe_G. Viene inizializzato con un array vuoto;

– + `insertSuccess : bool`

Descrizione: attributo che identifica se l'inserimento della nuova Recipe_G è avvenuto con successo. Viene inizializzato a false;

– + `insertError : bool`

Descrizione: attributo che identifica se l'inserimento della nuova Recipe_G è fallito. Viene inizializzato a false;

– + `metricQuantityError : bool`

Descrizione: attributo che identifica se si sono inserite almeno due metriche prima di poter effettuare l'inserimento di una nuova Recipe_G. Viene inizializzato a false e viene modificato tramite la chiamata al metodo **checkMetricsQuantity()** nel caso la quantità fosse inferiore a quanto descritto.

- **Metodi associati allo \$scope e privati:**

– + `insertRecipe() : void`

Descrizione: metodo che viene invocato quando si vuole inserire una nuova Recipe_G. Questo prima controlla se i campi obbligatori sono stati inseriti e se il numero di metriche minimo è stato aggiunto attraverso il metodo **checkMetricsQuantity()**. Se tutti i valori sono corretti allora crea un oggetto JSON_G con i dati prelevati dal form_G e chiama il metodo **createRecipe(value)** del servizio **RecipeAdminService**. Il formato dell'oggetto JSON passato dovrà essere il seguente:

```
{
  'title': vm.titleRecipe,
  'desc': vm.descRecipe,
  'admin_id': id,
```

```

    'metrics': vm.metrics
}

- + addMetric(cat, typeCat, value) : void

```

Descrizione: metodo che aggiunge la metrica_G inserita nel form_G in un array che contiene tutte le metriche aggiunte e cioè **metrics**. Prima di inserirla controlla se i campi siano stati inseriti correttamente. Il formato dell'oggetto JSON_G inserito nell'array dovrà essere il seguente:

```

{
  'id': val,
  'category': cat,
  'category_type': typeCat
}

```

```
- + updateTypeMetric(category) : void
```

Descrizione: metodo che aggiorna l'attributo **types** quando dal form_G viene scelta la categoria_G della metrica_G. Questi valori vengono presi dal servizio **RecipeService** attraverso la chiamata al metodo **getMetricTypeNode(category)**;

```
- - checkMetricsQuantity() : bool
```

Descrizione: metodo che controlla che nell'array **metrics** ci siano almeno due valori e restituisce true in tal caso.

- **Servizi AngularJS_G utilizzati:** N/A

3.2.12.1.7 client::controller::admin::RatingsRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina che mostra la lista di tutte le Recipe_G votate con la media del valore assegnato dagli utenti;
- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'amministratore di visualizzare la lista di tutte le Recipe_G votate, in ordine dalla più votata, ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;
- **Relazioni con altre classi:**
 - client::view_G::admin::Ratings
 - client::controller::admin::RatingsCtrl
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

RatingsCtrl
+ listRecipes : array di oggetti JSON
- getListOfRecipes() : void

Figura 47: Classe - client::controller::admin::RatingsCtrl

3.2.12.1.8 client::controller::admin::RatingsCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina che consente ad un amministratore di visualizzare tutte le Recipe_G che sono state votate dagli utenti;
- **Utilizzo:** viene utilizzata per gestire le operazioni che servono ad andare a visualizzare la lista di tutte le Recipe_G che sono state votate dagli utenti, ordinandole dalla più votata alla meno;

- **Relazioni con altre classi:**

- client::model::services::RecipeAdminService
- client::view_G::admin::Ratings
- client::controller::admin::RatingsRoute

- **Attributi associati allo \$scope:**

- + listRecipes : array di oggetti JSON_G

Descrizione: attributo che contiene la lista di tutte le Recipe_G aventi una valutazione. Il formato degli oggetti JSON_G contenuti nell'array è il seguente:

```
{
    id: '42',
    title: 'Titolo della Recipe\gloss{}',
    desc: 'Descrizione della Recipe\gloss{}',
    ratings: '3.44'
}
```

- **Metodi associati allo \$scope e privati:**

- - getListOfRecipes() : void

Descrizione: metodo che recupera la lista della recipe_G, aventi una valutazione, attraverso la chiamata del metodo **getRecipesListAll()** del servizio **recipeAdminService**. Quando i dati sono disponibili, riempie l'array **listRecipes**. L'invocazione del metodo avviene subito nel momento in cui un utente accede alla pagina **Ratings**.

- **Servizi AngularJS_G utilizzati:** N/A

3.2.12.1.9 client::controller::admin::UserConfigRoute

- **Descrizione:** la classe serve a gestire l'indirizzamento e l'assegnazione di uno stato al template_G HTML riguardante la pagina che mostra la lista degli utenti registrati al sistema;

- **Utilizzo:** viene utilizzata per reindirizzare l'utente al template_G HTML della pagina che permette all'amministratore di visualizzare la lista degli utenti registrati al sistema ed assegnare al template scelto il controller opportuno per gestire i diversi compiti;
- **Relazioni con altre classi:**
 - client::view_G::admin::UserConfig
 - client::controller::admin::UserConfigCtrl
- **Attributi associati allo \$scope:** N/A
- **Metodi associati allo \$scope e privati:** N/A
- **Servizi AngularJS_G utilizzati:**
 - \$stateProvider

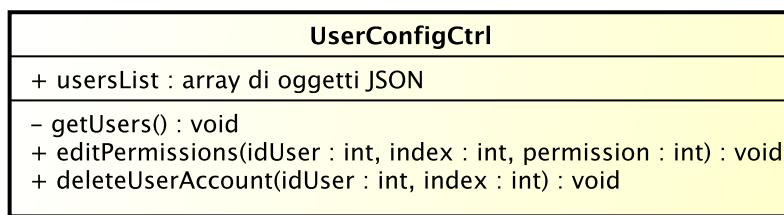


Figura 48: Classe - client::controller::admin::UserConfigCtrl

3.2.12.1.10 client::controller::admin::UserConfigCtrl

- **Descrizione:** la classe è il controller che serve a gestire la logica applicativa riguardante la pagina che consente ad un amministratore di visualizzare la lista degli utenti registrati al sistema;
- **Utilizzo:** viene utilizzata per gestire le operazioni che servono ad andare a visualizzare la lista degli utenti registrati al sistema, potendo su di essi, andare a modificarne i permessi o a cancellare l'account;
- **Relazioni con altre classi:**
 - client::model::services::UserAdminService
 - client::view_G::admin::UserConfig
 - client::controller::admin::UserConfigRoute
- **Attributi associati allo \$scope:**
 - + **usersList** : array di oggetti JSON_G

Descrizione: attributo che contiene la lista di tutti gli utenti registrati nel sistema. Il formato degli oggetti JSON_G contenuti nell'array è il seguente:

```
{
  access_token: '3253224346356',
  username: 'MashUp',
  email: 'info@mashup-unipd.it',
  permission: 'user'
}
```

- Metodi associati allo \$scope e privati:

- `- getUsers() : void`

Descrizione: metodo che recupera la lista degli utenti attraverso la chiamata del metodo `getListOfUsers` del servizio `userAdminService`. Quando i dati sono disponibili, riempie l'array `usersList`. L'invocazione del metodo avviene subito nel momento in cui un utente accede alla pagina `UsersConfig`;

- `+ editPermissions(idUser, index, permission) : void`

Descrizione: metodo che modifica il tipo dell'utente, attraverso la chiamata del metodo `editUserPermissions(idUser, permission)` del servizio `userAdminService`. Quando la modifica è stata effettuata viene visualizzato un messaggio di successo e viene anche modificato il valore relativo presente nell'array `usersList` in modo da rispecchiare visivamente la diversa situazione;

- `+ deleteUserAccount(idUser, index) : void`

Descrizione: metodo che cancella l'utente, attraverso la chiamata del metodo `deleteUserAccount(idUser)` del servizio `userAdminService`. Quando la cancellazione è stata effettuata viene visualizzato un messaggio di successo e viene anche rimosso il valore relativo presente nell'array `usersList` in modo da rispecchiare visivamente la diversa situazione.

- Servizi AngularJS_G utilizzati: N/A

3.3 Server (Back-end)

3.3.1 server

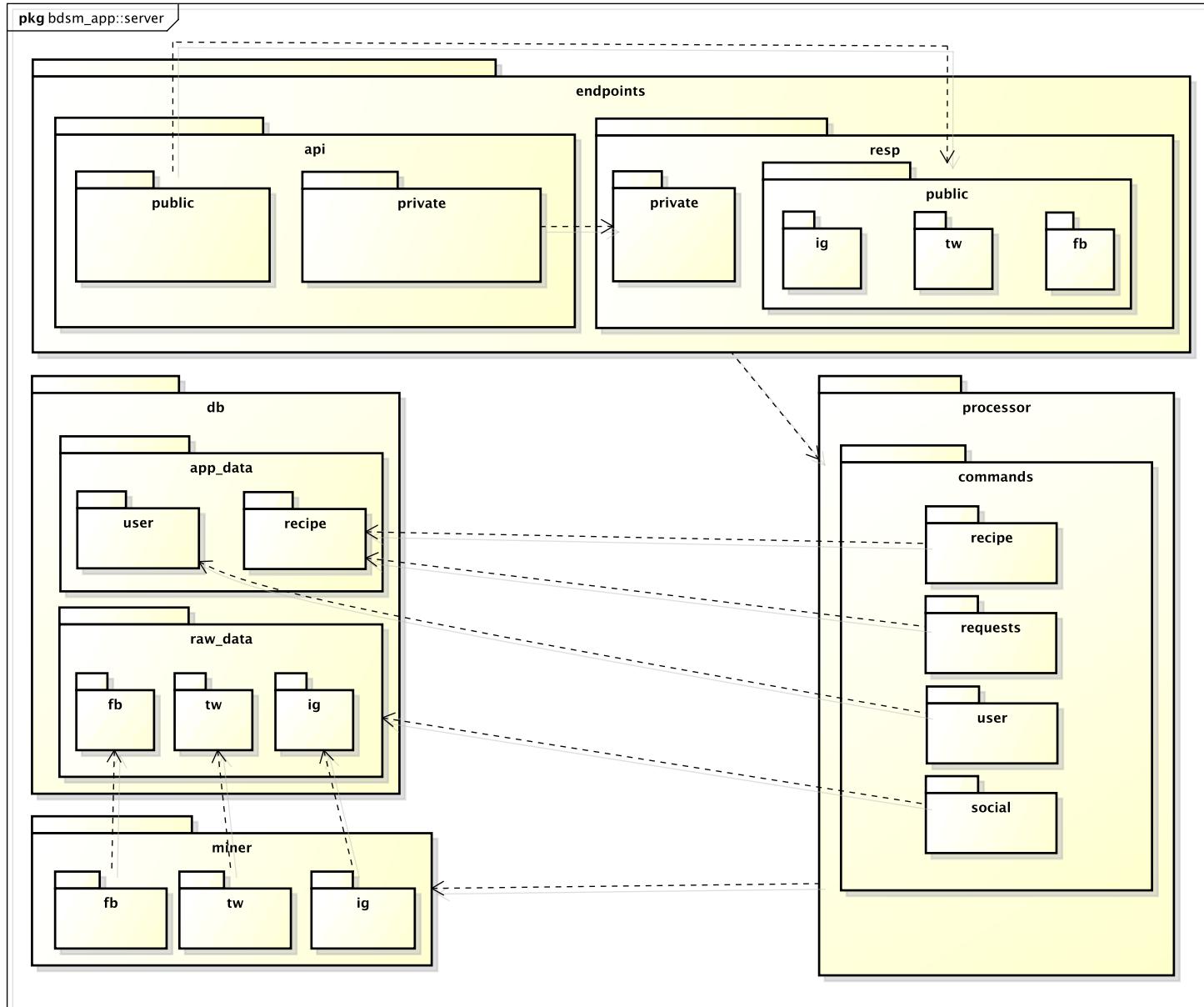


Figura 49: Package - server

- **Descrizione:** è il package_G che racchiude tutte le parti del back-end. È quindi l'insieme dei componenti che si occupa di soddisfare le richieste provenienti dal front-end, interagire col database_G, raccogliere i dati provenienti dai social networks, elaborarli ed esporli attraverso servizi REST_G.

- **Package_G contenuti:**

- `server::processorG`
- `server::db`
- `server::minerG`

- server::endpoints
- **Interazione con altri componenti:** interagisce con il client definito nella sezione 3.2 tramite i servizi REST offerti dal modulo in questione;

3.3.2 server::db

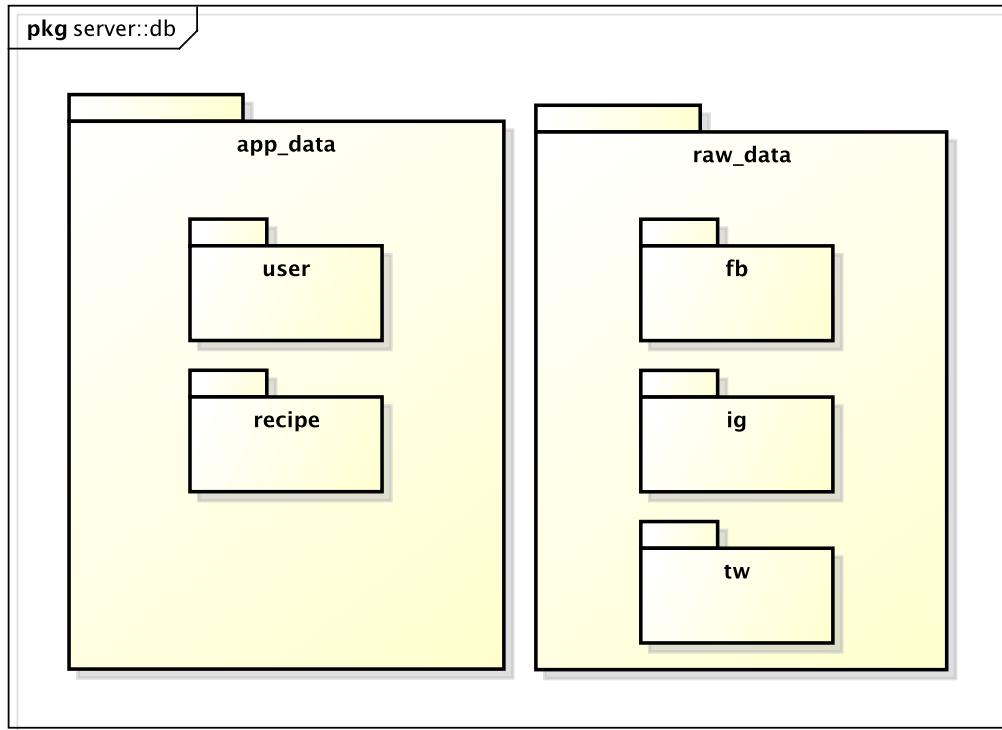


Figura 50: Package - server::db

- **Descrizione:** è il package_G che contiene le componenti che gestiscono e mantengono coerente la base di dati. Esse utilizzano standard proprietari Google per la loro implementazione. Sono suddivise in due package: uno atto a rappresentare il modello dei dati grezzi, l'altro per parametri del software e degli utenti;
- **Padre:** server;
- **Package_G contenuti:**
 - server::app_data.
 - server::raw_data;

3.3.3 server::db::raw_data

- **Descrizione:** package_G che definisce il modello dei dati grezzi ricavati dai vari social network;
- **Padre:** server::db;
- **Package_G contenuti:**
 - server::db::raw_data::fb
 - server::db::raw_data::tw
 - server::db::raw_data::ig

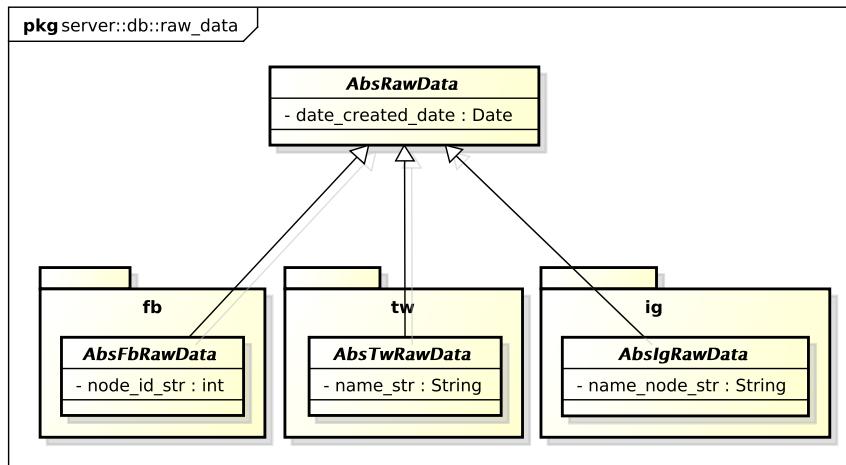


Figura 51: Package - server::db::raw_data

3.3.3.1 Classi

3.3.3.1.1 server::db::raw_data::AbsRawData

- **Descrizione:** classe astratta che definisce il modello di un dato grezzo;
- **Utilizzo:** la classe funge da padre per tutte le classi rappresentanti un dato grezzo;
- **Relazioni con altre classi:**
 - server::db::raw_data::fb::AbsFbRawData
 - server::db::raw_data::tw::AbsTwRawData
 - server::db::raw_data::ig::AbsIgRawData
 - server::db::raw_data::fb::RawFbPageTrend
 - server::db::raw_data::fb::RawFbEventTrend
 - server::db::raw_data::fb::RawFbPostTrendTrend
 - server::db::raw_data::tw::RawTwUserTrend
 - server::db::raw_data::tw::RawTwUserTweet
 - server::db::raw_data::ig::RawIgUserTrend
 - server::db::raw_data::ig::RawIgHashtagTrend
 - server::db::raw_data::ig::RawIgMedia
- **Attributi:**
 - + `date_created_date` : Date

Descrizione: data acquisizione dati grezzi
- **Metodi:** N/A

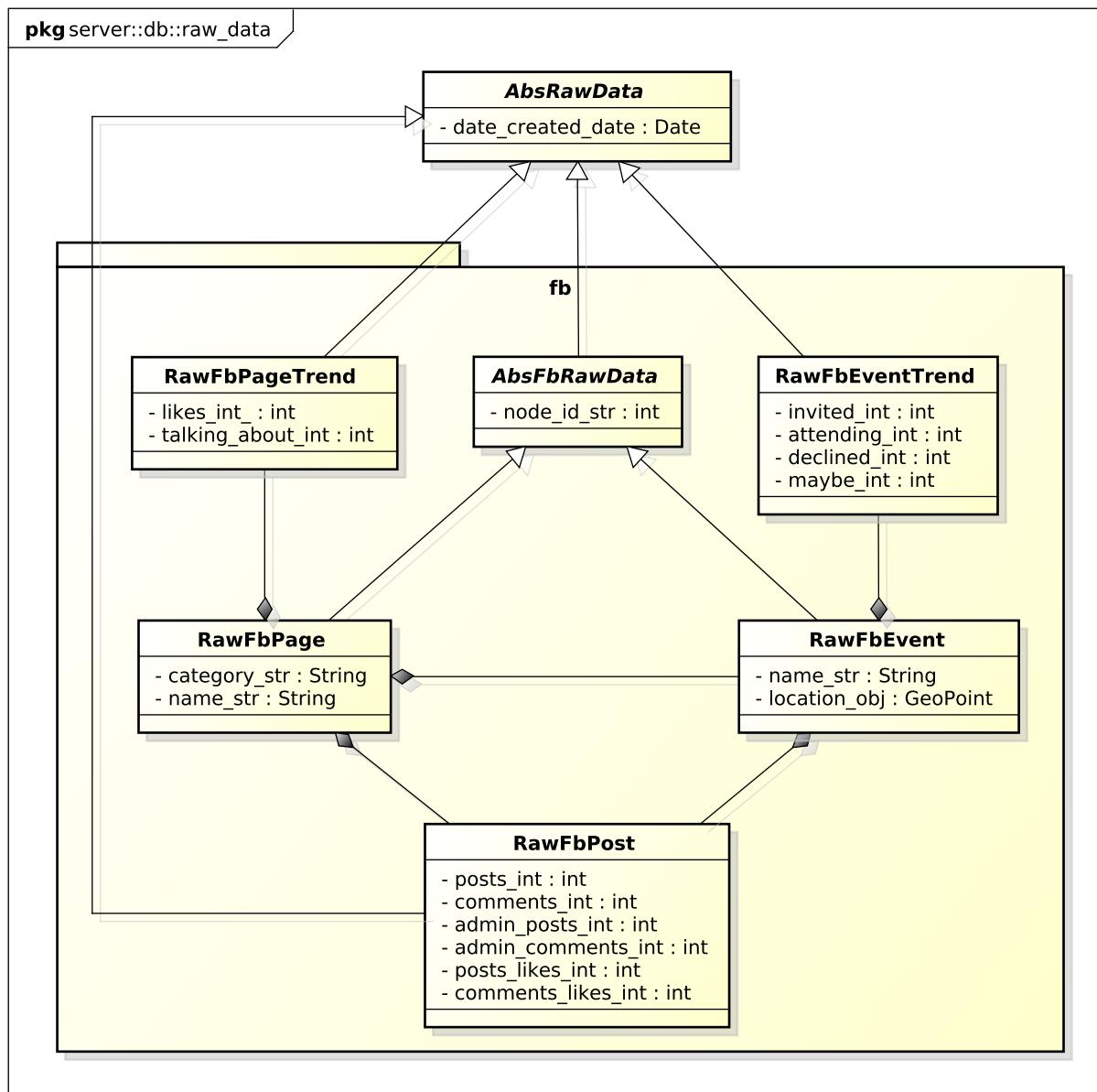


Figura 52: Package - `server::db::raw_data::fb`

3.3.4 server::db::raw_data::fb

- **Descrizione:** è il package_G contenente le classi che definiscono i modelli dei dati grezzi relativi a Facebook;
- **Padre:** server::db::raw_data
- **Interazione con altri componenti:**
 - server::db

3.3.4.1 Classi

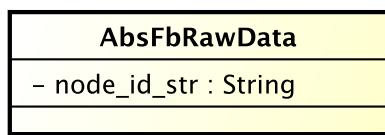


Figura 53: Classe - server::db::raw_data::fb::AbsFbRawData

3.3.4.1.1 server::db::raw_data::fb::AbsFbRawData

- **Descrizione:** classe astratta che definisce il modello dei dati grezzi relativi a Facebook;
- **Utilizzo:** la classe contiene l'id fornito dall'utente il quale permette di identificare univocamente la risorsa nel social media;
- **Classi ereditate:** server::db::raw_data::AbsRawData
- **Attributi:**
 - + node_id_str : String

Descrizione: codice identificativo di un evento o di una pagina Facebook.
- **Metodi:** N/A

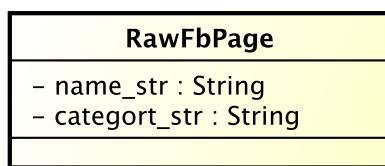


Figura 54: Classe - server::db::raw_data::fb::RawFbPage

3.3.4.1.2 server::db::raw_data::fb::RawFbPage

- **Descrizione:** classe che rappresenta il modello di una pagina Facebook;
- **Utilizzo:** la classe fornisce metodi per memorizzare i dati statici di una pagina Facebook;
- **Classi ereditate:** server::db::raw_data::AbsFbRawData
- **Relazioni con altre classi:**

- server::db::raw_data::fb::RawFbEvent
- server::db::raw_data::fb::RawFbPageTrend
- server::db::raw_data::fb::RawFbPostTrend

- **Attributi:**

- + `name_str` : `String`

Descrizione: nome della pagina Facebook.

- + `category_str` : `String`

Descrizione: categoria_G della pagina Facebook.

- **Metodi:** N/A

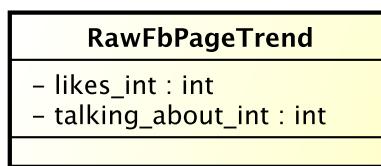


Figura 55: Classe - server::db::raw_data::fb::RawFbPageTrend

3.3.4.1.3 server::db::raw_data::fb::RawFbPageTrend

- **Descrizione:** classe che rappresenta il modello del trend dei dati una pagina Facebook;

- **Utilizzo:** la classe viene utilizzata per memorizzare e il numero di like e di talking about di ogni singola pagina. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;

- **Classi ereditate:** server::db::raw_data::AbsRawData

- **Attributi:**

- + `likes_int` : int

Descrizione: numero dei likes di una determinata pagina Facebook,

- + `talking_about_int` : int

Descrizione: numero di talking about di una determinata pagina Facebook.

- **Metodi:** N/A

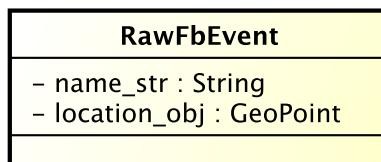


Figura 56: Classe - server::db::raw_data::fb::RawFbEvent

3.3.4.1.4 server::db::raw_data::fb::RawFbEvent

- **Descrizione:** classe che rappresenta il modello di un evento Facebook;

- **Utilizzo:** la classe fornisce metodi per memorizzare i dati statici di un evento Facebook;
- **Classi ereditate:** server::db::raw_data::AbsFbRawData
- **Relazioni con altre classi:**
 - server::db::raw_data::fb::RawFbEventTrend
 - server::db::raw_data::fb::RawFbPostTrend
- **Attributi:**
 - + `name_str` : `String`
Descrizione: nome dell'evento Facebook.
 - + `location_obj` : `GeoPoint`
Descrizione: location dell'evento espressa in latitudine e longitudine.
- **Metodi:** N/A

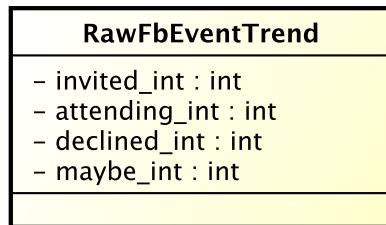


Figura 57: Classe - server::db::raw_data::fb::RawFbEventTrend

3.3.4.1.5 server::db::raw_data::fb::RawFbEventTrend

- **Descrizione:** classe che rappresenta il modello del trend di un evento su Facebook;
- **Utilizzo:** la classe viene utilizzata per memorizzare il numero di utenti invitati, partecipanti e non di un evento. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;
- **Classi ereditate:** server::db::raw_data::AbsRawData
- **Attributi:**
 - + `invited_int` : `int`
Descrizione: numero delle persone invitate all'evento Facebook.
 - + `attending_int` : `int`
Descrizione: numero persone partecipanti all'evento Facebook
 - + `declined_int` : `int`
Descrizione: numero persone che hanno rifiutato la partecipazione all'evento Facebook.
 - + `maybe_int` : `int`
Descrizione: numero persone incerte se partecipare all'evento Facebook.
- **Metodi:** N/A

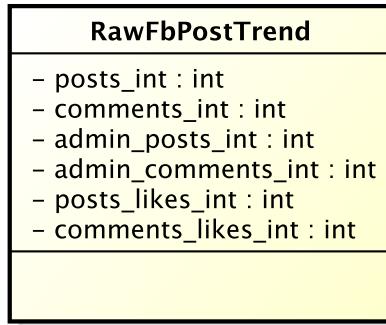


Figura 58: Classe - server::db::raw_data::fb::RawFbPostTrend

3.3.4.1.6 server::db::raw_data::fb::RawFbPostTrend

- **Descrizione:** classe che rappresenta il modello del trend dei post di una pagina o di un evento su Facebook;
- **Utilizzo:** viene utilizzata per memorizzare i dati relativi al trend dei post di una pagina o un pagina o evento Facebook. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;
- **Attributi:**

– + `posts_int` : int

Descrizione: numero dei post presenti nella pagina o nell'evento.

– + `comments_int` : int

Descrizione: numero totale di commenti ai post della pagina o dell'evento.

– + `admin_posts_int` : int

Descrizione: numero dei post effettuati esclusivamente da una pagina Facebook e non da terzi.

– + `admin_comments_int` : int

Descrizione: numero dei commenti effettuati esclusivamente da una pagina Facebook e non da terzi.

– + `posts_like_int` : int

Descrizione: numero totale dei likes ai post di una pagina o evento Facebook.

– + `comments_like_int` : int

Descrizione: numero totale ai likes dei commenti presenti nei post di una pagina o evento Facebook.

- **Metodi:** N/A

3.3.5 server::db::raw_data::tw

- **Descrizione:** è il package_G contenente le classi che definiscono i modelli dei dati grezzi relativi a Twitter;
- **Padre:** server::db::raw_data
- **Interazione con altri componenti:**
 - server::db

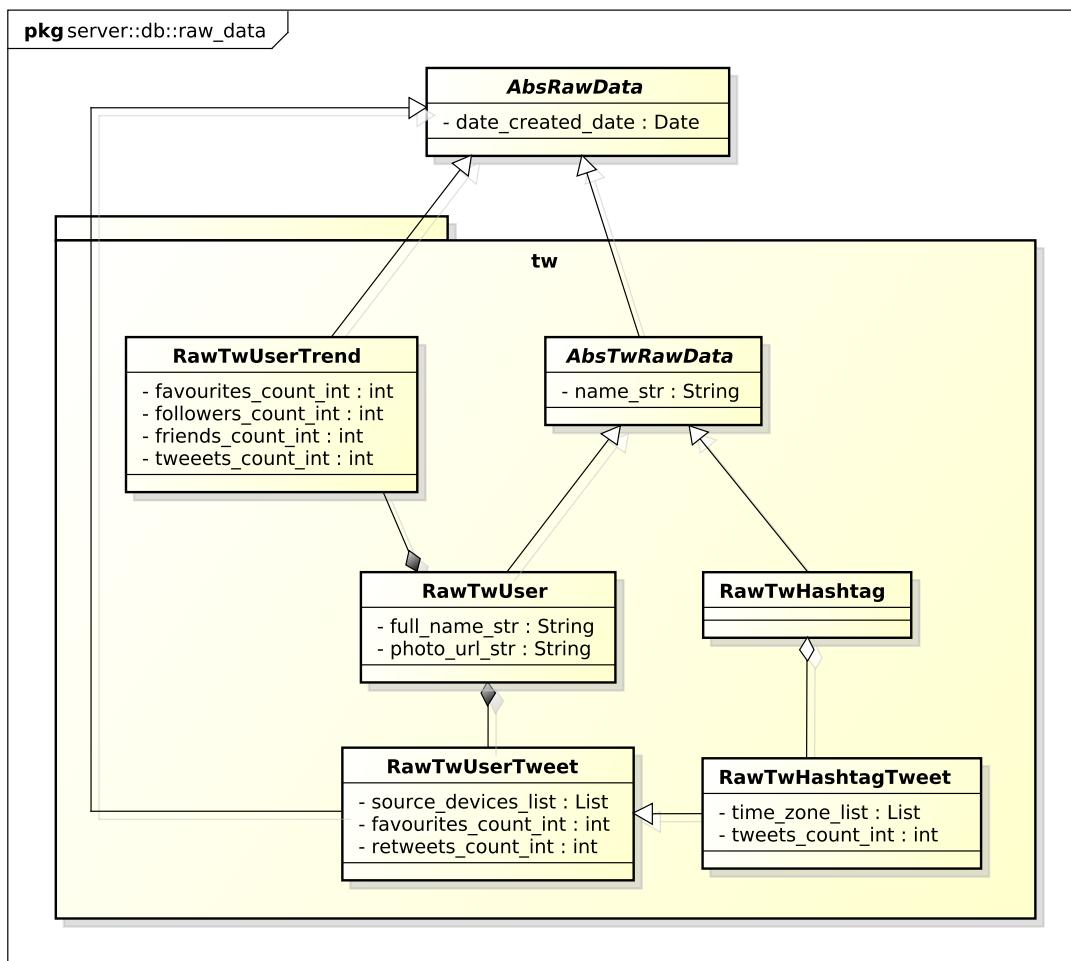


Figura 59: Package - server::db::raw_data::tw

3.3.5.1 Classi

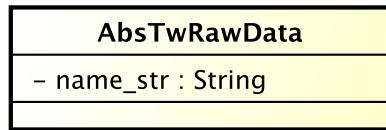


Figura 60: Classe - server::db::raw_data::tw::AbsTwRawData

3.3.5.1.1 server::db::raw_data::tw::AbsTwRawData

- **Descrizione:** classe astratta che definisce il modello dei dati grezzi relativi a Twitter;
- **Utilizzo:** la classe contiene l'id fornito dall'utente il quale permette di identificare univocamente la risorsa nel social media;
- **Classi ereditate:** server::db::raw_data::AbsRawData
- **Attributi:**
 - + `name_str` : String

Descrizione: nome identificativo della pagina o dell'hashtag Twitter.
- **Metodi:** N/A

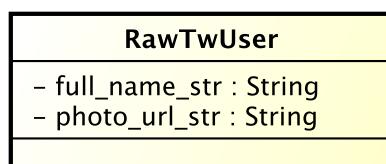


Figura 61: Classe - server::db::raw_data::tw::RawTwUser

3.3.5.1.2 server::db::raw_data::tw::RawTwUser

- **Descrizione:** classe che definisce il modello dei dati di un utente Twitter;
- **Utilizzo:** la classe viene utilizzata per fornire una descrizione completa dell'utente Twitter. Vengono forniti metodi automatici per il conteggio dei parametri che verranno utilizzati per seguire un trend;;
- **Classi ereditate:** server::db::raw_data::AbsTwRawData
- **Relazioni con altre classi:**
 - server::db::raw_data::tw::RawTwUserTrend
 - server::db::raw_data::tw::RawTwUserTweet
- **Attributi:**
 - + `full_name_str` : String

Descrizione: nome completo dell'utente Twitter.

 - + `photo_url_str` : String

Descrizione: indirizzo url della foto profilo dell'utente Twitter.
- **Metodi:** N/A

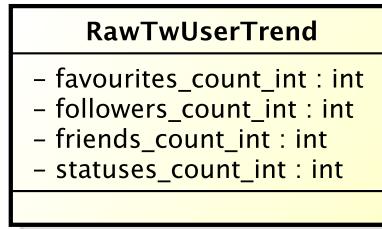


Figura 62: Classe - server::db::raw_data::tw::RawTwUserTrend

3.3.5.1.3 server::db::raw_data::tw::RawTwUserTrend

- **Descrizione:** classe che definisce il modello dei dati del trend di un utente Twitter;
- **Utilizzo:** la classe viene utilizzata per memorizzare il numero di favoriti, di followers, di friends e statuses di un determinato utente Twitter. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;
- **Classi ereditate:** server::db::raw_data::AbsTwRawData
- **Attributi:**
 - + favourites_count_int : int
Descrizione: numero totale dei preferiti assegnati dall'utente.
 - + followers_count_int : int
Descrizione: numero dei followers di un utente Twitter.
 - + friends_count_int : int
Descrizione: numero dei following di un utente Twitter.
 - + tweets_count_int : int
Descrizione: numero di tweets di un utente Twitter.
- **Metodi:** N/A

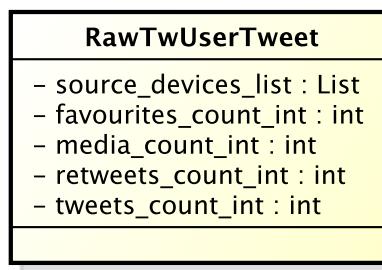


Figura 63: Classe - server::db::raw_data::tw::RawTwUserTweet

3.3.5.1.4 server::db::raw_data::tw::RawTwUserTweet

- **Descrizione:** classe che definisce il modello dei dati del trend dei tweet relativi ad un utente Twitter;
- **Utilizzo:** la classe viene utilizzata per fornire una descrizione dettagliata di un tweet creato da un utente specifico su Twitter;

- **Classi ereditate:** server::db::raw_data::AbsRawData
 - **Attributi:**
 - `+ source_devices_list : List`
Descrizione: lista composta da quantità e tipo dispositivi utilizzati in relazione ai tweet di un utente Twitter.
 - `+ favourites_count_int : int`
Descrizione: totale dei preferiti aggiunti ai tweet di un utente.
 - `+ retweets_count_int : int`
Descrizione: numero totale dei retweet ai tweets di utente Twitter.
 - **Metodi:** N/A
- 3.3.5.1.5 server::db::raw_data::tw::RawTwHashtag**
- **Descrizione:** classe che definisce il modello dei dati di un hashtag su Twitter;
 - **Utilizzo:** la classe viene utilizzata per fornire una descrizione minimale di un hashtag su Twitter. Sebbene tale classe non presenti metodi o attributi, risulta molto utile per distinguere un hashtag Twitter da un'altra metrica, effettuando un type checking;
 - **Classi ereditate:** server::db::raw_data::AbsTwRawData
 - **Relazioni con altre classi:**
 - server::db::raw_data::tw::RawTwHashtagTweet
 - **Attributi:** N/A
 - **Metodi:** N/A



Figura 64: Classe - server::db::raw_data::tw::RawTwHashtagTweet

- 3.3.5.1.6 server::db::raw_data::tw::RawTwHashtagTweet**
- **Descrizione:** classe che definisce il modello dei dati del trend dei tweet relativi ad un hashtag Twitter;
 - **Utilizzo:** la classe viene utilizzata per fornire una descrizione della locazione spaziale di un tweet relativo all'hashtag su Twitter. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;
 - **Classi ereditate:** server::db::raw_data::RawTwUserTweet
 - **Attributi:**
 - `+ time_zone_list : List`

Descrizione: contiene informazioni sulle timezone ricavate dai vari tweet contenenti un determinato hashtag Twitter e dalle occorrenze delle stesse.

- + `tweets_count_int : int`

Descrizione: totale dei tweet contenenti un determinato hashtag.

- **Metodi:** N/A

3.3.6 server::db::raw_data::ig

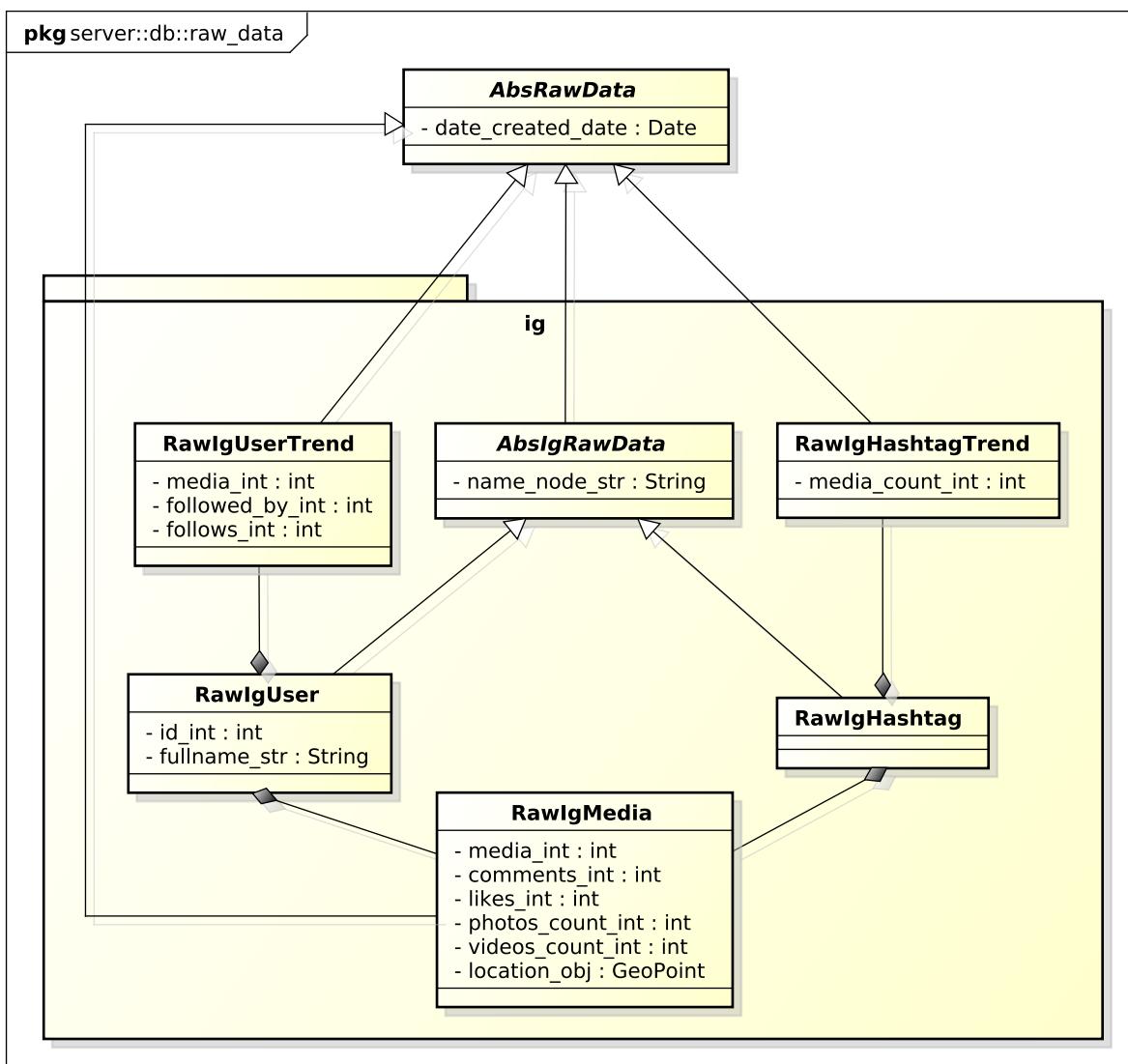


Figura 65: Package - server::db::raw_data::ig

- **Descrizione:** è il package_G contenente le classi che definiscono i modelli dei dati grezzi relativi a Instagram;
- **Padre:** server::db::raw_data
- **Interazione con altri componenti:**
 - server::db

3.3.6.1 Classi

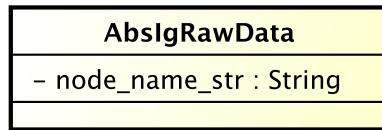


Figura 66: Classe - server::db::raw_data::ig::AbsIgRawData

3.3.6.1.1 server::db::raw_data::ig::AbsIgRawData

- **Descrizione:** classe astratta che definisce il modello dei dati grezzi relativi ad Instagram;
- **Utilizzo:** la classe contiene l'id fornito dall'utente il quale permette di identificare univocamente la risorsa nel social media;
- **Classi ereditate:** server::db::raw_data::AbsRawData
- **Attributi:**
 - + node_name_str : String

Descrizione: nome identificativo di un utente o di un hashtag Instagram.
- **Metodi:** N/A

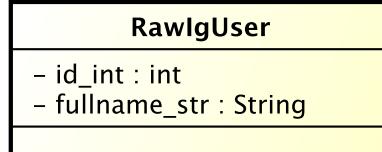


Figura 67: Classe - server::db::raw_data::ig::RawIgUser

3.3.6.1.2 server::db::raw_data::ig::RawIgUser

- **Descrizione:** classe che definisce il modello dei dati di un utente Instagram;
- **Utilizzo:** la classe viene utilizzata per memorizzare i dettagli di un utente Instagram;
- **Classi ereditate:** server::db::raw_data::AbsIgRawData
- **Relazioni con altre classi:**
 - server::db::raw_data::ig::RawIgUserTrend
 - server::db::raw_data::ig::RawIgMedia
- **Attributi:**
 - + id_int : int

Descrizione: numero identificativo di utente Instagram.

 - + fullname_str : String

Descrizione: nome completo di utente Instagram.
- **Metodi:** N/A

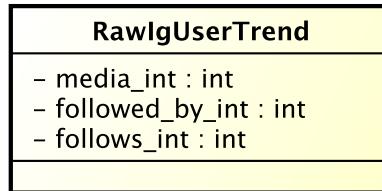


Figura 68: Classe - server::db::raw_data::ig::RawIgUserTrend

3.3.6.1.3 server::db::raw_data::ig::RawIgUserTrend

- **Descrizione:** classe che definisce il modello dei dati del trend di un utente Instagram;
- **Utilizzo:** la classe viene utilizzata per memorizzare il numero di media, di followed e follows di una determinata persona. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;
- **Classi ereditate:** server::db::raw_data::AbsRawData
- **Attributi:**
 - + `media_int` : int
Descrizione: numero di media caricati dall'utente su Instagram.
 - + `followed_by_int` : int
Descrizione: numero di followed dell'utente su Instagram.
 - + `follows_int` : int
Descrizione: numero di following dell'utente su Instagram.
- **Metodi:** N/A

3.3.6.1.4 server::db::raw_data::ig::RawIgHashtag

- **Descrizione:** classe che definisce il modello dei dati di un hashtag Instagram;
- **Utilizzo:** la classe viene utilizzata per fornire una descrizione minimale dell'hashtag su Instagram. Sebbene tale classe non fornisca metodi o campi dati, risulta molto utile per distinguere un hashtag Instagram da un'altra metrica, tramite type checking;
- **Classi ereditate:** server::db::raw_data::AbsIgRawData
- **Relazioni con altre classi:**
 - server::db::raw_data::ig::RawIgHashtagTrend
 - server::db::raw_data::ig::RawIgMedia
- **Attributi:** N/A
- **Metodi:** N/A



Figura 69: Classe - server::db::raw_data::ig::RawIgHashtagTrend

3.3.6.1.5 server::db::raw_data::ig::RawIgHashtagTrend

- **Descrizione:** classe che definisce il modello dei dati del trend di un hashtag Instagram;
- **Utilizzo:** la classe viene utilizzata per memorizzare il numero di media caricati di un determinato hashtag. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;
- **Classi ereditate:** server::db::raw_data::AbsRawData
- **Attributi:**
 - + `media_count_int` : int
Descrizione: numero totale di media relativi ad un hashtag su Instagram.
- **Metodi:** N/A

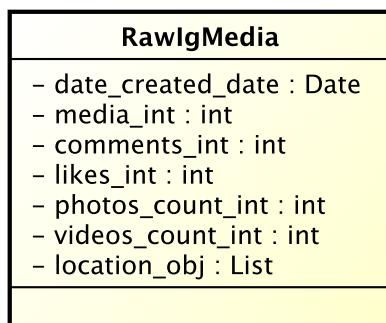


Figura 70: Classe - server::db::raw_data::ig::RawIgMedia

3.3.6.1.6 server::db::raw_data::ig::RawIgMedia

- **Descrizione:** classe che definisce il modello dei dati di un media relativo ad Instagram;
- **Utilizzo:** la classe viene utilizzata per fornire una descrizione dettagliata del trend dei media relativi ad un utente o un hashtag specifico su Instagram. Come per tutti gli oggetti di tipo trend, vengono ricavati i dati fino a 3 giorni prima della creazione dell'oggetto;
- **Classi ereditate:** server::db::raw_data::AbsRawData
- **Attributi:**
 - + `media_int` : int
Descrizione: numero di media ricavati.
 - + `comments_int` : int

Descrizione: numero totale dei commenti presenti in tutti i media ricavati.

- + `likes_int : int`

Descrizione: numero totale dei likes presenti in tutti i media ricavati.

- + `photos_count_int : int`

Descrizione: numero totale di media di tipo foto.

- + `videos_count_int : int`

Descrizione: numero totale di media di tipo video.

- + `location_obj : GeoPoint`

Descrizione: localizzazione tramite latitudine e longitudine di un media su Instagram.

- **Metodi:** N/A

3.3.7 server::db::app_data

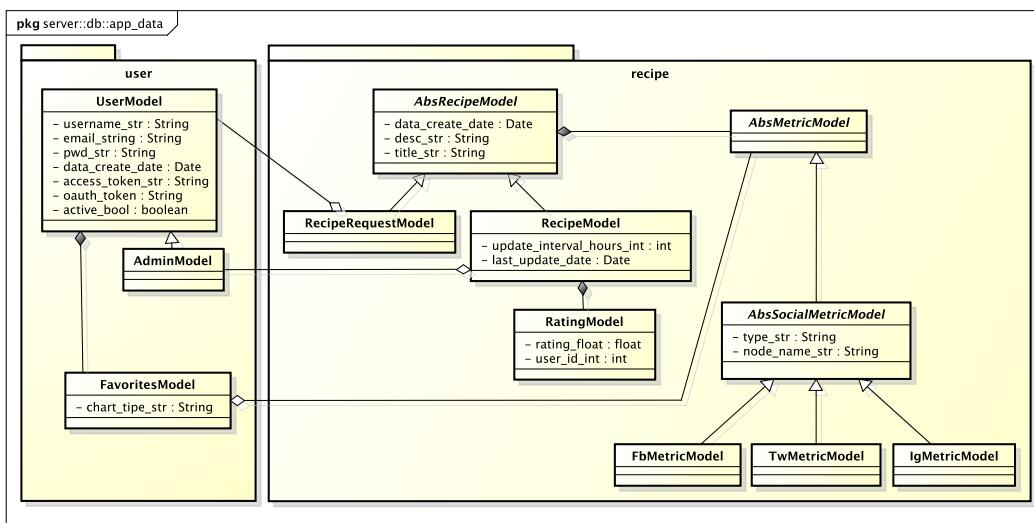


Figura 71: Package - server::db::app_data

- **Descrizione:** è il package_G che contiene la definizione dei modelli degli utenti registrati e le loro preferenze. Contiene inoltre il modello delle Recipe_G che l'amministratore decide di creare;

- **Padre:** server::db

- **Package_G contenuti**

- server::db::app_data::user
- server::db::app_data::recipe_G

3.3.8 server::db::app_data::user

- **Descrizione:** è il package_G che contiene la definizione dei modelli degli utenti registrati e le loro preferenze;

- **Padre:** server::db::app_data
- **Interazione con altri componenti:**
 - server::db::app_data::recipe_G

3.3.8.1 Classi

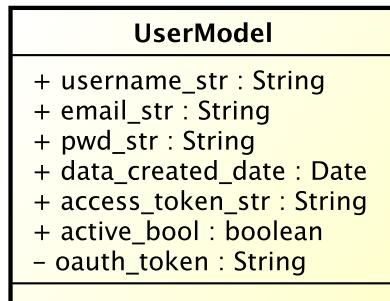


Figura 72: Classe - server::db::app_data::user::UserModel

3.3.8.1.1 server::db::app_data::user::UserModel

- **Descrizione:** classe che definisce il modello dei dati degli utenti all'interno della base di dati;
- **Utilizzo:** la classe utilizzata per aggiungere, modificare o eliminare un utente dal database_G;
- **Relazioni con altre classi:**
 - server::db::app_data::user::FavouritesModel
 - server::db::app_data::user::AdminModel
 - server::db::app_data::recipe_G::RecipeRequestModel
- **Attributi:**
 - `+ username_str : String`
Descrizione: lo username dell'utente.
 - `+ email_str : String`
Descrizione: l'email dell'utente.
 - `+ pwd_str : String`
Descrizione: la password dell'utente.
 - `+ data_created_date : Date`
Descrizione: la data creazione account dell'utente.
 - `+ access_token_str : String`
Descrizione: l'access token utilizzato per i servizi REST_G.
 - `+ oauth_token : String`
Descrizione: il token utilizzato per il processo di autenticazione dell'utente.
 - `+ active_bool : boolean`
Descrizione: valore che indica se l'utente attivo nel sistema.
- **Metodi:** N/A

3.3.8.1.2 server::db::app_data::user::AdminModel

- **Descrizione:** classe che definisce il modello dei dati degli utenti amministratori all'interno della base di dati;
- **Utilizzo:** la classe specializza l'utente amministratore. Viene utilizzata esclusivamente per distinguere un amministratore da un utente normale tramite type checking;
- **Classi ereditate:** server::db::app_data::user::UserModel;
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::RecipeModel
- **Attributi:** N/A
- **Metodi:** N/A

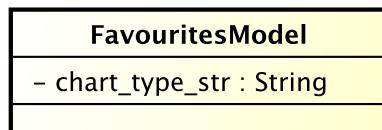


Figura 73: Classe - server::db::app_data::user::FavouritesModel

3.3.8.1.3 server::db::app_data::user::FavouritesModel

- **Descrizione:** classe che definisce il modello dei dati relativo ai preferiti dell'utente;
- **Utilizzo:** la classe viene utilizzata per memorizzare e ricavare le View_G preferite relative ad un utente.
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::AbsMetricModel
- **Attributi:**
 - `chart_type_str : String`

Descrizione: tipo di grafico scelto dall'utente e salvato nei preferiti.
- **Metodi:** N/A

3.3.9 server::db::app_data::recipe

- **Descrizione:** è il package_G che contiene la definizione dei modelli delle Recipe_G che l'amministratore decide di creare;
- **Padre:** server::db::app_data
- **Interazione con altri componenti:**
 - server::db::app_data::user

3.3.9.1 Classi

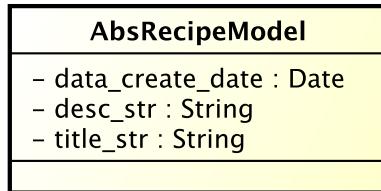


Figura 74: Classe - server::db::app_data::recipe::AbsRecipeModel

3.3.9.1.1 server::db::app_data::recipe::AbsRecipeModel

- **Descrizione:** classe astratta che rappresenta un modello comune per Recipe_G e richiesta di aggiunta Recipe;
- **Utilizzo:** la classe mantiene l'estensibilità per eventuali nuovi tipi di Recipe_G;
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::RecipeModel
 - server::db::app_data::recipe_G::RecipeRequestModel
 - server::db::app_data::recipe_G::AbsMetricModel
- **Attributi:**
 - `data_create_date` : Date
Descrizione: data di creazione della Recipe_G.
 - `desc_str` : String
Descrizione: descrizione dettagliata dell Recipe_G.
 - `title_str` : String
Descrizione: titolo della Recipe_G.
- **Metodi:** N/A

3.3.9.1.2 server::db::app_data::recipe::RecipeRequestModel

- **Descrizione:** classe che definisce il modello dei dati per la richiesta di aggiunta Recipe_G;
- **Utilizzo:** la classe specializza la richiesta identificandone l'utente tramite il rapporto parent-child delle classi di Google Datastore;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsRecipeModel
- **Attributi:** N/A
- **Metodi:** N/A

3.3.9.1.3 server::db::app_data::recipe::RecipeModel

- **Descrizione:** classe che definisce il modello dei dati relativo ad una Recipe_G;
- **Utilizzo:** la classe specializza la ricetta. Vengono forniti i campi dati per il possibile incremento temporale dei dati;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsRecipeModel

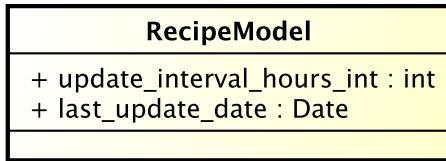


Figura 75: Classe - server::db::app_data::recipe::RecipeModel

- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::RatingModel
- **Attributi:**
 - + `update_interval_hours_int` : int
Descrizione: intervallo di tempo per la schedulazione dell'update della Recipe_G.
 - + `last_update_date` : Date
Descrizione: data dell'ultimo processo di update Recipe_G.
- **Metodi:** N/A

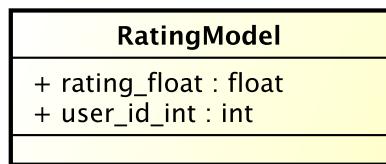


Figura 76: Classe - server::db::app_data::recipe::RatingModel

3.3.9.1.4 server::db::app_data::recipe::RatingModel

- **Descrizione:** classe che rappresenta il modello del rating di una Recipe_G;
- **Utilizzo:** viene utilizzata per risalire al voto di ogni utente per una determinata Recipe_G;
- **Attributi:**
 - + `rating_float` : float
Descrizione: punteggio gradimento relativo alla Recipe_G.
 - + `user_id_int` : int
Descrizione: identificativo dell'utente votante.
- **Metodi:** N/A

3.3.9.1.5 server::db::app_data::recipe::AbsMetricModel

- **Descrizione:** classe che definisce il modello dei dati di una metrica_G contenuta in una Ricetta;
- **Utilizzo:** la classe rappresenta il modello dei dati di una metrica_G generale;
- **Relazioni con altre classi:**

- server::db::app_data::recipe_G::AbsSocialMetricModel
- **Attributi:** N/A
- **Metodi:** N/A

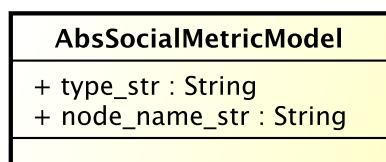


Figura 77: Classe - server::db::app_data::recipe::AbsSocialMetricModel

3.3.9.1.6 server::db::app_data::recipe::AbsSocialMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative ai social media;
- **Utilizzo:** la classe rappresenta il modello di una metrica_G relativa ad un social network fornendo il tipo di metrica e il suo identificativo;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsMetricModel
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::FbMetricModel
 - server::db::app_data::recipe_G::IgMetricModel
 - server::db::app_data::recipe_G::TwMetricModel
- **Attributi:**
 - `+ type_str : String`
 - Descrizione:** tipo della metrica_G (pagina, evento, hashtag).
 - `+ node_name_str : String`
 - Descrizione:** nome identificativo della metrica_G.
- **Metodi:** N/A

3.3.9.1.7 server::db::app_data::recipe::FbMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative a Facebook;
- **Utilizzo:** viene utilizzata esclusivamente per offrire una distinzione di tipo per le metriche relative a Facebook rispetto ai restanti social network;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsSocialMetricModel;
- **Attributi:** N/A
- **Metodi:** N/A

3.3.9.1.8 server::db::app_data::recipe::IgMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative a Instagram;
- **Utilizzo:** viene utilizzata esclusivamente per offrire una distinzione di tipo per le metriche relative a Instagram rispetto ai restanti social network;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsSocialMetricModel;
- **Attributi:** N/A
- **Metodi:** N/A

3.3.9.1.9 server::db::app_data::recipe::TwMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative a Twitter;
- **Utilizzo:** viene utilizzata esclusivamente per offrire una distinzione di tipo per le metriche relative a Twitter rispetto ai restanti social network;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsSocialMetricModel;
- **Attributi:** N/A
- **Metodi:** N/A

3.3.10 server::processor

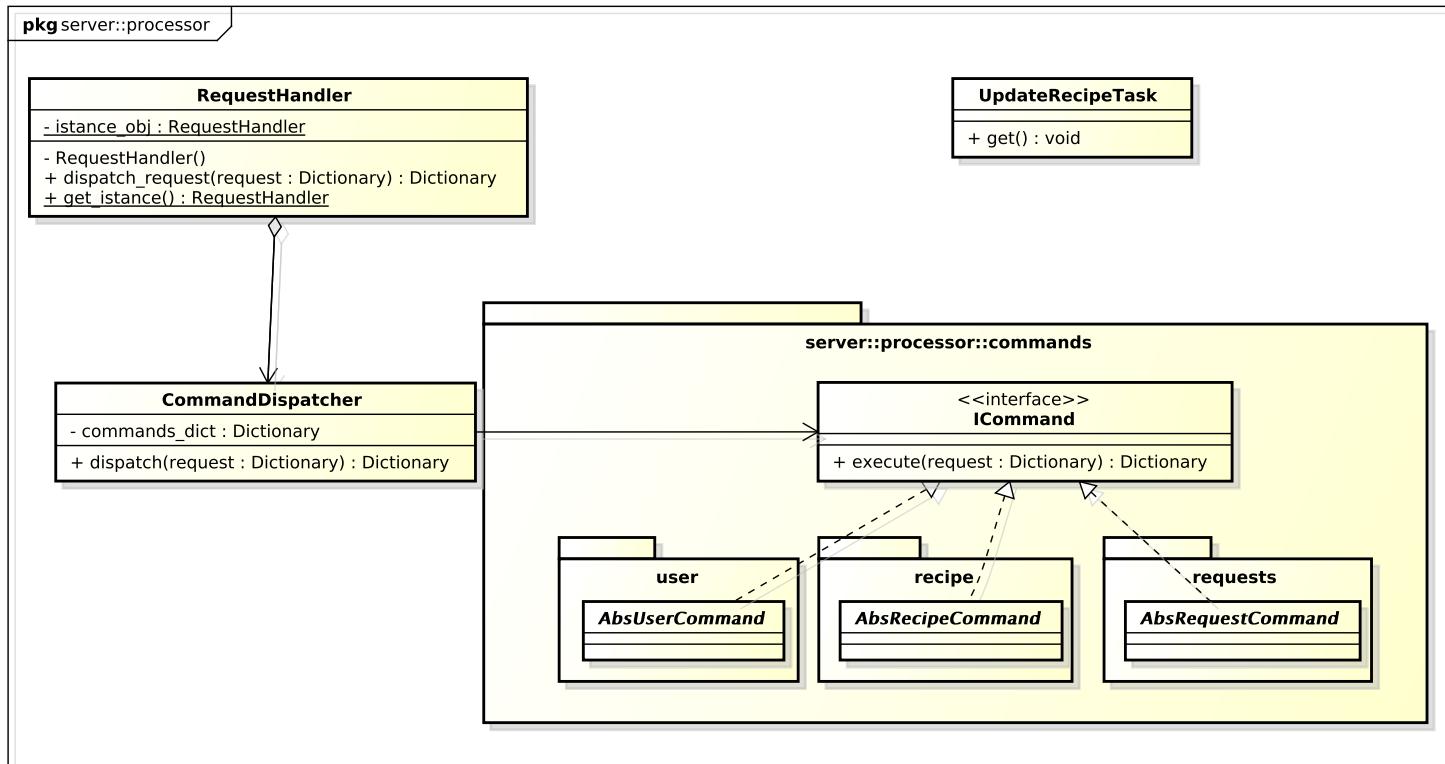


Figura 78: Package - `server::processor`

- **Descrizione:** è il package_G che contiene le componenti che gestiscono le richieste in arrivo dal client grazie ai servizi REST_G definiti nel package `server::endpoints::apiG`;
- **Padre:** server
- **Package_G contenuti:** `server::processorG::commands`
- **Interazione con altri componenti:**
 - `server::minerG`
 - `server::endpoints`
 - `server::db`

3.3.10.1 Classi



Figura 79: Classe - `server::processor::RequestHandler`

3.3.10.1.1 `server::processor::RequestHandler`

- **Descrizione:** è la classe che implementa il pattern Front Controller e si occupa di raccogliere le richieste provenienti dal client. Implementa inoltre il pattern Singleton, perciò un'unica istanza di tale classe vivrà nel sistema;
- **Utilizzo:** viene invocata dalle classi presenti nel package_G `server::endpoints::apiG` in seguito ad una chiamata ai servizi REST_G offerti dal sistema e trasferisce la richiesta alla classe `CommandDispatcher` che si occuperà di invocare il relativo comando per soddisfare tale richiesta;
- **Relazioni con altre classi:**
 - `server::processorG::CommandDispatcher`
- **Attributi:**
 - `- static instance_obj : RequestHandler`

Descrizione: rappresenta la singola istanza della classe `RequestHandler`.
- **Metodi:**
 - `- request_handler()`

Descrizione: costruttore privato del singleton `RequestHandler`.

 - `+ dispatch_request(request_name: String, request_args: Dictionary) : Dictionary`

Descrizione: metodo che si occupa di far confluire la richiesta in arrivo alla classe `CommandDispatcher`. I dati della richiesta vengono rappresentati come un dizionario chiave-valore, come quelli della risposta che saranno ritornati al chiamante.

 - `+ static get_instance() : RequestHandler`

Descrizione: ritorna la singola istanza della classe `RequestHandler` o ne crea una se tale classe non risulta ancora istanziata.

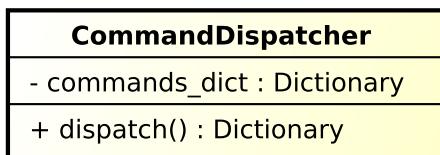


Figura 80: Classe - `server::processor::CommandDispatcher`

3.3.10.1.2 `server::processor::CommandDispatcher`

- **Descrizione:** classe che implementa il pattern Command e si occupa di far confluire una determinata richiesta al relativo comando contenente la logica per soddisfarla;
- **Utilizzo:** contiene un dizionario che mappa tutte le tipologie di richieste al relativo comando che sarà invocato tramite il metodo `dispatch()`. Si occupa inoltre di ritornare al Front Controller un dizionario contenente un'eventuale risposta per il client;
- **Relazioni con altre classi:**

- server::processor_G::commands:: ICommand

- **Attributi:**

- – `commands_dict : Dictionary`

Descrizione: contiene una mappa che collega ogni tipologia di richiesta ad un determinato comando. Questo dizionario ci permette di avere una lista completa e ordinata di associazioni tra i metodi delle API_G e i comandi che devono utilizzare in modo da facilitarci la gestione delle chiamate ai comandi. Una richiesta è rappresentata da una specifica stringa.

- **Metodi:**

- + `dispatch(request_name: String, request: Dictionary) : Dictionary`

Descrizione: metodo che si occupa di far confluire un determinata richiesta (ed i dati associati mappati in un dizionario) ad un determinato comando grazie all'attributo `commands_dict`. Ritorna la risposta alla chiamata tramite un dizionario.

3.3.11 server::processor::commands

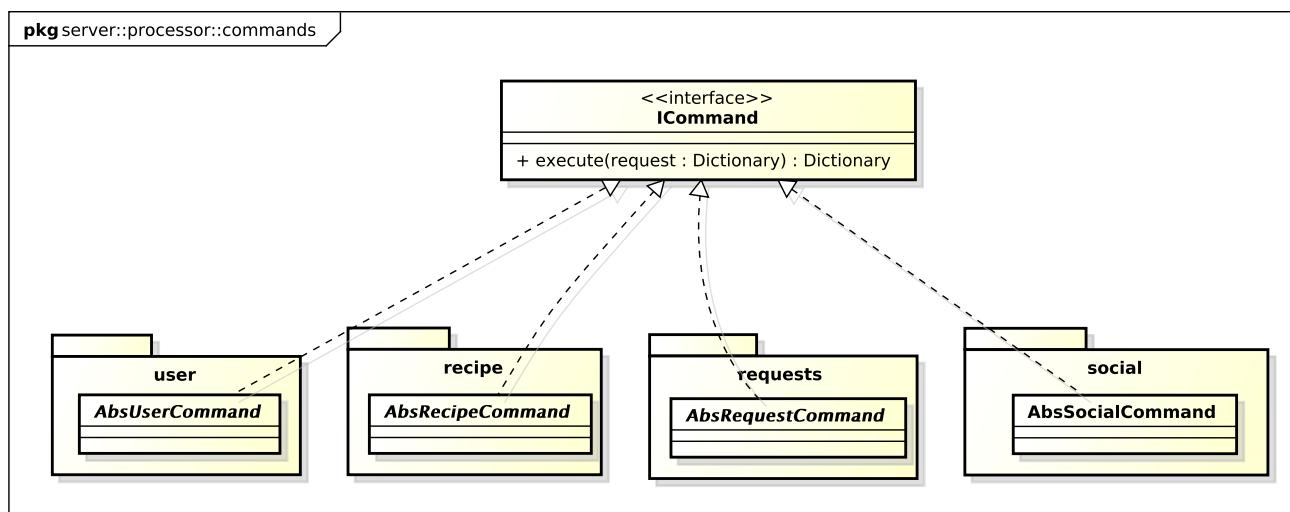


Figura 81: Package - server::processor::commands

- **Descrizione:** contiene le classi ed i package_G che definiscono i diversi comandi contenenti la logica necessaria a soddisfare le varie richieste in arrivo dal client;
- **Padre:** server::processor_G
- **Package_G contenuti:**
 - server::processor_G::commands::recipe_G
 - server::processor_G::commands::requests
 - server::processor_G::commands::user
 - server::processor_G::commands::social
- **Interazione con altri componenti:**
 - server::db

3.3.11.1 Classi

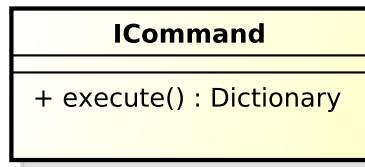


Figura 82: Classe - server::processor::ICommand

3.3.11.1.1 server::processor::commands::ICommand

- **Descrizione:** rappresenta un'interfaccia comune per tutti i comandi presenti nel package `G server::commands` e figli;
- **Utilizzo:** espone un metodo `execute()`, il quale sarà implementato da tutti i comandi concreti;
- **Attributi:** N/A
- **Metodi:**

– `+ execute(request: Dictionary) : Dictionary`

Descrizione: metodo astratto che definisce un contratto per tutte le classi che implementano `ICommand`. Accetta un parametro di tipo `Dictionary`, contente i dati della richiesta, e ritorna un oggetto dello stesso tipo contente i dati da ritornare alla chiamata (se necessari) ed informazioni successo o meno di esecuzione del comando.

3.3.12 server::processor::commands::user

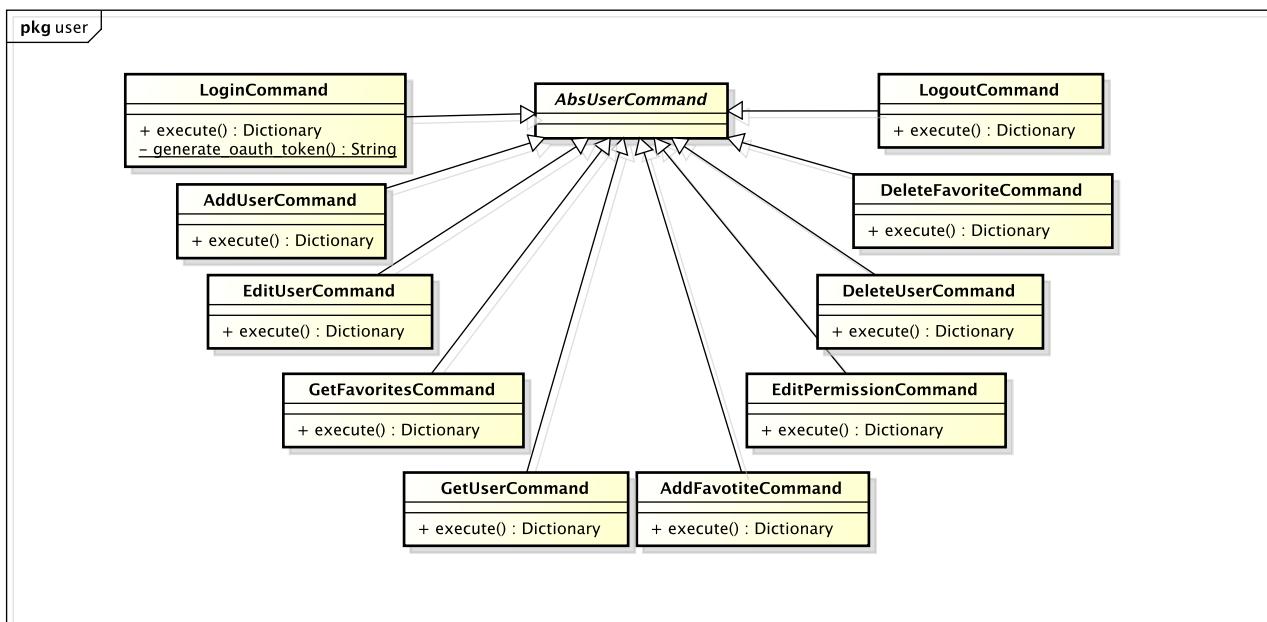


Figura 83: Package - server::processor::commands::user

- **Descrizione:** contiene tutti i comandi relativi alla gestione degli utenti;

- **Padre:** server::processor_G::commands;
- **Interazione con altri componenti:**
 - server::db

3.3.12.1 Classi

3.3.12.1.1 server::processor::commands::user::AbsUserCommand

- **Descrizione:** rappresenta una classe comune a tutti i comandi relativi alla gestione degli utenti;
- **Utilizzo:** è stata rappresentata come classe in quanto potrà contenere uno o più metodi di utilità comuni a tutti i comandi relativi alla gestione degli utenti;
- **Relazioni con altre classi:**
 - server::processor_G::commands:: ICommand
- **Attributi:** N/A
- **Metodi:** N/A

3.3.12.1.2 server::processor::commands::user::LoginCommand

- **Descrizione:** definisce la logica per effettuare il login al sistema;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per scambiare le informazioni di accesso al sistema;
- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: esegue l'accesso al sistema.

3.3.12.1.3 server::processor::commands::user::LogoutCommand

- **Descrizione:** definisce la logica per effettuare la disconnessione dal sistema;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per effettuare l'operazione di logout dal sistema;
- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: esegue la disconnessione dal sistema.

3.3.12.1.4 server::processor::commands::user::AddUserCommand

- **Descrizione:** definisce la logica per aggiungere un nuovo utente al database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per aggiungere un nuovo utente al database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: aggiunge un nuovo utente al database_G.

3.3.12.1.5 server::processor::commands::user::GetUserCommand

- **Descrizione:** definisce la logica per ricavare un determinato utente dal database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ricavare un determinato utente dal database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: ritorna un determinato utente presente nel database_G.

3.3.12.1.6 server::processor::commands::user::DeleteUserCommand

- **Descrizione:** definisce la logica per rimuovere un determinato utente dal database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per rimuovere un determinato utente dal database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: elimina un determinato utente dal database_G.

3.3.12.1.7 server::processor::commands::user::EditUserCommand

- **Descrizione:** definisce la logica per modificare i dati un determinato utente dal database_G;

- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per modificare i dati un determinato utente dal database_G;

- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand

- **Attributi:** N/A

- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: modifica i dati associati ad un determinato utente presente nel database_G.

3.3.12.1.8 server::processor::commands::user::EditPermissionCommand

- **Descrizione:** definisce la logica per modificare i permessi di un determinato utente;

- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per modificare i permessi di un determinato utente;

- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand

- **Attributi:** N/A

- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: modifica i permessi di un determinato utente presente nel database_G.

3.3.12.1.9 server::processor::commands::user::GetFavouritesCommand

- **Descrizione:** definisce la logica per ottenere le View_G preferite di un determinato utente;

- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere le View_G preferite di un determinato utente;

- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand

- **Attributi:** N/A

- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: ritorna i preferiti associati ad un determinato utente.

3.3.12.1.10 server::processor::commands::user::DeleteFavouriteCommand

- **Descrizione:** definisce la logica per rimuovere un determinato preferito da un utente specifico;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per rimuovere un determinato preferito da un utente specifico;
- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand
- **Attributi:** N/A
- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: rimuove dal database_G un determinato preferito associato ad un utente.

3.3.12.1.11 server::processor::commands::user::AddFavouriteCommand

- **Descrizione:** definisce la logica per aggiungere un preferito ad un determinato utente;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per aggiungere un preferito ad un determinato utente;
- **Relazioni con altre classi:**
 - server::processor_G::commands::user::AbsUserCommand
- **Attributi:** N/A
- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: aggiunge al database_G un preferito associato ad un determinato utente.

3.3.13 server::processor::commands::recipe

- **Descrizione:** contiene tutti i comandi relativi alla gestione delle Recipe_G;
- **Padre:** server::commands
- **Interazione con altri componenti:**
 - server::db

3.3.13.1 Classi

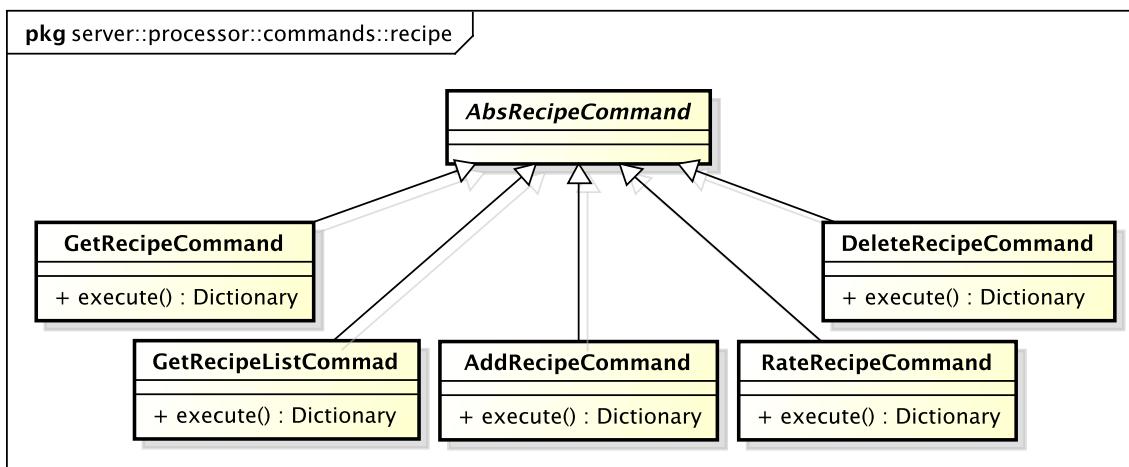


Figura 84: Package - server::processor::commands::recipe

3.3.13.1.1 server::processor::commands::recipe::AbsRecipeCommand

- **Descrizione:** rappresenta una classe comune a tutti i comandi relativi alla gestione delle Recipe_G;
- **Utilizzo:** è stata rappresentata come classe in quanto potrà contenere uno o più metodi di utilità comuni a tutti i comandi relativi alla gestione delle Recipe_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands:: ICommand;
- **Attributi:** N/A
- **Metodi:** N/A

3.3.13.1.2 server::processor::commands::recipe::GetRecipeCommand

- **Descrizione:** definisce la logica per ottenere una determinata Recipe_G e la lista delle metriche associate ad essa;
 - **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere una determinata Recipe_G e la lista delle metriche associate ad essa;
 - **Relazioni con altre classi:**
 - server::processor_G::commands::recipe_G::AbsRecipeCommand
 - **Attributi:** N/A
 - **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`
- Descrizione:** restituisce i dati associati ad una determinata Recipe_G presente nel database_G.

3.3.13.1.3 server::processor::commands::recipe::GetRecipeListCommand

- **Descrizione:** definisce la logica per ottenere la lista delle Recipe_G presenti nel sistema;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere la lista delle Recipe_G presenti nel sistema;
- **Relazioni con altre classi:**
 - server::processor_G::commands::recipe_G::AbsRecipeCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce la lista delle Recipe_G presenti nel database_G.

3.3.13.1.4 server::processor::commands::recipe::AddRecipeCommand

- **Descrizione:** definisce la logica per aggiungere una nuova Recipe_G al database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per aggiungere una nuova Recipe_G al database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::recipe_G::AbsRecipeCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: aggiunge una nuova Recipe_G al database_G.

3.3.13.1.5 server::processor::commands::recipe::DeleteRecipeCommand

- **Descrizione:** definisce la logica per rimuovere una determinata Recipe_G dal database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per rimuovere una determinata Recipe_G dal database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::recipe_G::AbsRecipeCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: rimuove una determinata Recipe_G dal database_G.

3.3.13.1.6 server::processor::commands::recipe::RateRecipeCommand

- **Descrizione:** definisce la logica per aggiungere e modificare il rating ad un determinata Recipe_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per aggiungere e modificare il rating ad un determinata Recipe_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::recipe_G::AbsRecipeCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: aggiunge o modifica il rating di una determinata Recipe_G.

3.3.14 server::processor::commands::requests

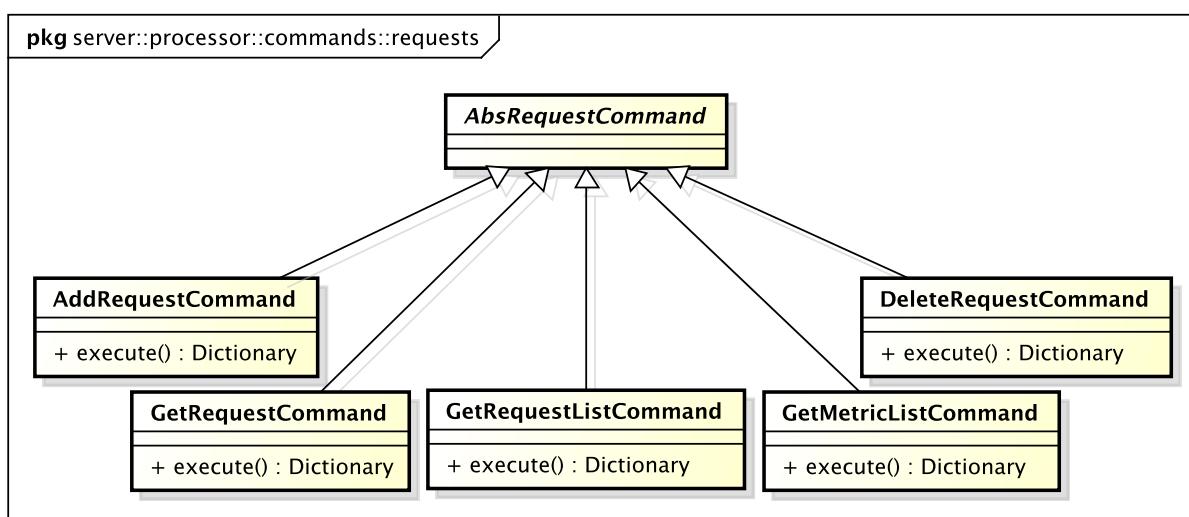


Figura 85: Package - server::processor::commands::requests

- **Descrizione:** contiene tutti i comandi relativi alla gestione delle richieste di aggiunta Recipe_G;
- **Padre:** server::processor_G::commands;
- **Interazione con altri componenti:**
 - server::db

3.3.14.1 Classi

3.3.14.1.1 server::processor::commands::requests::AbsRequestCommand

- **Descrizione:** rappresenta una classe comune a tutti i comandi relativi alla gestione delle richieste di aggiunta Recipe_G;
- **Utilizzo:** è stata rappresentata come classe in quanto potrà contenere uno o più metodi di utilità comuni a tutti i comandi relativi alla gestione delle richieste di aggiunta Recipe_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands:: ICommand
- **Attributi:** N/A
- **Metodi:** N/A

3.3.14.1.2 server::processor::commands::requests::GetRequestCommand

- **Descrizione:** definisce la logica per ottenere un determinata richiesta di aggiunta Recipe_G dal database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere una determinata richiesta di aggiunta Recipe_G dal database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::requests::AbsRequestCommand
- **Attributi:** N/A
- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati ad una determinata richiesta di aggiunta Recipe_G presente nel database_G.

3.3.14.1.3 server::processor::commands::requests::AddRequestCommand

- **Descrizione:** definisce la logica per aggiungere una nuova richiesta di aggiunta Recipe_G al database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per aggiungere una nuova richiesta di aggiunta Recipe_G al database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::requests::AbsRequestCommand
- **Attributi:** N/A
- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: aggiunge una nuova richiesta di aggiunta Recipe_G al database_G.

3.3.14.1.4 server::processor::commands::requests::GetRequestListCommand

- **Descrizione:** definisce la logica per ottenere la lista delle richieste di aggiunta Recipe_G presenti nel database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere la lista delle richieste di aggiunta Recipe_G presenti nel database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::requests::AbsRequestCommand
- **Attributi:** N/A
- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati ad una determinata richiesta di aggiunta Recipe_G presente nel database_G.

3.3.14.1.5 server::processor::commands::requests::DeleteRequestCommand

- **Descrizione:** definisce la logica per rimuovere una determinata richiesta di aggiunta Recipe_G dal database_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per rimuovere una determinata richiesta di aggiunta Recipe_G dal database_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::requests::AbsRequestCommand
- **Attributi:** N/A
- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: rimuove una determinata richiesta di aggiunta Recipe_G presente nel database_G.

3.3.14.1.6 server::processor::commands::requests::GetMetricsListCommand

- **Descrizione:** definisce la logica per ottenere la lista delle metriche presenti in una determinata richiesta di aggiunta Recipe_G;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere la lista delle metriche presenti in una determinata richiesta di aggiunta Recipe_G;
- **Relazioni con altre classi:**
 - server::processor_G::commands::requests::AbsRequestCommand
- **Attributi:** N/A
- **Metodi:**
 - + `execute(request: Dictionary) : Dictionary`

Descrizione: restituisce la lista delle metriche associate ad una determinata Recipe_G.

3.3.15 server::processor::commands::social

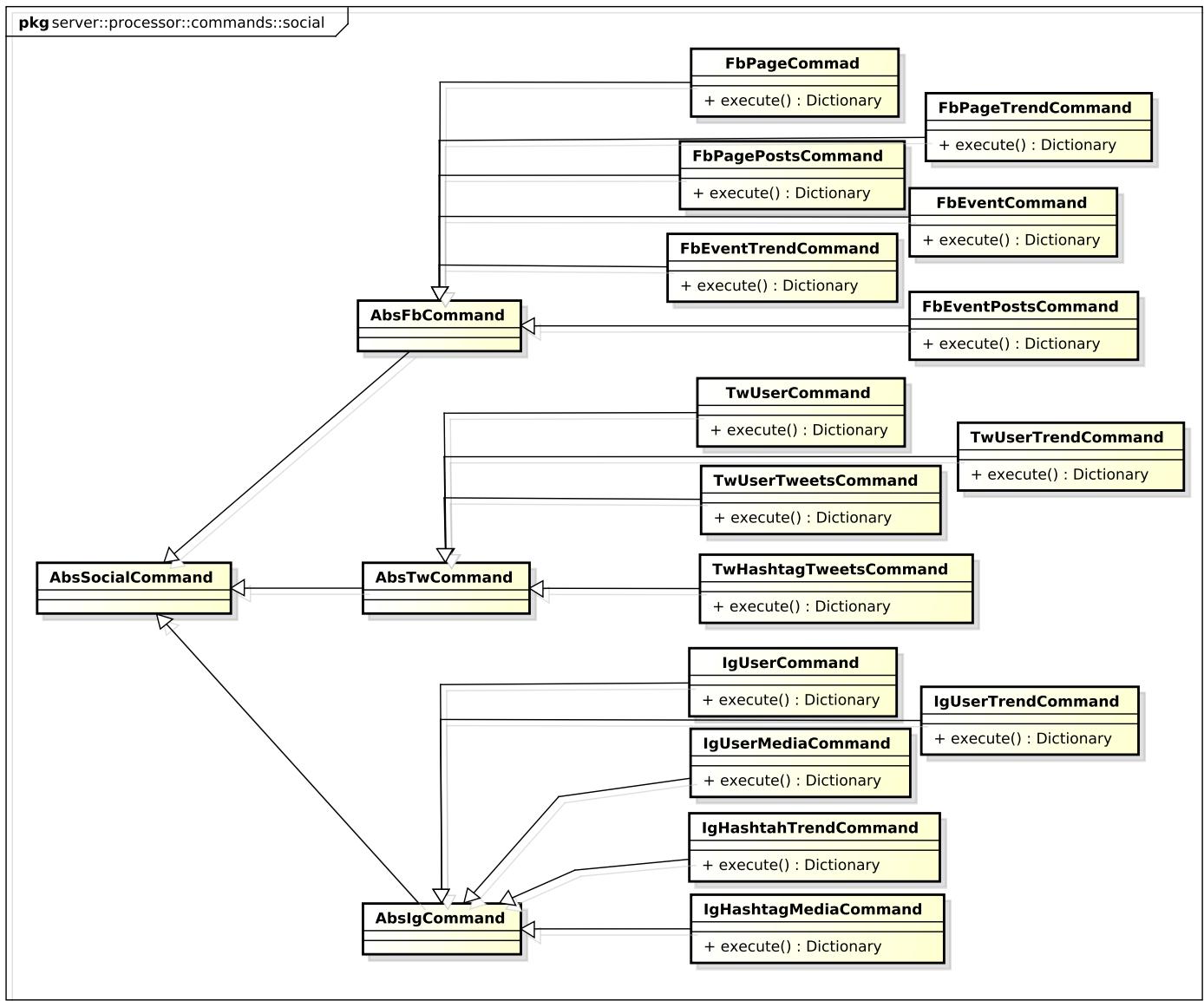


Figura 86: Package - server::processor::commands::social

- **Descrizione:** contiene tutti i comandi che per ottenere tutti i dati grezzi ricavati dai social network;
- **Padre:** server::processor_G::commands;
- **Interazione con altri componenti:**
 - server::db

3.3.15.1 Classi

3.3.15.1.1 server::processor::commands::social::AbsSocialCommand

- **Descrizione:** rappresenta una classe comune a tutti i comandi relativi alla gestione dei dati grezzi ricavati dai social network;

- **Utilizzo:** è stata rappresentata come classe in quanto potrà contenere uno o più metodi di utilità comuni a tutti i comandi relativi alla gestione dei dati grezzi ricavati dai social network;
- **Relazioni con altre classi:**
 - server::processor_G::commands:: ICommand
- **Attributi:** N/A
- **Metodi:** N/A

3.3.15.1.2 server::processor::commands::social::AbsFbCommand

- **Descrizione:** rappresenta una classe comune a tutti i comandi relativi alla gestione dei dati grezzi ricavati da Facebook;
- **Utilizzo:** è stata rappresentata come classe in quanto potrà contenere uno o più metodi di utilità comuni a tutti i comandi relativi alla gestione dei dati grezzi ricavati da Facebook;
- **Relazioni con altre classi:**
 - server::processor_G::commands::social::AbsSocialCommand
- **Attributi:** N/A
- **Metodi:** N/A

3.3.15.1.3 server::processor::commands::social::FbPageCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi ad una pagina Facebook;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi ad una pagina Facebook;
- **Relazioni con altre classi:**
 - server::processor_G::commands::social::AbsFbCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati ad una determinata pagina Facebook presente nel database_G.

3.3.15.1.4 server::processor::commands::social::FbPageTrendCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend di una pagina Facebook;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend di una pagina Facebook;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsFbCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend di una determinata pagina Facebook presente nel database_G.

3.3.15.1.5 server::processor::commands::social::FbPageEventCommand

- **Descrizione:** definisce la logica per ottenere i dati relativi a tutti gli eventi associati ad una pagina Facebook;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati relativi a tutti gli eventi associati ad una pagina Facebook;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsFbCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati a tutti gli eventi di una determinata pagina Facebook presente nel database_G.

3.3.15.1.6 server::processor::commands::social::FbEventCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi ad un evento Facebook;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi ad un evento Facebook;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsFbCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati ad un determinato evento Facebook presente nel database_G.

3.3.15.1.7 server::processor::commands::social::FbEventTrendCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend di un evento Facebook;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend di un evento Facebook;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsFbCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend di un determinato evento Facebook presente nel database_G.

3.3.15.1.8 server::processor::commands::social::FbEventPostCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend dei post di un evento Facebook;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend dei post di un evento Facebook;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsFbCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend dei post di un determinato evento Facebook presente nel database_G.

3.3.15.1.9 server::processor::commands::social::AbsTwCommand

- **Descrizione:** rappresenta una classe comune a tutti i comandi relativi alla gestione dei dati grezzi ricavati da Twitter;
- **Utilizzo:** è stata rappresentata come classe in quanto potrà contenere uno o più metodi di utilità comuni a tutti i comandi relativi alla gestione dei dati grezzi ricavati da Twitter;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsSocialCommand`
- **Attributi:** N/A
- **Metodi:** N/A

3.3.15.1.10 server::processor::commands::social::TwUserCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi ad un utente Twitter;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi ad un utente Twitter;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsTwCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati ad un determinato utente Twitter presente nel database_G.

3.3.15.1.11 server::processor::commands::social::TwUserTrendCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend di un utente Twitter;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend di un utente Twitter;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsTwCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend di un determinato utente Twitter presente nel database_G.

3.3.15.1.12 server::processor::commands::social::TwUserTweetsCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend dei tweet di un utente Twitter;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend dei tweet di un utente Twitter;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsTwCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend dei tweet di un determinato utente Twitter presente nel database_G.

3.3.15.1.13 server::processor::commands::social::TwHashtagTweetsCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi associati al trend dei tweet relativi ad un determinato hashtag;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi associati al trend dei tweet relativi ad un determinato hashtag;
- **Relazioni con altre classi:**
 - server::processor_G::commands::social::AbsTwCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend dei tweet relativi ad un determinato hashtag Twitter presente nel database_G.

3.3.15.1.14 server::processor::commands::social::AbsIgCommand

- **Descrizione:** rappresenta una classe comune a tutti i comandi relativi alla gestione dei dati grezzi ricavati da Instagram;
- **Utilizzo:** è stata rappresentata come classe in quanto potrà contenere uno o più metodi di utilità comuni a tutti i comandi relativi alla gestione dei dati grezzi ricavati da Instagram;
- **Relazioni con altre classi:**
 - server::processor_G::commands::social::AbsSocialCommand
- **Attributi:** N/A
- **Metodi:** N/A

3.3.15.1.15 server::processor::commands::social::IgUserCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi ad un utente Instagram;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi ad un utente Instagram;
- **Relazioni con altre classi:**
 - server::processor_G::commands::social::AbsIgCommand
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati ad un determinato utente Instagram presente nel database_G.

3.3.15.1.16 server::processor::commands::social::IgUserTrendCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend di un utente Instagram;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend di un utente Instagram;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsIgCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend di un determinato utente Instagram presente nel database_G.

3.3.15.1.17 server::processor::commands::social::IgUserMediaCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend dei media di un utente Instagram;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend dei media di un utente Instagram;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsIgCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend dei media di un determinato utente Instagram presente nel database_G.

3.3.15.1.18 server::processor::commands::social::IgHashtagTrendCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend di un hashtag di Instagram;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend di un hashtag di Instagram;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsIgCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`

Descrizione: restituisce i dati associati al trend relativo ad un determinato hashtag Instagram presente nel database_G.

3.3.15.1.19 server::processor::commands::social::IgHashtagMediaCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend dei media associati ad un hashtag di Instagram;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend dei media associati ad un hashtag di Instagram;
- **Relazioni con altre classi:**
 - `server::processorG::commands::social::AbsIgCommand`
- **Attributi:** N/A
- **Metodi:**
 - `+ execute(request: Dictionary) : Dictionary`
Descrizione: restituisce i dati associati al trend relativo ad un determinato hashtag Instagram presente nel database_G.

3.3.16 server::miner

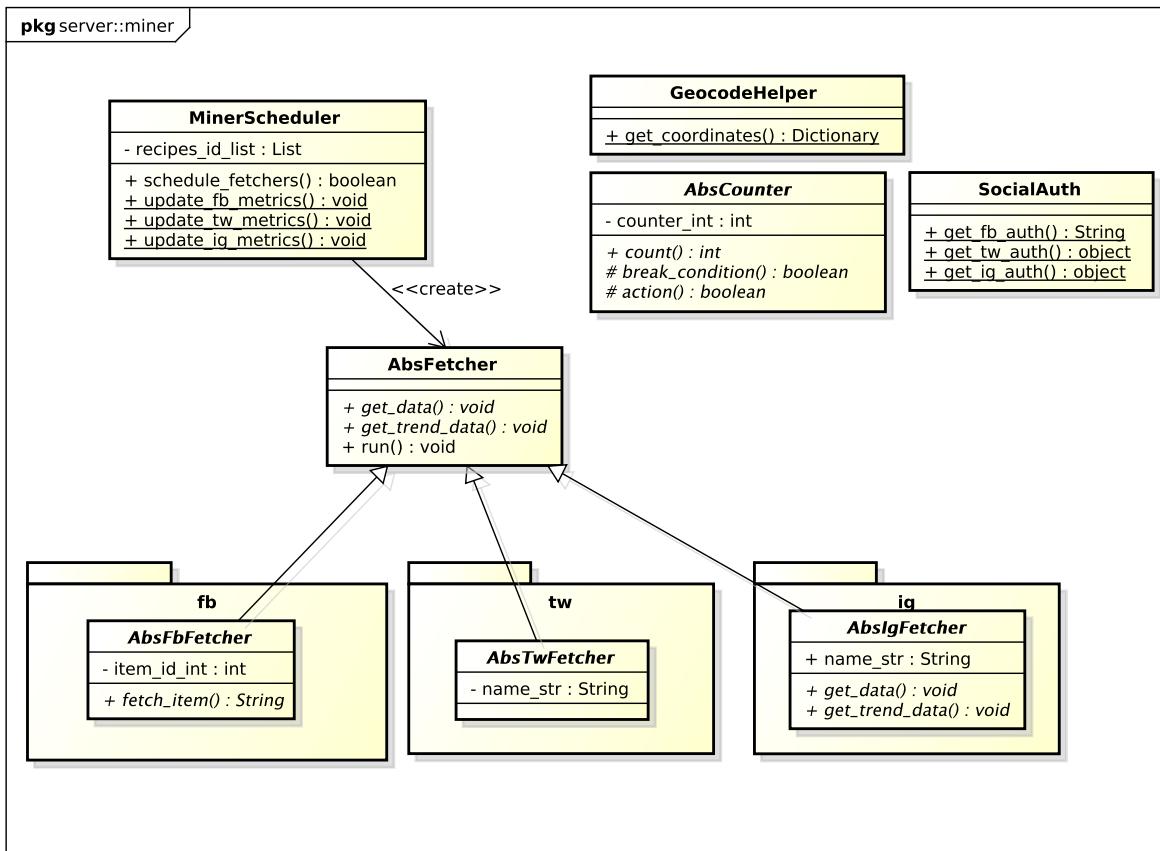


Figura 87: Package - server::miner

- **Descrizione:** è il package_G che contiene tutte le classi che includono i metodi per prelevare i dati grezzi dai vari social network e salvarli nel database_G;
- **Padre:** server
- **Package_G contenuti:**
 - server::miner_G::fb
 - server::miner_G::ig
 - server::miner_G::tw
- **Interazione con altri componenti:**
 - server::processor_G
 - server::db

3.3.16.1 Classi

3.3.16.1.1 server::miner::MinerScheduler

- **Descrizione:** classe che si occupa di creare i fetcher che preleveranno i dati per ogni metrica_G di ogni social network. La classe implementa metodi statici in quanto, nel caso si scelga di utilizzare le funzionalità di multi-threading di Google App Engine_G, questi non creerebbero conflitti interni;

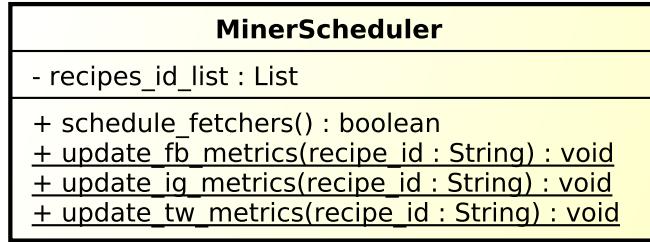


Figura 88: Classe - server::miner::MinerScheduler

- **Utilizzo:** contiene la lista degli id delle Recipe_G da aggiornare ed un metodo che inizializza e avvia i vari fetcher;
- **Relazioni con altre classi:**
 - server::miner_G::AbsFetcher
- **Attributi:**
 - `recipes_id_list : List`
Descrizione: lista di recipes da aggiornare;
- **Metodi:**
 - `+ __init__(recipeG_id_list : int) : void`
Descrizione: costruttore della classe di default. Inizializza la variabile recipe_G_id_list con la lista delle ricette da aggiornare;
 - `+ schedule_fetchers() : void`
Descrizione: avvia l'update delle recipes contenute nella lista passata;
 - `+ static update_fb_metrics(recipeG_id : String) : void`
Descrizione: il metodo controlla il tipo di recipe_G e avvia l'aggiornamento della recipe di Facebook;
 - `+ static update_ig_metrics(recipeG_id : String) : void`
Descrizione: il metodo controlla il tipo di recipe_G e avvia l'aggiornamento della recipe di Instagram;
 - `+ static update_tw_metrics(recipeG_id : String) : void`
Descrizione: il metodo controlla il tipo di recipe_G e avvia l'aggiornamento della recipe di Twitter;

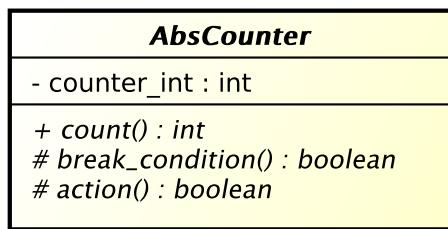


Figura 89: Classe - server::miner::AbsCounter

3.3.16.1.2 server::miner::AbsCounter

- **Descrizione:** classe astratta che rappresenta il padre delle classi counter delle varie metriche;
- **Utilizzo:** descrive lo scheletro dell'algoritmo di counting necessario ad effettuare il conteggio di determinati dati ricavati con lo scopo ottenere un trend;
- **Relazioni con altre classi:**
 - server::miner_G::fb::AbsFbCounter
 - server::miner_G::tw::AbsTwCounter
 - server::miner_G::ig::AbsIgCounter
- **Attributi:**
 - `counter_int : int`

Descrizione: contiene la somma richiesta;
- **Metodi:**
 - `+ __init__() : void`

Descrizione: costruttore della classe di default. Inizializza la variabile del contatore a 0;

 - `+ abstract action() : void`

Descrizione: il metodo in questa classe viene solamente marcato come astratto, verrà implementato successivamente nelle classi figlie. Il metodo viene utilizzato in ogni elemento iterato;
- `# abstract break_condition() : void`

Descrizione: il metodo in questa classe viene solamente marcato come astratto, verrà implementato successivamente nelle classi figlie. Il metodo blocca il count degli elementi;
- `# abstract count() : void`

Descrizione: il metodo in questa classe viene solamente marcato come astratto, verrà implementato successivamente nelle classi figlie. Il metodo avvia il count degli elementi ;

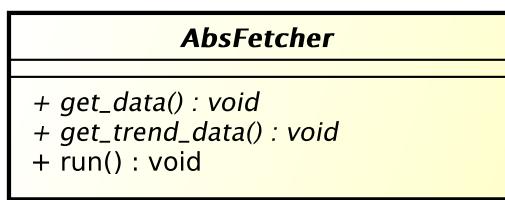


Figura 90: Classe - server::miner::AbsFetcher

3.3.16.1.3 server::miner::AbsFetcher

- **Descrizione:** classe astratta che rappresenta il padre delle classi fetcher dei vari social network;
- **Utilizzo:** è utilizzata per mantenere l'estensibilità nel caso l'applicazione venga estesa con altre API_G oltre a quelli dei social network presi in considerazione;

- **Relazioni con altre classi:**

- server::miner_G::fb::AbsFbFetcher
- server::miner_G::tw::AbsTwFetcher
- server::miner_G::ig::AbsIgFetcher

- **Attributi:** N/A

- **Metodi:**

- `+ abstract get_data() : void`

Descrizione: il metodo in questa classe viene solamente marcato come astratto, verrà implementato successivamente nelle classi figlie. Il metodo ottiene i dati statici del post;

- `+ abstract get_trend_data() : void`

Descrizione: il metodo in questa classe viene solamente marcato come astratto, verrà implementato successivamente nelle classi figlie. Il metodo ottiene i dati del trend;

- `+ run() : void`

Descrizione: il metodo avvia sequenzialmente get_data ed get_trend_data;

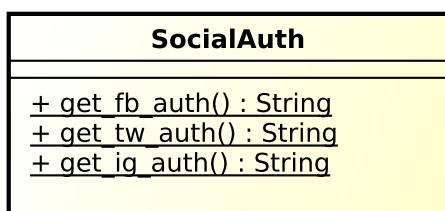


Figura 91: Classe - server::miner::SocialAuth

3.3.16.1.4 server::miner::SocialAuth

- **Descrizione:** classe che rappresenta la fase di autenticazione dei social network. Essendo una classe di utilità, essa fornisce dei metodi statici per ottenere facilmente i dati relativi ai permessi da utilizzare per le API_G dei vari social network.

- **Utilizzo:** fornisce metodi statici per effettuare l'autenticazione ai social network presi in considerazione;

- **Relazioni con altre classi:**

- server::miner_G::fb::FbEventFetcher
- server::miner_G::fb::FbPageFetcher
- server::miner_G::fb::PostsCounter
- server::miner_G::tw::TwHashtagFetcher
- server::miner_G::tw::HashtagTweetCounter
- server::miner_G::tw::TwUserFetcher
- server::miner_G::tw::UserTweetCounter
- server::miner_G::ig::IgUserFetcher

- server::miner_G::ig::IgHashtagFetcher
- server::miner_G::ig::MediaCounter

- **Attributi:** N/A

- **Metodi:**

- + **static get_fb_auth() : String**

Descrizione: il metodo deve essere dichiarato statico. Il metodo ha il compito di effettuare l'autenticazione su Facebook e di restituire l'access token valido per le chiamate;

- + **static get_tw_auth() : String**

Descrizione: il metodo deve essere dichiarato statico. Il metodo ha il compito di effettuare l'autenticazione su Twitter e di restituire l'access token valido per le chiamate;

- + **static get_ig_auth() : String**

Descrizione: il metodo deve essere dichiarato statico. Il metodo ha il compito di effettuare l'autenticazione su Instagram e di restituire il l'access token valido per le chiamate;

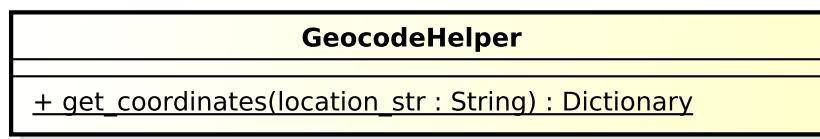


Figura 92: Classe - server::miner::GeocodeHelper

3.3.16.1.5 server::miner::GeocodeHelper

- **Descrizione:** classe che si occupa di convertire la geolocalizzazione da una località a coordinate geografiche. Essa è stata implementata utilizzando metodi statici in quanto essendo un metodo di utilità, non presenta la necessità di interagire con l'istanza della classe.

- **Utilizzo:** fornisce metodi statici ai social network presi in considerazione per la conversione dei luoghi in latitudine e longitudine;

- **Relazioni con altre classi:**

- server::miner_G::fb::FbEventFetcher
- server::miner_G::tw::UserTweetCounter
- server::miner_G::ig::MediaCounter

- **Attributi:** N/A

- **Metodi:**

- + **static get_coordinates(location_str : String) : Dictionary**

Descrizione: il metodo deve essere dichiarato statico. Il metodo ha il compito di ricevere un indirizzo o una città e restituire le relative coordinate di latitudine e longitudine sotto forma di dizionario;

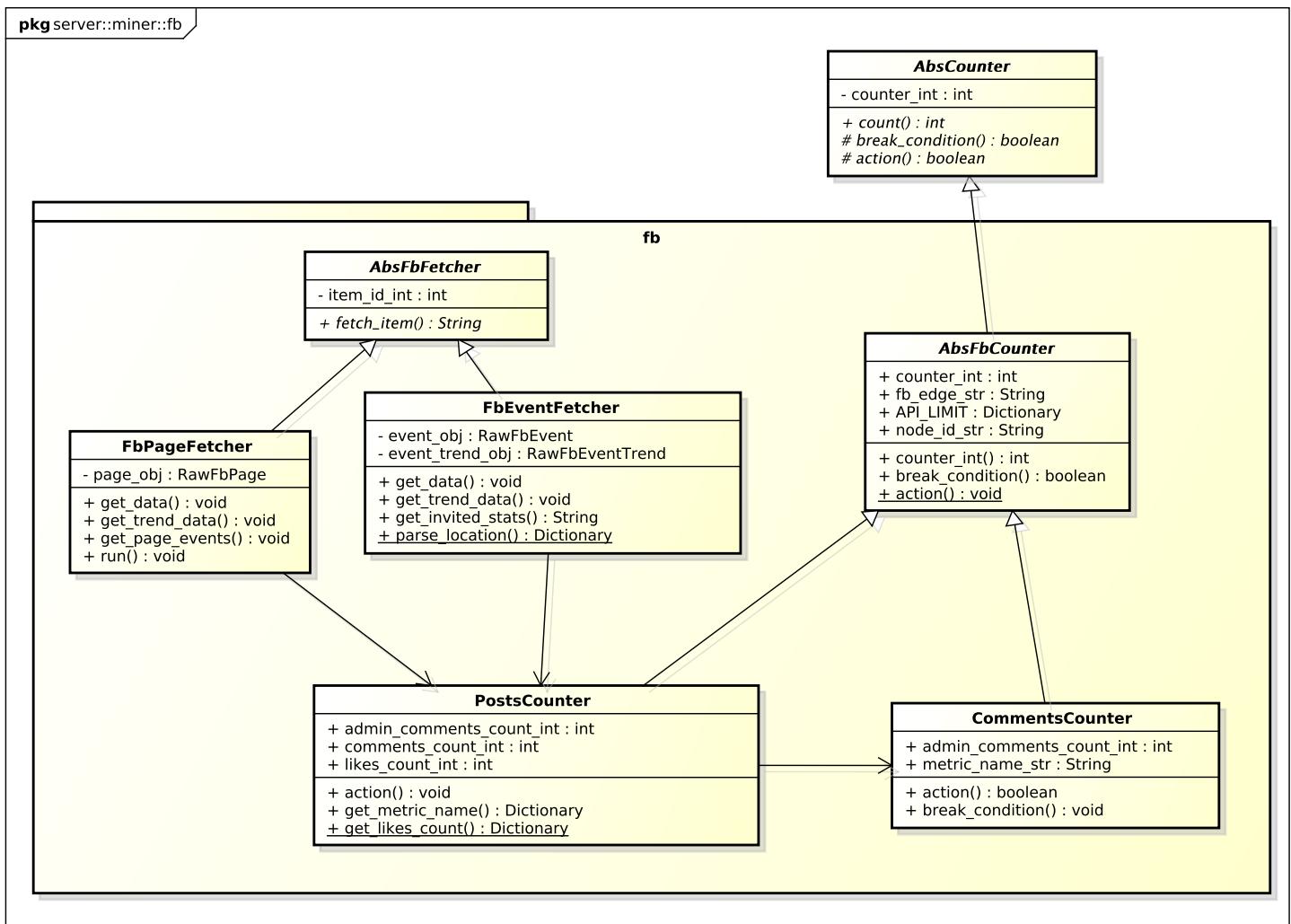


Figura 93: Package - server::miner::fb

3.3.17 server::miner::fb

- **Descrizione:** è il package_G che contiene tutte le classi che includono i metodi per prelevare i dati da Facebook e salvarli nel database_G;
- **Padre:** server::miner_G
- **Interazione con altri componenti:**
 - server::db

3.3.17.1 Classi

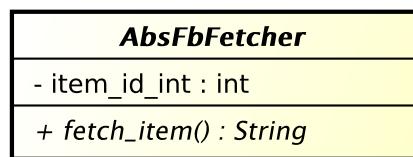


Figura 94: Classe - server::miner::fb::AbsFbFetcher

3.3.17.1.1 server::miner::fb::AbsFbFetcher

- **Descrizione:** classe astratta che rappresenta il padre di ogni fetcher relativo a Facebook;
- **Utilizzo:** contiene un metodo che ricava i dati statici di un'entità Facebook ed un metodo astratto che verrà definito nelle classi figlie;
- **Classi ereditate:** server::miner_G::AbsFetcher
- **Relazioni con altre classi:**
 - server::miner_G::fb::FbPageFetcher
 - server::miner_G::fb::FbEventFetcher
- **Attributi:**
 - + node_id_str : String

Descrizione: contiene ed identifica l'id univoco della risorsa sul sito Facebook;
- **Metodi:**
 - + __init__() : void

Descrizione: costruttore della classe di default. Inizializza la variabile node_id_str con l'id della risorsa;

 - + abstract fetch_item() : String

Descrizione: il metodo attiva l'autenticazione sulle API_G di Facebook ed effettua la chiamata utilizzando node_id_str. Ritorna i dati ottenuti dalla chiamata;

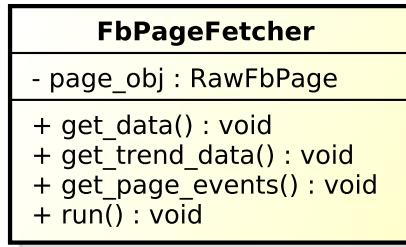


Figura 95: Classe - server::miner::fb::FbPageFetcher

3.3.17.1.2 server::miner::fb::FbPageFetcher

- **Descrizione:** classe che si occupa ricava i dati dalle pagine Facebook;
- **Utilizzo:** contiene un campo che descrive i dati statici di una pagina, un campo che descrive i dati dinamici ed un metodo che li ricava tramite l'utilizzo della classe PostsCounter;
- **Classi ereditate:** server::miner_G::fb::AbsFbFetcher
- **Relazioni con altre classi:**
 - server::miner_G::fb::PostsCounter
- **Attributi:**
 - `- page_obj : RawFbPage`
Descrizione: la variabile contiene i risultati della chiamata effettuata tramite le API_G di Facebook riguardante una pagina utente;
- **Metodi:**
 - `+ __init__(name_str : String) : void`
Descrizione: costruttore della classe di default. Inizializza la variabile page_obj ed effettua la costruzione della classe astratta derivata passando l'id della risorsa;
 - `+ get_data() : void`
Descrizione: il metodo recupera i dati statici della pagina utente dalle API_G di Facebook e li memorizza nella base di dati;
 - `+ get_trend_data() : void`
Descrizione: il metodo ha il compito di recuperare tramite le API_G di Facebook i dati della pagina utente, di rielaborarli e salvarli nella base di dati;
 - `+ get_page_events() : void`
Descrizione: il metodo tramite chiamate alle API_G di Facebook scarica tutti i dati associati agli eventi di una pagina, li rielabora e li inserisce nella base di dati;
 - `+ run() : void`
Descrizione: il metodo invoca la costruzione della classe astratta derivata ed avvia l'acquisizione dei dati. Inoltre avvia l'acquisizione dei dati statici e dinamici delle pagine eventi su Facebook;

FbEventFetcher
- event_obj : RawFbEvent
+ parent_key_int : int
+ get_data() : void
+ get_trend_data() : void
+ get_invited_stats() : Dictionary
+ parse_location(event : String) : Dictionary

Figura 96: Classe - server::miner::fb::FbEventFetcher

3.3.17.1.3 server::miner::fb::FbEventFetcher

- **Descrizione:** classe che ricava i dati di un evento Facebook;
- **Utilizzo:** contiene un campo che descrive i dati statici di un evento, un campo che descrive i dati dinamici e un metodo che li ricava tramite l'utilizzo della classe PostsCounter;
- **Classi ereditate:** server::miner_G::fb::AbsFbFetcher
- **Relazioni con altre classi:**
 - server::miner_G::fb::PostsCounter
- **Attributi:**
 - + parent_key_int : int

Descrizione: viene indicata la chiave del padre per poter collegare il nuovo nodo come figlio;
 - - event_obj : RawFbEvent

Descrizione: la variabile contiene i risultati della chiamata effettuata tramite le API_G di Facebook riguardante una pagina evento;
- **Metodi:**
 - + __init__() : void

Descrizione: costruttore della classe di default. Inizializza la classe astratta derivata passandogli l'id della risorsa. Inizializza inoltre event_obj e parent_key con i dati a disposizione;
 - + get_invited_stats() : Dictionary

Descrizione: il metodo recupera i dati statici riguardanti le statistiche di partecipazione della pagina evento tramite le API_G di Facebook e li memorizza nella base di dati;
 - + get_data() : void

Descrizione: il metodo recupera i dati statici della pagina evento tramite le API_G di Facebook e li memorizza nella base di dati;
 - + get_trend_data() : void

Descrizione: il metodo ha il compito di recuperare tramite le API_G di Facebook i dati della pagina evento, di rielaborarli e salvarli nella base di dati;

– + static parse_location(event: String) : Dictionary

Descrizione: il metodo deve essere dichiarato statico. Tramite l'uso del metodo GeoCodeHelper si ricavano latitudine e longitudine e si restituiscono con tipo dizionario;

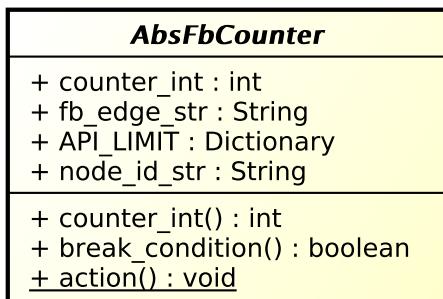


Figura 97: Classe - server::miner::fb::AbsFbCounter

3.3.17.1.4 server::miner::fb::AbsFbCounter

- **Descrizione:** classe astratta che rappresenta il padre per tutte le classi counter delle metriche di Facebook;

- **Utilizzo:** classe che contiene l'id delle metriche su cui effettuare il counting, questa classe effettua l'overloading dei metodi della classe padre specificando la struttura dell'algoritmo per il counting su dati Facebook;

- **Classi ereditate:** server::miner_G::AbsCounter

- **Relazioni con altre classi:**

- server::miner_G::fb::PostsCounter
- server::miner_G::fb::CommentsCounter

- **Attributi:**

- + API_G_LIMIT : Dictionary

Descrizione: la variabile contiene un dizionario di chiavi-valore, riguardanti i limiti delle chiamate alle API_G Facebook;

- + node_id_str : String

Descrizione: la variabile contiene l'id della risorsa univoca presente in Facebook;

- + fb_edge_str : String

Descrizione: la variabile contiene il parametro richiesto dalle API_G di Facebook per ricavare i vari tipi di dato associati ad una metrica_G evento o pagina;

- + counter_int : int

Descrizione: la variabile contiene il numero di occorrenze dell'oggetto specifico di Facebook;

- **Metodi:**

- + __init__(node_id : int, fb_edge : String)

Descrizione: costruttore della classe di default. Il metodo inizializza le variabili passate e inizializza a 0 il counter _int;

– + `counter_int() : int`

Descrizione: il metodo conta le occorrenze dell'oggetto Facebook passato dal costruttore;

– + `break_condition() : boolean`

Descrizione: il metodo definisce le condizioni di arresto del loop per il counting;

– + `abstract action() : void`

Descrizione: il metodo in questa classe viene solamente marcato come astratto, verrà implementato successivamente nelle classi figlie. Il metodo viene utilizzato in ogni elemento ciclato;

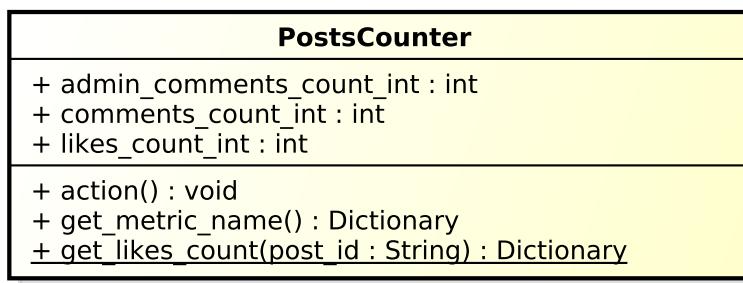


Figura 98: Classe - server::miner::fb::PostsCounter

3.3.17.1.5 server::miner::fb::PostsCounter

- **Descrizione:** classe che descrive l'algoritmo per calcolare il numero dei post per ogni evento o pagina;

- **Utilizzo:** classe che contiene un campo per il numero dei like e un campo per il numero dei talking about per ogni post di Facebook;

- **Classe ereditate:** server::miner_G::fb::AbsFbCounter

- **Relazioni con altre classi:**

 - server::miner_G::fb::CommentsCounter

- **Attributi:**

 - + `admin_comments_count_int : int`

Descrizione: la variabile contiene il numero di commenti dell'amministratore;

 - + `comments_count_int : int`

Descrizione: la variabile contiene il numero di commenti totali;

 - + `likes_count_int : int`

Descrizione: la variabile contiene il numero di likes della pagina;

- **Metodi:**

 - + `__init__(node_id_str : String, fb_edge_str : String) : void`

Descrizione: costruttore della classe di default. Il metodo invoca la costruzione della classe astratta derivata settando l'id della risorsa e il tipo di metrica_G associata. Inoltre inizializza a 0 le variabili per il conteggio;

– + `action() : void`

Descrizione: il metodo viene definito. Effettua la somma dei commenti e dei likes di una pagina Facebook;

– + `get_metric_name(): Dictionary`

Descrizione: il metodo identifica il tipo di pagina analizzate e restituisce un dizionario contenente il tipo;

– + `static get_likes_count(post_id : String) : Dictionary`

Descrizione: il metodo in questa classe viene marcato astratto, implementa il conteggio dei like nei post di una pagina Facebook. Ritorna un dizionario con la quantità di likes ottenuti;

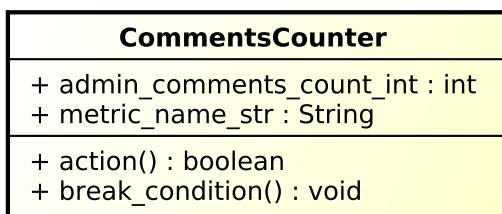


Figura 99: Classe - server::miner::fb::CommentsCounter

3.3.17.1.6 server::miner::fb::CommentsCounter

- **Descrizione:** classe che descrive l'algoritmo per calcolare il numero di commenti per ogni post;

- **Utilizzo:** classe che ricava il numero di commenti per ogni post;

- **Classe ereditate:** server::miner_G::fb::AbsFbCounter

- **Attributi:**

– + `admin_comments_count_int : int`

Descrizione: variabile contenente il numero di commenti effettuati dall'admin della pagina Facebook ;

– + `metric_name_str : String`

Descrizione: la variabile contiene il parametro ricavare i vari tipi di dato associati ad una metrica_G evento o pagina; ;

- **Metodi:**

– + `__init__(node_id_str : int, metric_name_str: String, fb_edge_str : String) : void`

Descrizione: costruttore della classe di default. Il metodo invoca il costruttore della classe astratta passandogli l'id della risorsa e il tipo di metrica_G da applicare alla risorsa. Vengono inoltre inizializzate le variabili a 0;

– + **action()** : boolean

Descrizione: il metodo effettua i conteggi per ricavare la quantità di commenti dell'utente amministratore;

– + **break_condition()** : void

Descrizione: il metodo viene definito ma non implementato;

3.3.18 server::miner::tw

- **Descrizione:** è il package_G che contiene tutte le classi che includono i metodi per prelevare i dati da Twitter e salvarli nel database_G;

- **Padre:** server::miner_G

- **Interazione con altri componenti:**

– server::db

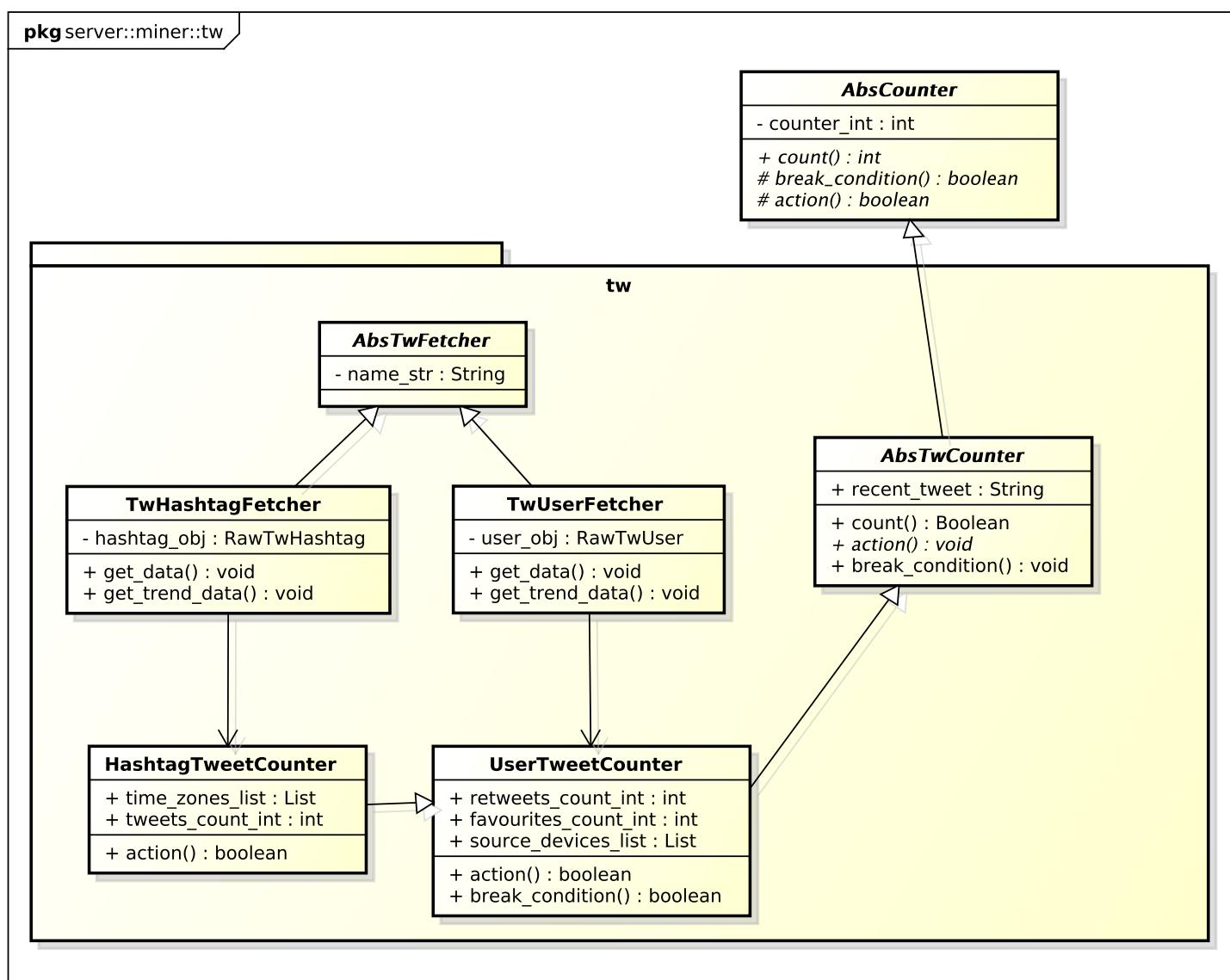


Figura 100: Package - server::miner::tw

3.3.18.1 Classi

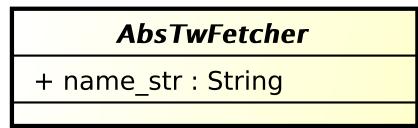


Figura 101: Classe - server::miner::tw::AbsTwFetcher

3.3.18.1.1 server::miner::tw::AbsTwFetcher

- **Descrizione:** classe astratta che rappresenta il padre di ogni fetcher relativo a Twitter;
- **Utilizzo:** contiene un metodo che ricava i dati statici di un'entità Twitter ed un metodo astratto che verrà definito nelle classi figlie;
- **Classi ereditate:** server::miner_G::AbsFetcher
- **Relazioni con altre classi:**
 - server::miner_G::tw::TwHashtagFetcher
 - server::miner_G::tw::TwUserFetcher
- **Attributi:**
 - + `name_str : String`

Descrizione: la variabile contiene l'id univoco della risorsa presente nel social network Twitter;
- **Metodi:**
 - + `__init__(name_str : String)`

Descrizione: costruttore della classe di default. Il metodo invoca la classe astratta per la costruzione;

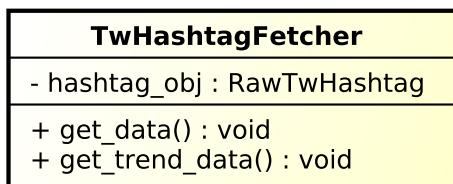


Figura 102: Classe - server::miner::tw::TwHashtagFetcher

3.3.18.1.2 server::miner::tw::TwHashtagFetcher

- **Descrizione:** classe che ricava i dati degli hashtag di Twitter;
- **Utilizzo:** classe che contiene un campo che descrive i dati statici di un hashtag di Twitter e un metodo che ricava i dati dinamici tramite l'utilizzo di HashtagTweetCounter;
- **Classi ereditate:** server::miner_G::tw::AbsTwFetcher
- **Relazioni con altre classi:**

- server::miner_G::tw::HashtagTweetCounter

- **Attributi:**

- - `hashtag_obj : RawTwHashtag`

Descrizione: la variabile contiene l'oggetto RawTwHashtag per poterne poi aggiungere i figli associati;

- **Metodi:**

- + `__init__(name_str : String)`

Descrizione: costruttore della classe di default. Il metodo invoca la classe astratta per la costruzione passando il nome della risorsa. Istanzia l'oggetto hashtag_obj;

- + `get_data() : void`

Descrizione: il metodo costruisce, se non già presente, l'oggetto padre contenente il nome e la data di creazione della risorsa ricercata;

- + `get_trend_data() : void`

Descrizione: il metodo ha il compito di ricercare i dati di trend per l'hashtag. Dopo averli sommati, vengono immagazzinati nella base di dati;

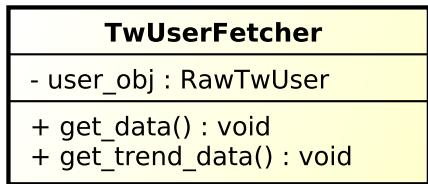


Figura 103: Classe - server::miner::tw::TwUserFetcher

3.3.18.1.3 server::miner::tw::TwUserFetcher

- **Descrizione:** classe che ricava i dati degli utenti di Twitter;
- **Utilizzo:** classe che contiene un campo che descrive i dati statici dell'utente ed un metodo che ricava i dati dinamici tramite l'utilizzo di UserTweetCounter;
- **Classi ereditate:** server::miner_G::tw::AbsTwFetcher
- **Relazioni con altre classi:**

- server::miner_G::tw::UserTweetCounter

- **Attributi:**

- - `user_obj : RawTwUser`

Descrizione: la variabile contiene l'oggetto RawTwUser per poterne poi aggiungere i figli associati;

- **Metodi:**

- + `__init__(name_str: String) : void`

Descrizione: costruttore della classe di default. Il metodo invoca la classe astratta per la costruzione passando il nome della risorsa. Istanzia l'oggetto user_obj;

– + `get_data() : void`

Descrizione: il metodo tramite le API_G di Twitter ottiene i dati statici della pagina utente e li memorizza nella base di dati;

– + `get_trend_data() : void`

Descrizione: il metodo tramite le API_G di Twitter ottiene i dati dinamici della pagina utente e tramite ulteriori chiamate ai metodi ne ricava i dati di trend. Alla fine della elaborazione viene creato l'oggetto padre ed i relativi figli;

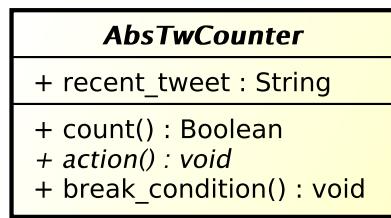


Figura 104: Classe - server::miner::tw::AbsTwCounter

3.3.18.1.4 server::miner::tw::AbsTwCounter

- **Descrizione:** classe astratta che rappresenta il padre per tutte le classi counter delle metriche di Twitter;

- **Utilizzo:** oltre a contenere l'id delle metriche su cui effettuare il counting, questa classe effettua l'overloading dei metodi della classe padre specificando la struttura dell'algoritmo per il counting su dati Twitter;

- **Classi ereditate:** server::miner_G::AbsCounter

- **Relazioni con altre classi:**
 - server::miner_G::UserTweetCounter

- **Attributi:**
 - + `recent_tweet : String`

Descrizione: la variabile contiene il nome della metrica_G rispetto alla pagina Twitter analizzata;

- **Metodi:**
 - + `__init__(recent_tweet : String) : void`

Descrizione: costruttore della classe di default. Il metodo invoca la classe astratta per la costruzione passando la metrica_G della risorsa. Istanzia l'oggetto recent_tweet;

- + `count() : Boolean`

Descrizione: il metodo conta le occorrenze dell'oggetto passato implementando l'azione si stop quando necessario. Ritorna False quando l'azione non è più necessaria;

- + abstract `action()` : void
- Descrizione:** il metodo viene definito astratto ma non viene implementato;
- + abstract `break_condition()` : void
- Descrizione:** il metodo viene definito astratto ma non viene implementato;

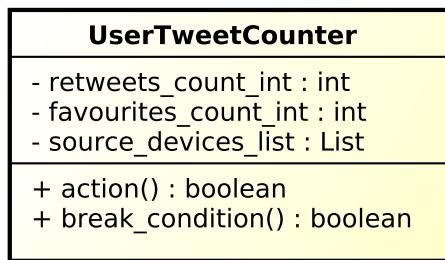


Figura 105: Classe - server::miner::tw::UserTweetCounter

3.3.18.1.5 server::miner::tw::UserTweetCounter

- **Descrizione:** classe che descrive l'algoritmo per il counting dei campi di un tweet, relativo ad un utente, di cui ci interessa fare un trend;
- **Utilizzo:** classe che ricava il numero dei retweets, il numero dei favoriti, il tipo di device, e il numero di media per ogni tweet;
- **Classi ereditate:** server::miner_G::tw::AbsTwCounter
- **Attributi:**

- + `favourites_count_int` : int

Descrizione: la variabile contiene il numero di favoriti contati nei tweets;

- + `retweets_count_int` : int

Descrizione: la variabile contiene il numero retweets contati nei tweets;

- + `source_devices_list` : List

Descrizione: la variabile contiene una lista chiave-valore di dispositivi e quantità utilizzati per la pubblicazione dei tweets;

- + `date_created_date` : Date

Descrizione: la variabile contiene la data di acquisizione dei dati;

- **Metodi:**

- + `__init__(name_str : String)` : void

Descrizione: costruttore della classe di default. Il metodo invoca la classe astratta per la costruzione passando l'id della risorsa. Istanzia i contatori a 0 e la lista vuota;

- + `action(tweet : TweetUserFetcher)` : void

Descrizione: il metodo invoca la somma dei campi dati dell'oggetto passato;

- + `break_condition(data : Date)` : Boolean

Descrizione: il metodo definisce la condizione di stop per i cicli di somma;



Figura 106: Classe - server::miner::tw::HashtagTweetCounter

3.3.18.1.6 server::miner::tw::HashtagTweetCounter

- **Descrizione:** classe che descrive l'algoritmo per il counting dei campi di un tweet, relativo ad un hashtag, di cui ci interessa fare un trend;

- **Utilizzo:** classe che ricava anche la time zone di ogni tweet;

- **Classi ereditate:** server::miner_G::tw::UserTweetCounter

- **Attributi:**

- + favourites_count_int : int

Descrizione: la variabile contiene la somma dei favoriti dell'hashtag Twitter;

- + time_zone_list : Date

Descrizione: la variabile contiene la lista formata da chiave-valore dei timezone della pubblicazione dell'hashtag Twitter;

- + tweets_count_int : int

Descrizione: la variabile contiene la somma dei tweets dell'hashtag Twitter;

- **Metodi:**

- + __init__(name_str: String) : void

Descrizione: costruttore della classe di default. Il metodo invoca la classe astratta per la costruzione passando l'id della risorsa. Istanzia i contatori a 0, la lista vuota e la data di creazione;

- + action(tweet : TwHashtagFetcher) : void

Descrizione: il metodo definisce i campi dati da sommare;

3.3.19 server::miner::ig

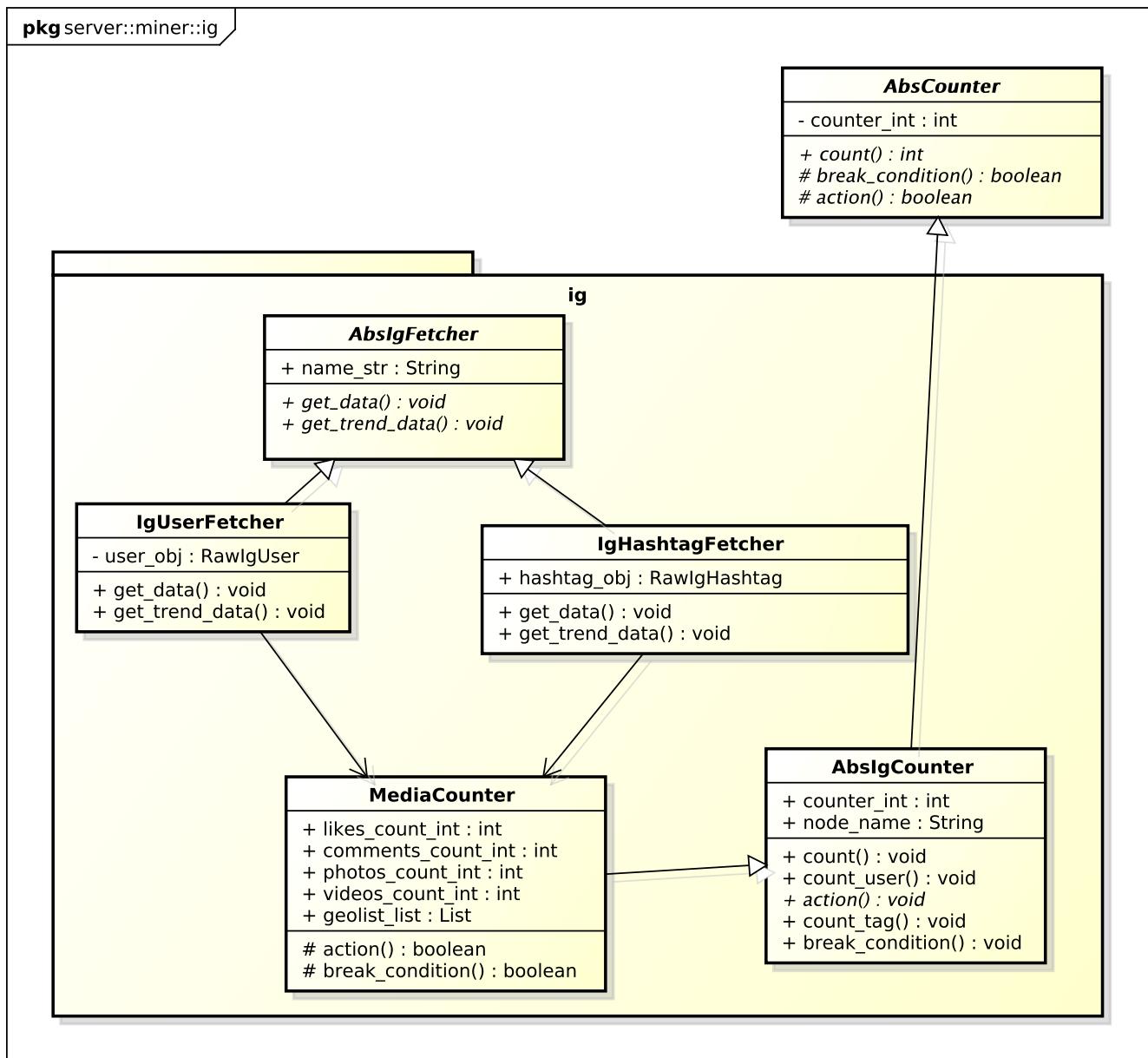


Figura 107: Package - server::miner::ig

- **Descrizione:** è il package_G che contiene tutte le classi che includono i metodi per prelevare i dati da Instagram e salvarli nel database_G;
- **Padre:** server::miner_G
- **Interazione con altri componenti:**
 - server::db

3.3.19.1 Classi

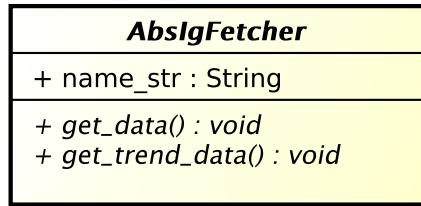


Figura 108: Classe - server::miner::ig::AbsIgFetcher

3.3.19.1.1 server::miner::ig::AbsIgFetcher

- **Descrizione:** classe astratta che rappresenta il padre di ogni fetcher relativo a Instagram;
- **Utilizzo:** contiene un metodo che ricava i dati statici di un'entità Instagram ed un metodo astratto che verrà definito nelle classi figlie;
- **Classi ereditate:** server::miner_G::AbsFetcher
- **Relazioni con altre classi:**
 - server::miner_G::ig::IgUserFetcher
 - server::miner_G::ig::IgHashtagFetcher
- **Attributi:**
 - + `name_str` : `String`

Descrizione: la variabile contiene l'id univoco della risorsa presente nel social network Instagram;
- **Metodi:**
 - + `__init__(name_str : String) : void`

Descrizione: costruttore della classe di default. Il metodo invoca la costruzione della classe astratta derivata. Inoltre inizializza `name_str` con l'id appropriato della risorsa;

 - + `abstract get_data() : void`

Descrizione: il metodo viene definito astratto ma non viene implementato;
- + `abstract get_trend_data() : void`

Descrizione: il metodo viene definito astratto ma non viene implementato;

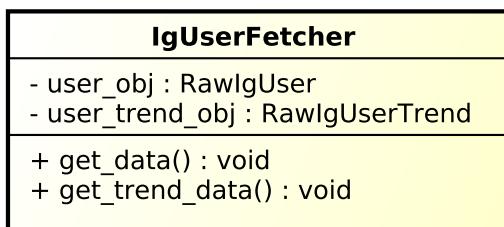


Figura 109: Classe - server::miner::ig::IgUserFetcher

3.3.19.1.2 server::miner::ig::IgUserFetcher

- **Descrizione:** classe che ricava i dati dagli utenti di Instagram;
- **Utilizzo:** classe che contiene un campo che descrive i dati statici dell'utente ed un metodo che ricava i dati dinamici tramite l'utilizzo di `MediaCounter`;
- **Classi ereditate:** `server::minerG::ig::AbsIgFetcher`
- **Relazioni con altre classi:**
 - `server::minerG::ig::MediaCounter`
- **Attributi:**
 - `+ user_obj : RawIgUser`

Descrizione: la variabile contiene la metrica_G cui andranno collegati i figli. Nel seguente caso la metrica riguarda l'utente di Instagram;
- **Metodi:**
 - `+ __init__(name_str : String)`

Descrizione: costruttore della classe di default. Il metodo invoca la costruzione della classe astratta derivata. Inoltre inizializza la variabile `user_obj`;

 - `+ get_data() : void`

Descrizione: il metodo tramite le chiamate alle API_G di Instagram recupera i dati statici dell'utente e li memorizza nella base di dati. Effettua un controllo preventivo che questi non siano già esistenti;

 - `+ get_trend_data() : void`

Descrizione: il metodo tramite le chiamate alle API_G di Instagram recupera i dati di trend dell'utente. Effettua le opportune chiamate per l'aggregazione dei dati e aggiunge i dati come figli all'oggetto `user_obj` creato precedentemente;

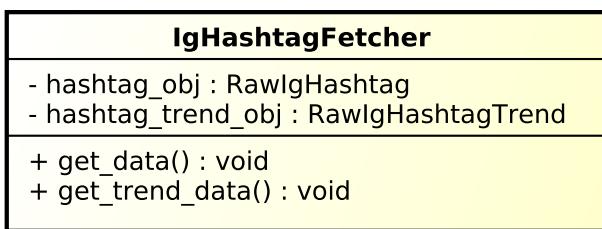


Figura 110: Classe - `server::miner::ig::IgHashtagFetcher`

3.3.19.1.3 `server::miner::ig::IgHashtagFetcher`

- **Descrizione:** classe che ricava i dati dagli hashtag di Instagram;
- **Utilizzo:** classe che contiene un campo che descrive i dati statici dell'hashtag ed un metodo che ricava i dati dinamici tramite l'utilizzo di `MediaCounter`;
- **Classi ereditate:** `server::minerG::ig::AbsIgFetcher`
- **Relazioni con altre classi:**

- server::miner_G::ig::MediaCounter

- **Attributi:**

- – `hashtag_obj : RawIgHashtag`

Descrizione: la variabile contiene la metrica_G cui andranno collegati i figli. Nel seguente caso la metrica riguarda l'hashtag su Instagram;

- **Metodi:**

- + `_init_(name_str : String) : void`

Descrizione: costruttore della classe di default. Il metodo invoca la costruzione della classe astratta derivata. Inoltre inizializza la variabile hashtag_obj;

- + `get_data() : void`

Descrizione: il metodo tramite le chiamate alle API_G di Instagram recupera i dati statici dell'hashtag e li memorizza nella base di dati. Effettua un controllo preventivo che questi non siano già esistenti;

- + `get_trend_data() : void`

Descrizione: il metodo tramite le chiamate alle API_G di Instagram recupera i dati di trend dell'hashtag. Effettua le opportune chiamate per l'aggregazione dei dati e aggiunge i dati come figli all'oggetto hashtag_obj creato precedentemente;

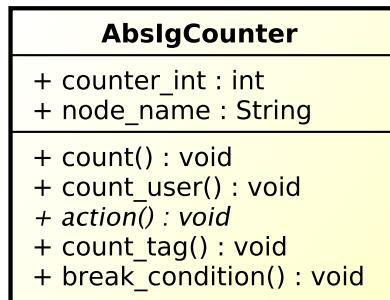


Figura 111: Classe - server::miner::ig::AbsIgCounter

3.3.19.1.4 server::miner::ig::AbsIgCounter

- **Descrizione:** classe astratta che rappresenta il padre per tutte le classi counter delle metriche relative ad Instagram;

- **Utilizzo:** classe che contiene l'id delle metriche su cui effettuare il counting, questa classe effettua l'overloading dei metodi della classe padre per specificarne l'utilizzo sui dati relativi ad Instagram;

- **Classi ereditate:** server::miner_G::AbsCounter

- **Relazioni con altre classi:**

- server::miner_G::ig::MediaCounter

- **Attributi:**

- + `counter_int : int`

Descrizione: la variabile contiene il numero di cicli effettuati per la somma;

– + `node_name : String`

Descrizione: la variabile contiene l'id della risorsa utile per la memorizzazione dei dati acquisiti successivamente;

- **Metodi:**

– + `__init__(node_name : String) : void`

Descrizione: costruttore della classe di default. Il metodo invoca la costruzione della classe astratta derivata. Inoltre inizializza gli attributi presenti;

– + `count() : void`

Descrizione: il metodo ha il compito di avviare il count dei trend utente e dei trend hashtag;

– + `count_user() : void`

Descrizione: il metodo ha il compito di sommare i dati dinamici dell'utente per ottenere i dati di trend;

– + `count_tag() : void`

Descrizione: il metodo ha il compito di sommare i dati dinamici dell'hashtag per ottenere i dati di trend;

– + `abstract action() : void`

Descrizione: il metodo viene definito astratto ma non viene implementato;

– + `abstract break_condition() : void`

Descrizione: il metodo viene definito astratto ma non viene implementato;

MediaCounter
- <code>likes_count_int : int</code> - <code>comments_count_int : int</code> - <code>photos_count_int : int</code> - <code>videos_count_int : int</code> - <code>geolist_list : List</code>
<code># action() : boolean</code> <code># break_condition() : boolean</code>

Figura 112: Classe - server::miner::ig::MediaCounter

3.3.19.1.5 server::miner::ig::MediaCounter

- **Descrizione:** classe che descrive l'algoritmo per il counting delle metriche necessarie a descrivere il trend dei media di un utente o di un hashtag Instagram;
- **Utilizzo:** classe che ricava il numero di commenti, il numero di like, il numero di foto e il numero di video di un media;
- **Classi ereditate:** server::miner_G::AbsIgCounter

- **Attributi:**

- + `likes_count_int : int`

Descrizione: la variabile contiene il numero di likes sommati;

- + `comments_count_int : int`

Descrizione: la variabile contiene il numero di commenti sommati;

- + `photos_count_int : int`

Descrizione: la variabile contiene il numero di foto sommate;

- + `videos_count_int : int`

Descrizione: la variabile contiene il numero di video sommati;

- + `geolist_list : List`

Descrizione: la variabile contiene una lista formata da chiave-valore di GeoPoint riguardanti la localizzazione dei media;

- **Metodi:**

- + `__init__() : void`

Descrizione: costruttore della classe di default. Il metodo invoca la costruzione della classe astratta derivata. Inoltre inizializza gli attributi presenti a 0;

- # `action() : void`

Descrizione: il metodo invoca le funzioni di somma dei campi dato da acquisire;

- # `break_condition() : boolean`

Descrizione: il metodo definisce la condizione di stop per i cicli di somma;

3.3.20 server::endpoints

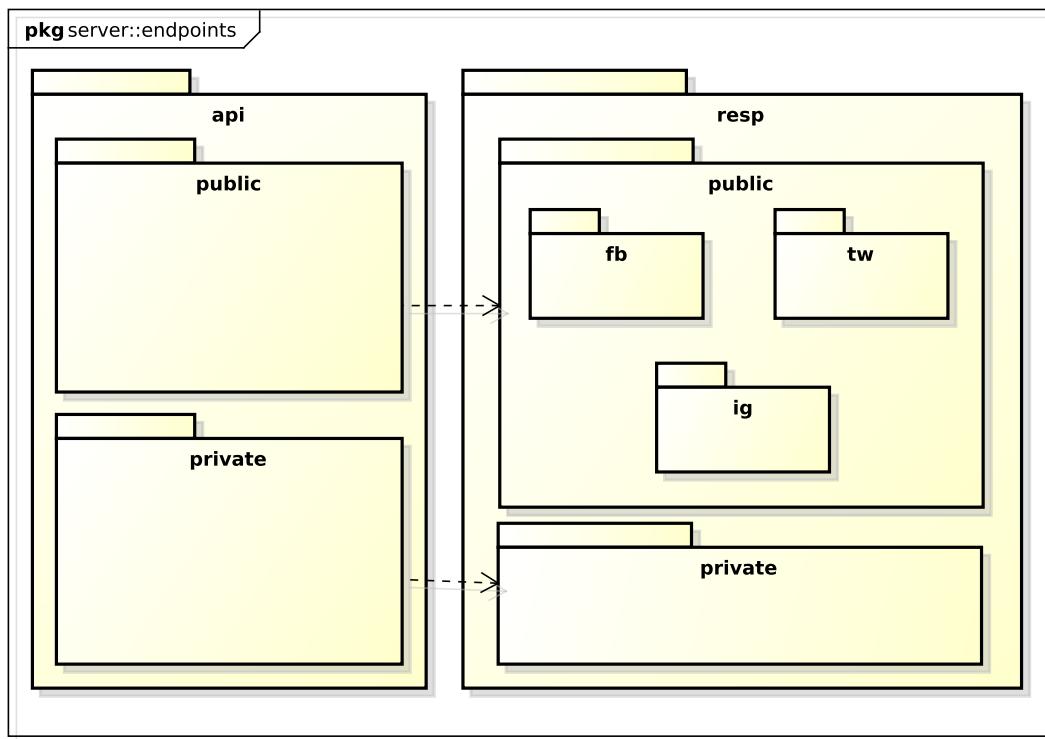


Figura 113: Package - server::endpoints

- **Descrizione:** è il package_G che contiene tutte le classi che implementano l’interfaccia REST_G del sistema tramite Google Cloud Endpoints;
- **Padre:** server
- **Package_G contenuti:**
 - server::endpoints::api_G
 - server::endpoints::resp
- **Interazione con altri componenti:**
 - server::processor_G
 - server::db

3.3.21 server::endpoints::api

In questo package_G e nei suoi package figli viene utilizzata la classe **ResourceContainer** offerta dai Google Cloud Endpoints per le richieste contenenti dei dati da inoltrare al server.

- **Descrizione:** è il package_G che definisce le web API_G offerte dall’applicazione e utilizzate dal client;
- **Padre:** server::endpoints
- **Package_G contenuti:**

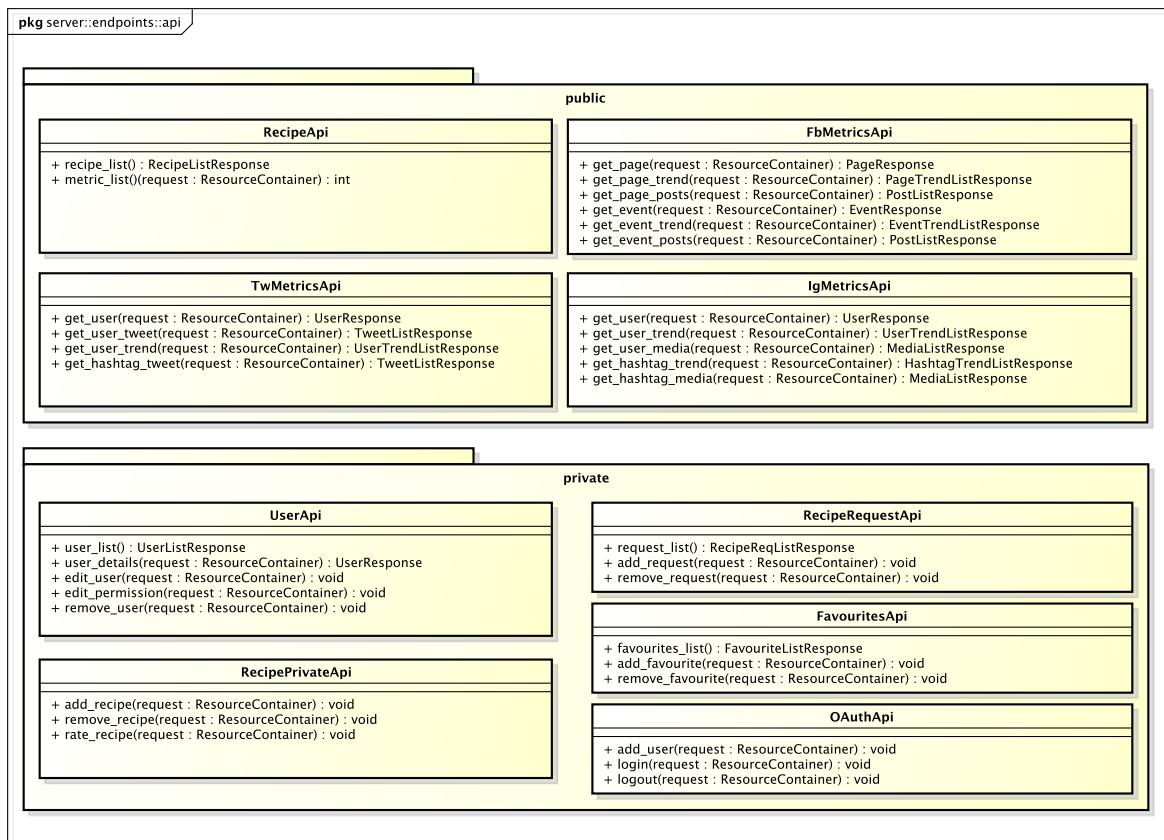


Figura 114: Package - server::endpoints::api

- server::endpoints::api_G::public
- server::endpoints::api_G::private

- **Interazione con altri componenti:**

- client::model::services
- server::processor_G
- server::db
- server::endpoints::resp

3.3.22 server::endpoints::api::public

- **Descrizione:** è il package_G contenente l'implementazione delle web API_G pubbliche;

- **Padre:** server::endpoints::api_G

- **Interazione con altri componenti:**

- server::processor_G
- server::endpoints::resp::public

3.3.22.1 Classi

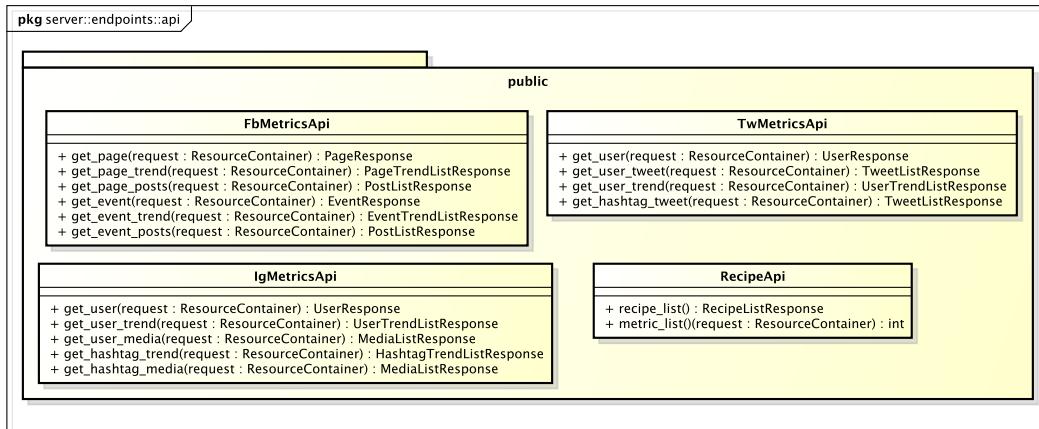


Figura 115: Package - server::endpoints::api::public



Figura 116: Classe - server::endpoints::api::public::RecipeApi

3.3.22.1.1 server::endpoints::api::public::RecipeApi

- **Descrizione:** classe utilizzata in seguito ad una chiamata del client per ricavare la lista delle Recipe_G e delle metriche;
- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client di recuperare la lista delle Recipe_G e delle metriche;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::RecipeListResponse;
- **Attributi:** N/A
- **Metodi:**
 - `public recipeG_list() : RecipeListResponse`

Descrizione: restituisce alla chiamata la lista delle Recipe_G presenti nel sistema.

`– public metric_list(request : ResourceContainer) : RecipeResponse`

Descrizione: restituisce alla chiamata delle informazioni generali su una specifica Recipe_G insieme alla lista delle metriche relative alla Recipe stessa.

3.3.22.1.2 server::endpoints::api::public::FbMetricsApi

- **Descrizione:** classe utilizzata in seguito ad una chiamata del client per ottenere i dati relativi ad una metrica_G di Facebook;
- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client dei dati relativi ad una pagina o ad un evento di Facebook;
- **Relazioni con altre classi:**

FbMetricsApi
<pre>+ get_page(request : ResourceContainer) : PageResponse + get_page_trend(request : ResourceContainer) : PageTrendListResponse + get_page_posts(request : ResourceContainer) : PostListResponse + get_event(request : ResourceContainer) : EventResponse + get_event_trend(request : ResourceContainer) : EventTrendListResponse + get_event_posts(request : ResourceContainer) : PostListResponse</pre>

Figura 117: Classe - server::endpoints::api::public::FbMetricsApi

- server::endpoints::resp::public::fb::PageResponse
- server::endpoints::resp::public::fb::PageTrendListResponse
- server::endpoints::resp::public::fb::PostListResponse
- server::endpoints::resp::public::fb::EventResponse
- server::endpoints::resp::public::fb::EventTrendListResponse

- **Attributi:** N/A

- **Metodi:**

- `public get_page(request : ResourceContainer) : PageResponse`
Descrizione: restituisce alla chiamata i dettagli di una pagina Facebook.
- `public get_page_trend(request : ResourceContainer) : PageTrendListResponse`
Descrizione: restituisce alla chiamata l'elenco dei trend relativi ad una pagina Facebook.
- `public get_page_posts(request : ResourceContainer) : PostListResponse`
Descrizione: restituisce alla chiamata l'elenco dei trend relativi ai post di una pagina Facebook.
- `public get_event(request : ResourceContainer) : EventResponse`
Descrizione: restituisce alla chiamata i dettagli di un event Facebook.
- `public get_event_trend(request : ResourceContainer) : EventTrendListResponse`
Descrizione: restituisce alla chiamata l'elenco dei trend relativi ad un evento Facebook.
- `public get_event_posts(request : ResourceContainer) : PostListResponse`
Descrizione: restituisce alla chiamata l'elenco dei trend relativi ai post di un evento Facebook.

TwMetricsApi
<pre>+ get_user(request : ResourceContainer) : UserResponse + get_user_trend(request : ResourceContainer) : UserTrendListResponse + get_user_tweet(request : ResourceContainer) : TweetListResponse + get_hashtag_tweet(request : ResourceContainer) : TweetListResponse</pre>

Figura 118: Classe - server::endpoints::api::public::TwMetricsApi

3.3.22.1.3 server::endpoints::api::public::TwMetricsApi

- **Descrizione:** classe utilizzata in seguito ad una chiamata del client per ottenere i dati relativi ad una metrica_G di Twitter;

- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client dei dati relativi ad un utente o un hashtag di Twitter;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::tw::UserResponse
 - server::endpoints::resp::public::tw::UserTrendListResponse
 - server::endpoints::resp::public::tw::TweetListResponse
- **Attributi:** N/A
- **Metodi:**
 - `public get_user(request : ResourceContainer) : UserResponse`
Descrizione: restituisce alla chiamata i dettagli relativi ad un utente Twitter.
 - `public get_user_trend(request : ResourceContainer) : UserTrendListResponse`
Descrizione: restituisce alla chiamata l'elenco dei trend relativi ad un utente Twitter.
 - `public get_user_tweet(request : ResourceContainer) : TweetListResponse`
Descrizione: restituisce alla chiamata l'elenco dei trend relativi ai tweet di un utente Twitter.
 - `public get_hashtag_tweet(request : ResourceContainer) : TweetListResponse`
Descrizione: restituisce alla chiamata l'elenco dei trend relativi ai tweet di un hashtag Twitter.

IgMetricsApi
<pre>+ get_user(request : ResourceContainer) : UserResponse + get_user_trend(request : ResourceContainer) : UserTrendListResponse + get_user_media(request : ResourceContainer) : MediaListResponse + get_hashtag_trend(request : ResourceContainer) : HashtagTrendListResponse + get_hashtag_media(request : ResourceContainer) : MediaListResponse</pre>

Figura 119: Classe - server::endpoints::api::public::IgMetricsApi

3.3.22.1.4 server::endpoints::api::public::IgMetricsApi

- **Descrizione:** classe utilizzata in seguito ad una chiamata del client per ottenere i dati relativi ad una metrica_G di Instagram;
- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client dei dati relativi ad un utente o hashtag di Instagram;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::ig::UserResponse
 - server::endpoints::resp::public::ig::UserTrendListResponse
 - server::endpoints::resp::public::ig::MediaListResponse
 - server::endpoints::resp::public::ig::HashtagTrendListResponse
- **Attributi:** N/A

- **Metodi:**

- `public get_user(request : ResourceContainer) : UserResponse`

Descrizione: restituisce alla chiamata i dettagli relativi ad un utente Instagram.

- `public get_user_trend(request : ResourceContainer) : UserTrendListResponse`

Descrizione: restituisce alla chiamata l'elenco dei trend relativi ad un utente Instagram.

- `public get_user_media(request : ResourceContainer) : MediaListResponse`

Descrizione: restituisce alla chiamata l'elenco dei trend relativi ai media di un utente Instagram.

- `public get_hashtag_trend(request : ResourceContainer) : HashtagTrendListResponse`

Descrizione: restituisce alla chiamata l'elenco dei trend relativi ad un hashtag Instagram.

- `public get_hashtag_media(request : ResourceContainer) : MediaListResponse`

Descrizione: restituisce alla chiamata l'elenco dei trend dei media relativi ad un hashtag Instagram.

3.3.23 server::endpoints::api::private

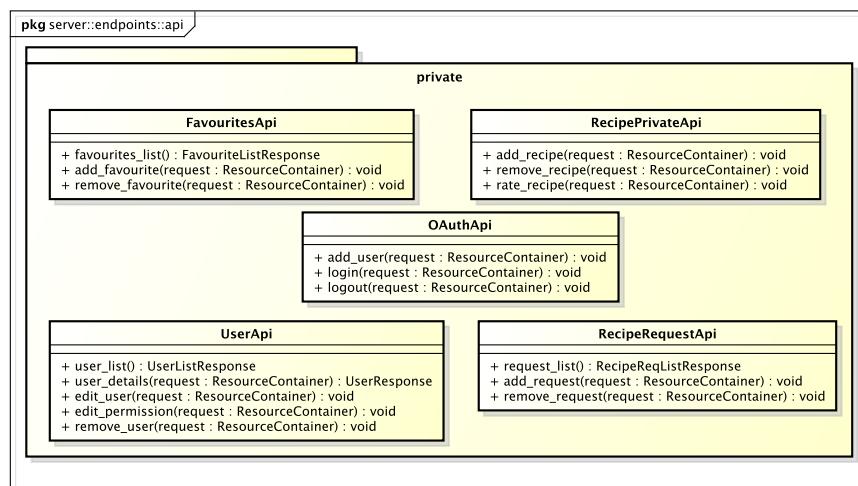


Figura 120: Package - server::endpoints::api::private

- **Descrizione:** è il package_G contenente l'implementazione delle web API_G private;

- **Padre:** server::endpoints::api_G

- **Interazione con altri componenti:**

- server::processor_G

- server::endpoints::resp::private

3.3.23.1 Classi

RecipePrivateApi
<pre>+ add_recipe(request : ResourceContainer) : void + remove_recipe(request : ResourceContainer) : void + rate_recipe(request : ResourceContainer) : void</pre>

Figura 121: Classe - server::endpoints::api::private::RecipePrivateApi

3.3.23.1.1 server::endpoints::api::private::RecipePrivateApi

- **Descrizione:** classe che rappresenta l'implementazione delle web API_G relative alla gestione delle Recipe_G;
- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client di aggiungere, rimuovere o valutare una Recipe_G;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::RecipeListResponse
- **Attributi:** N/A
- **Metodi:**
 - `public add_recipe(request : ResourceContainer) : void`
Descrizione: aggiunge una nuova Recipe_G al sistema.
 - `public remove_recipe(request : ResourceContainer) : void`
Descrizione: rimuove una specifica Recipe_G dal sistema.
 - `public rate_recipe(request : ResourceContainer) : void`
Descrizione: aggiunge un rating ad un specifica Recipe_G.

UserApi
<pre>+ user_list() : UserListResponse + user_details(request : ResourceContainer) : UserResponse + add_user(request : ResourceContainer) : void + edit_user(request : ResourceContainer) : void + edit_permission(request : ResourceContainer) : void + remove_user(request : ResourceContainer) : void</pre>

Figura 122: Classe - server::endpoints::api::private::UserApi

3.3.23.1.2 server::endpoints::api::private::UserApi

- **Descrizione:** classe che rappresenta l'implementazione delle web API_G relative alla gestione degli utenti;
- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client di ricavare la lista degli utenti, rimuovere un utente, ricavare o modificare i dati di un utente o per modificare i dati di un utente;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::UserResponse
 - server::endpoints::resp::private::UserListResponse

- **Attributi:** N/A

- **Metodi:**

– `public user_list() : UserListResponse`

Descrizione: restituisce alla chiamata la lista degli utenti registrati al sistema.

– `public user_details(request : ResourceContainer) : UserResponse`

Descrizione: restituisce alla chiamata i dettagli di un utente specifico.

– `public edit_user(request : ResourceContainer) : void`

Descrizione: modifica i dati associati ad un utente registrato nel sistema.

– `public edit_permission(request : ResourceContainer) : void`

Descrizione: modifica i permessi ad un utente registrato al sistema.

– `public remove_user(request : ResourceContainer) : void`

Descrizione: rimuove tutti i dati associati ad un utente registrato nel sistema.



Figura 123: Classe - server::endpoints::api::private::RecipeRequestApi

3.3.23.1.3 server::endpoints::api::private::RecipeRequestApi

- **Descrizione:** classe che rappresenta l'implementazione delle web API_G relative alla gestione delle richieste di aggiunta Recipe_G;

- **Utilizzo:** i suoi metodi vengono invocati quando viene inviata una richiesta di nuova Recipe_G o la rimozione di una richiesta esistente da parte del client;

- **Relazioni con altre classi:**

– server::endpoints::resp::private::RecipeReqListResponse

- **Attributi:** N/A

- **Metodi:**

– `public request_list() : RecipeReqListResponse`

Descrizione: ritorna l'elenco delle richieste di aggiunta Recipe_G presenti nel sistema.

– `public add_request(request : ResourceContainer) : void`

Descrizione: aggiunge una richiesta di aggiunta Recipe_G al sistema.

– `public remove_request(request : ResourceContainer) : void`

Descrizione: rimuove una richiesta di aggiunta Recipe_G dal sistema.



Figura 124: Classe - server::endpoints::api::private::FavouritesApi

3.3.23.1.4 server::endpoints::api::private::FavouritesApi

- **Descrizione:** classe che rappresenta l'implementazione delle web API_G relative alla gestione delle View_G preferite per ogni utente;
- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client l'inserimento di una View_G nei preferiti o quando viene richiesta la rimozione di una View aggiunta in precedenza;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::FavoriteListResponse
- **Attributi:** N/A
- **Metodi:**
 - `public favourites_list() : FavouriteListResponse`
Descrizione: ritorna alla chiamata la lista dei preferiti di uno specifico utente.
 - `public add_favourite(request : ResourceContainer) : void`
Descrizione: aggiunge un nuovo preferito ad un determinato utente.
 - `public remove_favourite(request : ResourceContainer) : void`
Descrizione: rimuove uno specifico preferito da un determinato utente.

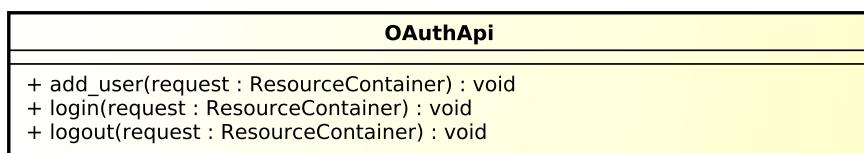


Figura 125: Classe - server::endpoints::resp::private::FavouriteResponse

3.3.23.1.5 server::endpoints::api::private::OAuthApi

- **Descrizione:** classe che rappresenta l'implementazione delle web API_G relative alla registrazione e all'autenticazione dell'utente;
- **Utilizzo:** i suoi metodi vengono invocati quando viene richiesto dal client la registrazione al sistema, quando viene richiesto di autenticarsi o de-autenticarsi;
- **Attributi:** N/A
- **Metodi:**
 - `public add_user(request : ResourceContainer) : void`
Descrizione: aggiunge un utente al sistema.

- `public login(request : ResourceContainer) : UserResponse`
 - Descrizione:** autentica l'utente al sistema e ritorna alla chiamata il token di autenticazione generato.
- `public add_user(request : ResourceContainer) : void`
 - Descrizione:** annulla l'autenticazione dell'utente al sistema rimuovendo il relativo codice di autenticazione.

3.3.24 server::endpoints::resp

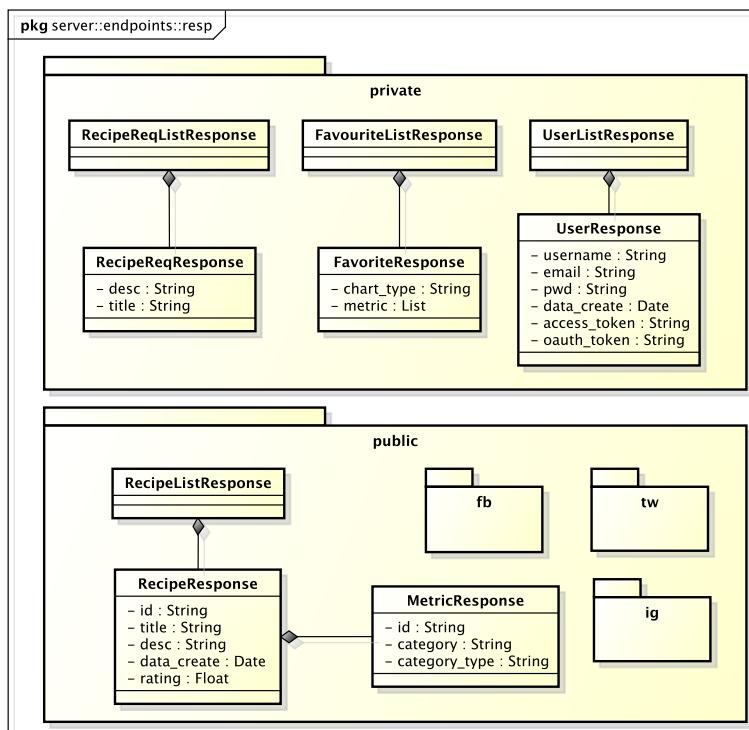


Figura 126: Package - server::endpoints::resp

- **Descrizione:** è il package_G che definisce il modello delle risposte da passare al client in seguito alle chiamate API_G;
- **Padre:** server::endpoints
- **Package_G contenuti:**
 - server::endpoints::resp::public
 - server::endpoints::resp::private

3.3.25 server::endpoints::resp::public

- **Descrizione:** è il package_G che definisce il modello delle risposte da passare al client in seguito alle chiamate delle API_G pubbliche;
- **Padre:** server::endpoints::resp
- **Package_G contenuti:**

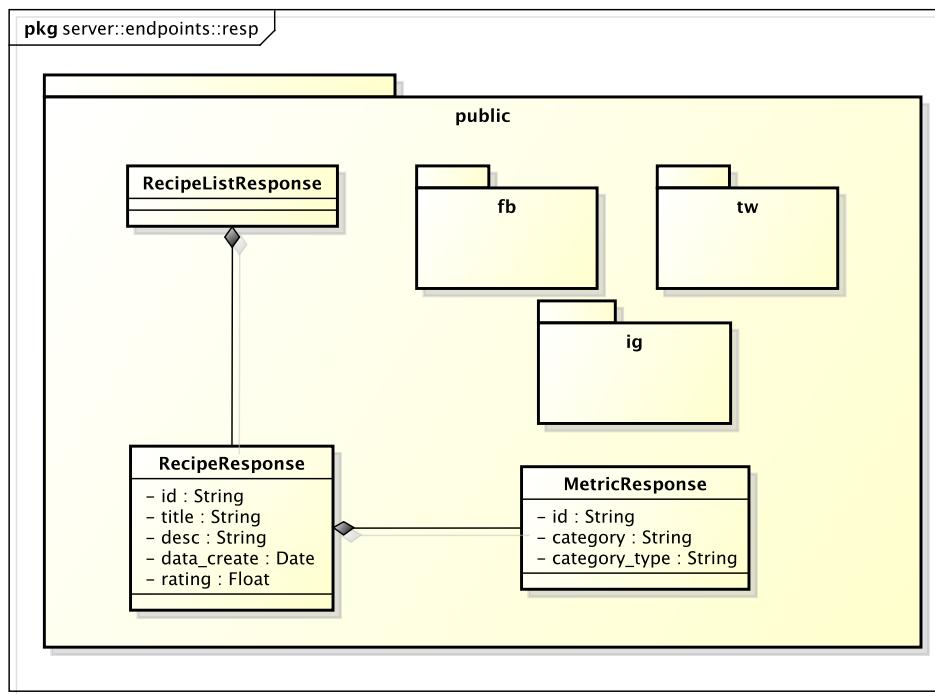


Figura 127: Package - server::endpoints::resp::public

- server::endpoints::resp::public::fb
- server::endpoints::resp::public::tw
- server::endpoints::resp::public::ig

- **Interazione con altri componenti:**

- server::endpoints::resp::private

3.3.25.1 Classi

3.3.25.1.1 server::endpoints::resp::public::MetricResponse

- **Descrizione:** rappresenta il modello dei dati di una metrica_G da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista delle metriche relative ad un Recipe_G al client;

- **Attributi:**

- **id** : String

Descrizione: id della metrica_G.

- **category** : String

Descrizione: categoria_G della metrica_G.

- **category_type** : String

Descrizione: tipo di metrica_G.

- **Metodi:** N/A

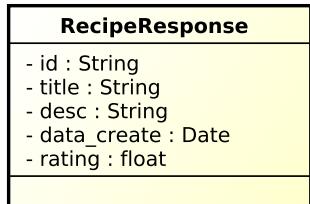


Figura 128: Classe - server::endpoints::resp::public::RecipeResponse

3.3.25.1.2 server::endpoints::resp::public::RecipeResponse

- **Descrizione:** rappresenta il modello dei dati di una Recipe_G da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response di una Recipe_G al client insieme alla lista delle metriche contenute dalla stessa;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::MetricResponse
- **Attributi:**
 - `id : String`
Descrizione: id della Recipe_G.
 - `title : String`
Descrizione: titolo della Recipe_G.
 - `desc : String`
Descrizione: descrizione della Recipe_G.
 - `data_create : Date`
Descrizione: data di creazione della Recipe_G.
 - `rating : Float`
Descrizione: rating della Recipe_G.
- **Metodi:** N/A

3.3.25.1.3 server::endpoints::resp::public::RecipeListResponse

- **Descrizione:** rappresenta il modello dei dati di una lista di Recipe_G da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response di una lista di Recipe_G al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::RecipeResponse
- **Attributi:** N/A
- **Metodi:** N/A

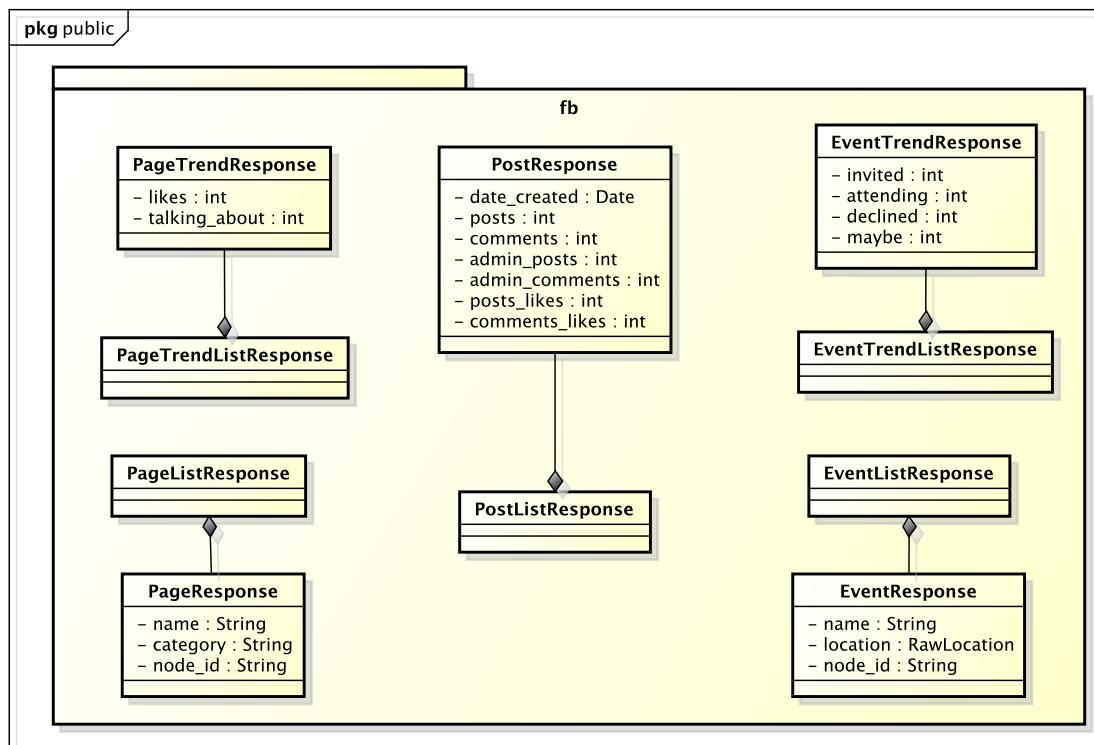


Figura 129: Package - server::endpoints::resp::public::fb

3.3.26 server::endpoints::resp::public::fb

- **Descrizione:** è il package_G contenente le classi che rappresentano il modello di dati delle componenti di Facebook da restituire al client;
- **Padre:** server::endpoints::resp::public

3.3.26.1 Classi

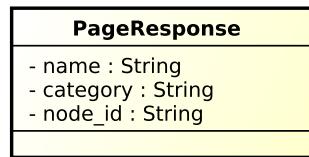


Figura 130: Classe - server::endpoints::resp::public::fb::PageResponse

3.3.26.1.1 server::endpoints::resp::public::fb::PageResponse

- **Descrizione:** rappresenta il modello dei dati di una pagina Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati di una pagina Facebook al client;
- **Attributi:**

– `name : String`

Descrizione: nome della pagina Facebook.

- `category : String`

Descrizione: categoria_G della pagina Facebook.

- `node_id : String`

Descrizione: id della pagina Facebook.

- **Metodi:** N/A

3.3.26.1.2 server::endpoints::resp::public::fb::PageListResponse

- **Descrizione:** rappresenta il modello dei dati della lista di pagine Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista delle pagine Facebook al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::fb::PageResponse
- **Attributi:** N/A
- **Metodi:** N/A



Figura 131: Classe - server::endpoints::resp::public::fb::PageTrendResponse

3.3.26.1.3 server::endpoints::resp::public::fb::PageTrendResponse

- **Descrizione:** rappresenta il modello dei dati dei trend di una pagina Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei trend di una pagina Facebook al client;
- **Attributi:**
 - `likes : int`
Descrizione: likes della pagina Facebook.
 - `talking_about : int`
Descrizione: talking about della pagina Facebook.
- **Metodi:** N/A

3.3.26.1.4 server::endpoints::resp::public::fb::PageTrendListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei trend di una pagina Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei trend di una pagina Facebook al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::fb::PageTrendResponse
- **Attributi:** N/A
- **Metodi:** N/A

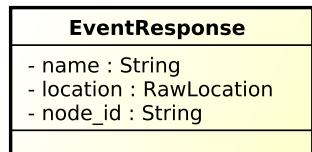


Figura 132: Classe - server::endpoints::resp::public::fb::EventResponse

3.3.26.1.5 server::endpoints::resp::public::fb::EventResponse

- **Descrizione:** rappresenta il modello dei dati di un evento di Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati di un evento Facebook al client;
- **Attributi:**
 - `name : String`
Descrizione: nome dell'evento Facebook.
 - `location : String`
Descrizione: location dell'evento Facebook.
 - `node_id : String`
Descrizione: id dell'evento Facebook.
- **Metodi:** N/A

3.3.26.1.6 server::endpoints::resp::public::fb::EventListResponse

- **Descrizione:** rappresenta il modello dei dati della lista di eventi Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista di eventi Facebook al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::fb::EventResponse
- **Attributi:** N/A
- **Metodi:** N/A

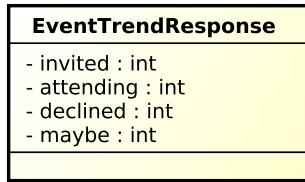


Figura 133: Classe - server::endpoints::resp::public::fb::EventTrendResponse

3.3.26.1.7 server::endpoints::resp::public::fb::EventTrendResponse

- **Descrizione:** rappresenta il modello dei dati dei trend di un evento Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei trend di un evento Facebook al client;
- **Attributi:**
 - **invited** : int
Descrizione: numero di invitati all'evento Facebook.
 - **attending** : int
Descrizione: numero di partecipanti all'evento Facebook.
 - **declined** : int
Descrizione: numero di utenti non partecipanti all'evento Facebook.
 - **maybe** : int
Descrizione: numero di utenti con la partecipazione in forse all'evento Facebook.
- **Metodi:** N/A

3.3.26.1.8 server::endpoints::resp::public::fb::EventTrendListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei trend di un evento Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei trend di un evento Facebook al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::fb::EventTrendResponse
- **Attributi:** N/A
- **Metodi:** N/A

3.3.26.1.9 server::endpoints::resp::public::fb::PostResponse

- **Descrizione:** rappresenta il modello dei dati dei post di Facebook da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati dei post Facebook al client;

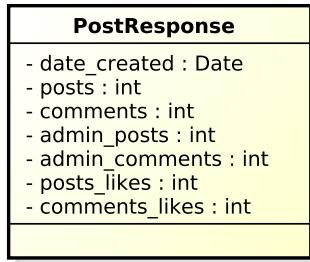


Figura 134: Classe - server::endpoints::resp::public::fb::PostResponse

- **Attributi:**

- `posts` : int

Descrizione: numero totale dei post di una pagina o evento Facebook.

- `comments` : int

Descrizione: numero totale dei commenti ai post di una pagina o evento Facebook.

- `admin_posts` : int

Descrizione: numero totale dei post di una pagina Facebook ed effettuati dalla pagina stessa.

- `admin_comments` : int

Descrizione: numero totale dei commenti ai post di una pagina Facebook ed effettuati dalla pagina stessa.

- `posts_likes` : int

Descrizione: numero totale dei likes ai post di una pagina o evento Facebook.

- `comments_likes` : int

Descrizione: numero totale dei likes ai commenti dei post di una pagina o evento Facebook.

- **Metodi:** N/A

3.3.26.1.10 server::endpoints::resp::public::fb::PostListResponse

- **Descrizione:** rappresenta il modello dei dati della lista di post di Facebook da ritornare al client;

- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista di post di Facebook al client;

- **Relazioni con altre classi:**

- server::endpoints::resp::public::fb::PostResponse

- **Attributi:** N/A

- **Metodi:** N/A

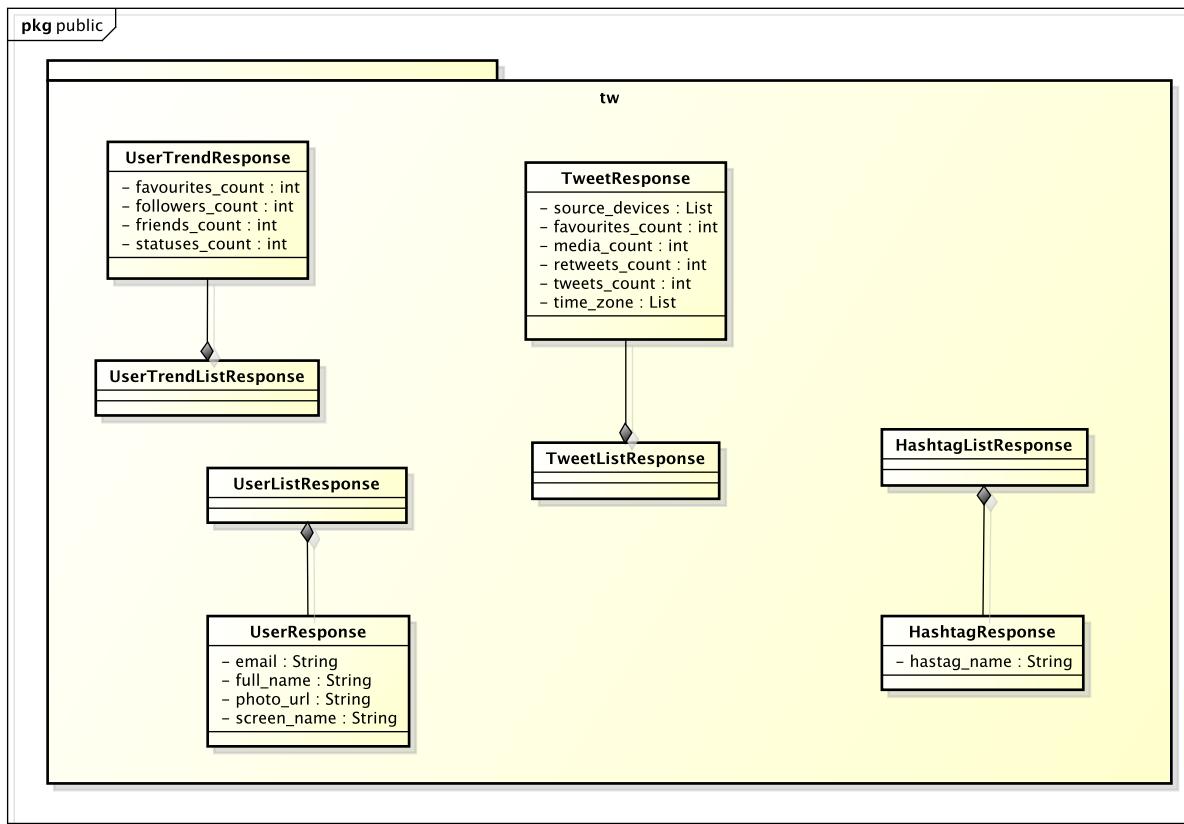


Figura 135: Package - server::endpoints::resp::public::tw

3.3.27 server::endpoints::resp::public::tw

- **Descrizione:** è il package_G contenente le classi che rappresentano il modello dei dati delle componenti di Twitter da restituire al client;
- **Padre:** server::endpoints::resp::public

3.3.27.1 Classi

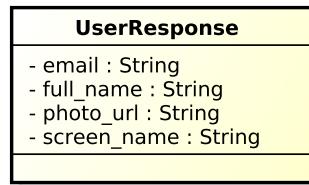


Figura 136: Classe - server::endpoints::resp::public::tw::UserResponse

3.3.27.1.1 server::endpoints::resp::public::tw::UserResponse

- **Descrizione:** rappresenta il modello dei dati di un profilo Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati di un profilo Twitter al client;
- **Attributi:**

- `email : String`
Descrizione: email dell’utente Twitter.
- `full_name : String`
Descrizione: nome completo dell’utente Twitter.
- `photo_url : String`
Descrizione: url della foto profilo dell’utente Twitter.
- `screen_name : String`
Descrizione: username dell’utente Twitter.

- **Metodi:** N/A

3.3.27.1.2 server::endpoints::resp::public::tw::UserListResponse

- **Descrizione:** rappresenta il modello dei dati della lista di profili Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista di profili Twitter al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::tw::UserResponse
- **Attributi:** N/A
- **Metodi:** N/A

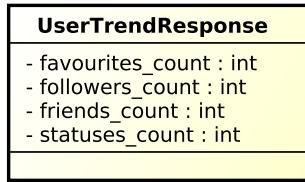


Figura 137: Classe - server::endpoints::resp::public::tw::UserTrendResponse

3.3.27.1.3 server::endpoints::resp::public::tw::UserTrendResponse

- **Descrizione:** rappresenta il modello dei dati dei trend di un profilo Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei trend di un profilo Twitter al client;
- **Attributi:**
 - `favourites_count : int`
Descrizione: numero totale di preferiti aggiungi dall’utente Twitter.
 - `followers_count : int`
Descrizione: numero totale dei followers dell’utente Twitter.
 - `friends_count : int`
Descrizione: numero totale dei following dell’utente Twitter.
 - `statuses_count : int`
Descrizione: numero totale dei tweet dell’utente Twitter.
- **Metodi:** N/A

3.3.27.1.4 server::endpoints::resp::public::tw::UserTrendListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei trend di un profilo Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei trend di un profilo Twitter al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::tw::UserTrendResponse
- **Attributi:** N/A
- **Metodi:** N/A



Figura 138: Classe - server::endpoints::resp::public::tw::HashtagResponse

3.3.27.1.5 server::endpoints::resp::public::tw::HashtagResponse

- **Descrizione:** rappresenta il modello dei dati di un hashtag di Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati di un hashtag di Twitter al client;
- **Attributi:**
 - `hashtag_name : String`

Descrizione: nome dell'hashtag Twitter.
- **Metodi:** N/A

3.3.27.1.6 server::endpoints::resp::public::tw::HashtagListResponse

- **Descrizione:** rappresenta il modello dei dati della lista degli hashtag di Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista degli hashtag di Twitter al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::tw::HashtagResponse
- **Attributi:** N/A
- **Metodi:** N/A

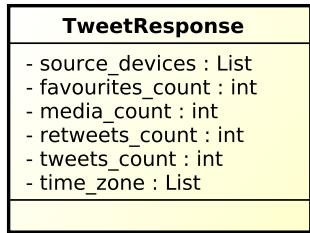


Figura 139: Classe - server::endpoints::resp::public::tw::TweetResponse

3.3.27.1.7 server::endpoints::resp::public::tw::TweetResponse

- **Descrizione:** rappresenta il modello dei dati dei tweet di Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati dei tweet di Twitter al client;
- **Attributi:**
 - `source_devices : List`
Descrizione: tipo ed occorrenza dei dispositivi raccolti dai tweet di un utente o di un hashtag Twitter.
 - `favourites_count : int`
Descrizione: numero totale dei preferiti dei tweet di un utente o di un hashtag Twitter.
 - `media_count : int`
Descrizione: numero totale dei media dei tweet di un utente o di un hashtag Twitter.
 - `retweets_count : int`
Descrizione: numero totale dei retweet dei tweet di un utente o di un hashtag Twitter.
 - `tweets_count : int`
Descrizione: numero totale dei tweet di un utente o di un hashtag Twitter.
 - `time_zone : List`
Descrizione: valore ed occorrenza delle timezone raccolte dai tweet relativi ad un hashtag Twitter.
- **Metodi:** N/A

3.3.27.1.8 server::endpoints::resp::public::tw::TweetListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei tweet Twitter da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei tweet di Twitter al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::tw::TweetResponse

- **Attributi:** N/A
- **Metodi:** N/A

3.3.28 server::endpoints::resp::public::ig

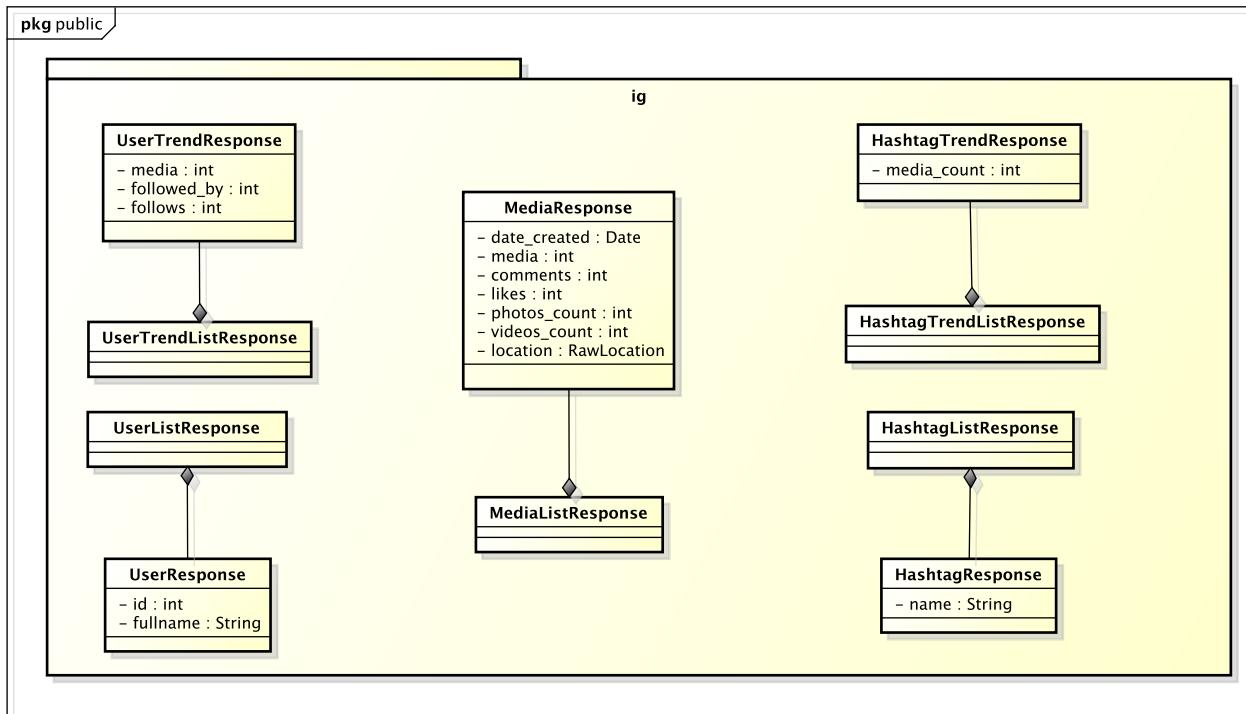


Figura 140: Package - server::endpoints::resp::public::ig

- **Descrizione:** è il package_G contenente le classi che rappresentano il modello dei dati delle componenti di Instagram da restituire al client;
- **Padre:** server::endpoints::resp::public

3.3.28.1 Classi

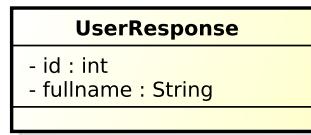


Figura 141: Classe - server::endpoints::resp::public::ig::UserResponse

3.3.28.1.1 server::endpoints::resp::public::ig::UserResponse

- **Descrizione:** rappresenta il modello dei dati di un profilo Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati di un profilo Instagram al client;
- **Attributi:**

- `id : int`

Descrizione: id dell’utente Instagram.

- `fullname : String`

Descrizione: nome completo dell’utente Instagram.

- **Metodi:** N/A

3.3.28.1.2 server::endpoints::resp::public::ig::UserListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei profili di Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei profili di Instagram al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::ig::UserResponse
- **Attributi:** N/A
- **Metodi:** N/A

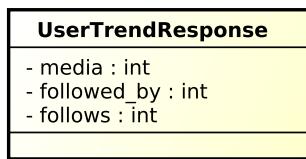


Figura 142: Classe - server::endpoints::resp::public::ig::UserTrendResponse

3.3.28.1.3 server::endpoints::resp::public::ig::UserTrendResponse

- **Descrizione:** rappresenta il modello dei dati dei trend di un profilo Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei trend di un profilo Instagram al client;
- **Attributi:**
 - `media : int`

Descrizione: numero di media dell’utente Instagram.
 - `followed_by : int`

Descrizione: numero di followed dell’utente Instagram.
 - `follows : int`

Descrizione: numero di following dell’utente Instagram.
- **Metodi:** N/A

3.3.28.1.4 server::endpoints::resp::public::ig::UserTrendListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei trend di un profilo Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei trend di un profilo Instagram al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::ig::UserTrendResponse
- **Attributi:** N/A
- **Metodi:** N/A

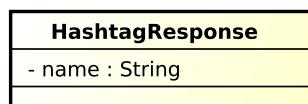


Figura 143: Classe - server::endpoints::resp::public::ig::HashtagResponse

3.3.28.1.5 server::endpoints::resp::public::ig::HashtagResponse

- **Descrizione:** rappresenta il modello dei dati di un hashtag di Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati di un hashtag di Instagram al client;
- **Attributi:**

– `name : String`

Descrizione: nome dell'hashtag Instagram.

- **Metodi:** N/A

3.3.28.1.6 server::endpoints::resp::public::ig::HashtagListResponse

- **Descrizione:** rappresenta il modello dei dati della lista degli hashtag di Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista degli hashtag di Instagram al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::ig::HashtagResponse
- **Attributi:** N/A
- **Metodi:** N/A



Figura 144: Classe - server::endpoints::resp::public::ig::HashtagTrendResponse

3.3.28.1.7 server::endpoints::resp::public::ig::HashtagTrendResponse

- **Descrizione:** rappresenta il modello dei dati dei trend di un hashtag di Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei trend di un hashtag di Instagram al client;
- **Attributi:**

– `media_count : int`

Descrizione: numero di media totali relativi ad un determinato hashtag Instagram.

- **Metodi:** N/A

3.3.28.1.8 server::endpoints::resp::public::ig::HashtagTrendListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei trend di un hashtag di Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei trend di un hashtag di Instagram al client;
- **Relazioni con altre classi:**

– server::endpoints::resp::public::ig::HashtagTrendResponse

- **Attributi:** N/A

- **Metodi:** N/A

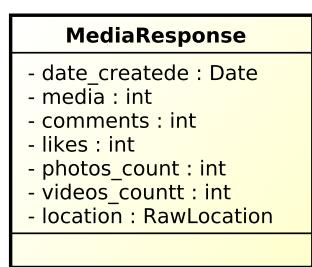


Figura 145: Classe - server::endpoints::resp::public::ig::MediaResponse

3.3.28.1.9 server::endpoints::resp::public::ig::MediaResponse

- **Descrizione:** rappresenta il modello dei dati dei media di Instagram da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response dei dati dei media di Instagram al client;

- **Attributi:**

- `date_created : Date`

Descrizione: data di creazione dell'oggetto di trend.

- `media : int`

Descrizione: numero dei media relativi un determinato utente o hash-tag Instagram.

- `comments : int`

Descrizione: numero dei commenti relativi ai media di un determinato utente o hashtag Instagram.

- `likes : int`

Descrizione: numero dei likes relativi ai media di un determinato utente o hashtag Instagram.

- `photos_count : int`

Descrizione: numero di foto relative ad un determinato utente o hash-tag Instagram.

- `videos_count : int`

Descrizione: numero di video relativi ad un determinato utente o hashtag Instagram.

- `location : List`

Descrizione: lista delle location relative ai media di un determinato utente o hashtag Instagram.

- **Metodi:** N/A

3.3.28.1.10 server::endpoints::resp::public::ig::MediaListResponse

- **Descrizione:** rappresenta il modello dei dati della lista dei media di Instagram da ritornare al client;

- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista dei media di Instagram al client;

- **Relazioni con altre classi:**

- server::endpoints::resp::public::ig::MediaResponse

- **Attributi:** N/A

- **Metodi:** N/A

3.3.29 server::endpoints::resp::private

- **Descrizione:** è il package_G che definisce i modelli delle risposte da passare al client in seguito alle chiamate API_G;

- **Padre:** server::endpoints::resp

- **Interazione con altri componenti:**

- server::endpoints::resp::public

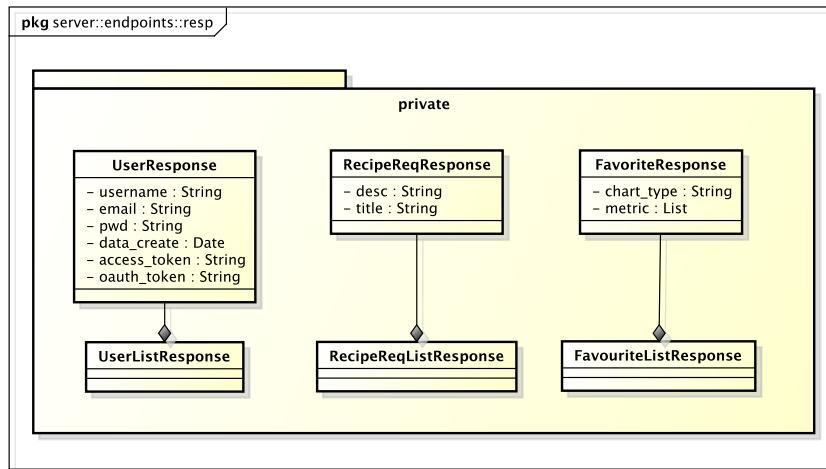


Figura 146: Package - server::endpoints::resp::private

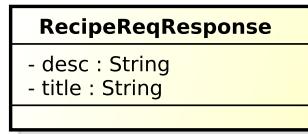


Figura 147: Classe - server::endpoints::resp::private::RecipeReqResponse

3.3.29.1 Classi

3.3.29.1.1 server::endpoints::resp::private::RecipeReqResponse

- **Descrizione:** rappresenta il modello dei dati di una richiesta di Recipe_G da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response di una richiesta di Recipe_G al client;
- **Attributi:**
 - `desc : String`
Descrizione: descrizione della richiesta di aggiunta Recipe_G.
 - `title : String`
Descrizione: titolo della richiesta di aggiunta Recipe_G.
- **Metodi:** N/A

3.3.29.1.2 server::endpoints::resp::private::RecipeReqListResponse

- **Descrizione:** rappresenta il modello dei dati di una lista di richieste di Recipe_G da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response di una lista di richieste di Recipe_G al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::RecipeReqResponse
- **Attributi:** N/A
- **Metodi:** N/A

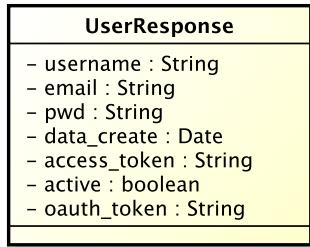


Figura 148: Classe - server::endpoints::resp::private::UserResponse

3.3.29.1.3 server::endpoints::resp::private::UserResponse

- **Descrizione:** rappresenta il modello dei dati di un utente da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response contenente i dati un utente al client;
- **Attributi:**

– `username : String`

Descrizione: username dell’utente registrato al sistema.

– `email : String`

Descrizione: email dell’utente registrato al sistema.

– `pwd : String`

Descrizione: password dell’utente registrato al sistema.

– `data_create : Date`

Descrizione: data di creazione dell’utente registrato al sistema.

– `access_token : String`

Descrizione: access token dell’utente registrato al sistema.

– `oauth_token: String`

Descrizione: codice di autenticazione dell’utente registrato al sistema.

- **Metodi:** N/A

3.3.29.1.4 server::endpoints::resp::private::UserListResponse

- **Descrizione:** rappresenta il modello dei dati della lista degli utenti registrati da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response della lista degli utenti al client;
- **Attributi:** N/A
- **Metodi:** N/A



Figura 149: Classe - server::endpoints::resp::private::FavouriteResponse

3.3.29.1.5 server::endpoints::resp::private::FavoriteResponse

- **Descrizione:** rappresenta il modello dei dati di una View_G preferita da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response di una View_G preferita al client;
- **Attributi:**
 - `chart_type : String`
Descrizione: impostazione utilizzata dal client per generare il grafico che l'utente ha aggiunto ai preferiti.
 - `metric : List`
Descrizione: metriche utilizzate nel grafico che l'utente ha aggiunto ai preferiti.
- **Metodi:** N/A

3.3.29.1.6 server::endpoints::resp::private::FavoriteListResponse

- **Descrizione:** rappresenta il modello dei dati di una lista di View_G preferite da ritornare al client;
- **Utilizzo:** viene utilizzata dalle API_G per restituire la response di una lista di View_G preferite al client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::FavoriteResponse
- **Attributi:** N/A
- **Metodi:** N/A

3.4 Database

Il database_G utilizzato, sia per quanto riguarda i dati grezzi sia per quanto riguarda i dati aggregati e le varie configurazioni, sarà di tipo schema-less_G. Questo significa che non c'è un diagramma di come i dati siano in relazione tra loro.

Il modo quindi nel quale saranno salvati nel database i dati rispecchierà il formato che viene descritto dal Model dell'applicazione come illustrato alla sezione 3.3.2.

4 Diagrammi di sequenza

Vengono riportati di seguito i diagrammi di sequenza delle operazioni principali dell'applicazione. Abbiamo mantenuto la divisione client server per una maggior chiarezza dei diagrammi e per sottolineare la loro indipendenza.

4.1 Client

Quasi tutte le operazioni offerte dall'applicazione comportano lo stesso tipo di interazione tra le classi presenti nella componente client. Ne sono quindi state descritte alcune ritenute tra le più significative.

Questi diagrammi riguardano solo la parte appunto del client. La comunicazione quindi con il server e le operazioni che esso esegue per ritornare dei valori al client viene mostrata nella sezione 4.2.

4.1.1 Login

Il seguente diagramma di sequenza descrive le collaborazioni tra gli oggetti del front-end dell'applicazione durante l'operazione di autenticazione.

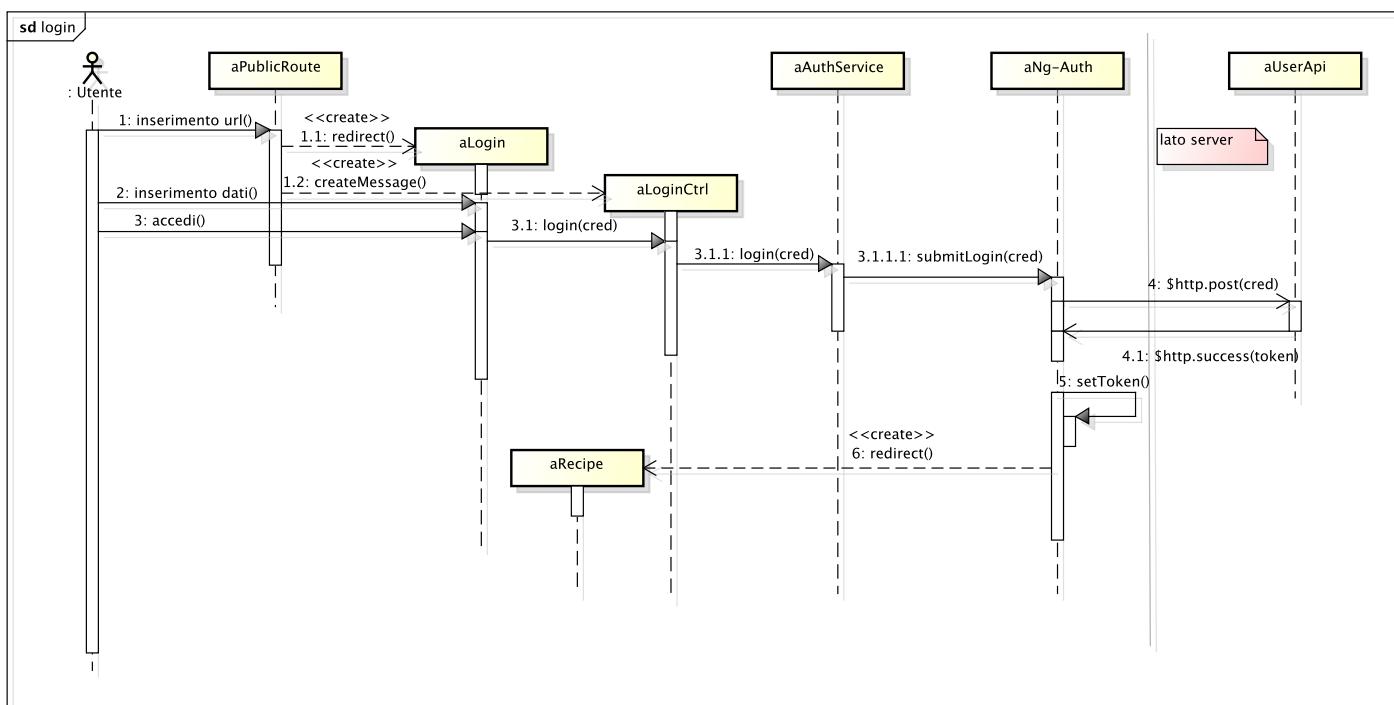


Figura 150: Diagramma di sequenza - Login lato client

Il diagramma descrive le interazioni tra gli oggetti che compongono il front-end quando un utente vuole effettuare il login nell'applicazione.

- **Utente non autenticato:** l'oggetto Utente non autenticato individua un qualsiasi attore che vuole effettuare l'accesso al sistema. L'utente inserisce i propri dati e preme il bottone **Login**;
- **aPublicRoute:** l'oggetto aPublicRoute re-indirizza l'utente alla corretta pagina, dalla quale potrà effettuare l'accesso;
- **aLogin (login.html):** l'oggetto aLogin è la vera pagina che permette l'autenticazione dell'utente. Qui verranno inserite le credenziali per accedere e sarà presente il pulsante che una volta premuto invocherà il metodo **login(cred)**;
- **aLoginCtrl:** l'oggetto aLoginCtrl è il controller che si occupa di gestire le funzioni disponibili in **aLogin**, in particolare di **login(cred)**. Questo metodo

invoca ulteriormente il metodo **login(cred)** dell'oggetto **aAuthService** passandogli come parametro un oggetto JSON_G contenente le credenziali recuperate dal form $_G$ di inserimento;

- **aAuthService**: l'oggetto aAuthService chiama il metodo **submitLogin(cred)** del servizio aNg-Auth
- **aNg-Auth**: l'oggetto aNg-Auth rappresenta un modulo esterno, iniettato come servizio tramite Dependency Injection, che gestisce le principali azioni di un utente per quando riguarda autenticazione, registrazione, cambio dati personali, etc. In questo caso viene utilizzato per autenticare un utente al sistema. Una volta ricevuta risposta dal server, viene immagazzinato un access token riguardante la sessione in corso. Attraverso quel token si potranno recuperare tutte le informazioni dell'utente ed effettuare determinate azioni previste dall'applicativo;
- **aRecipe (recipe $_G$.html)**: l'oggetto aRecipe è la pagina nella quale l'utente che ha inserito le credenziali corrette viene re-indirizzato.
- **aUserApi**: l'oggetto aUserApi viene usato in questo diagramma per rappresentare il Server, che riceve la chiamata REST $_G$ dal client e restituisce in risposta i dati richiesti.

4.1.2 Visualizzazione delle Recipe

Il seguente diagramma di sequenza descrive le collaborazioni tra gli oggetti del front-end dell'applicazione durante la visualizzazione delle Recipe $_G$.

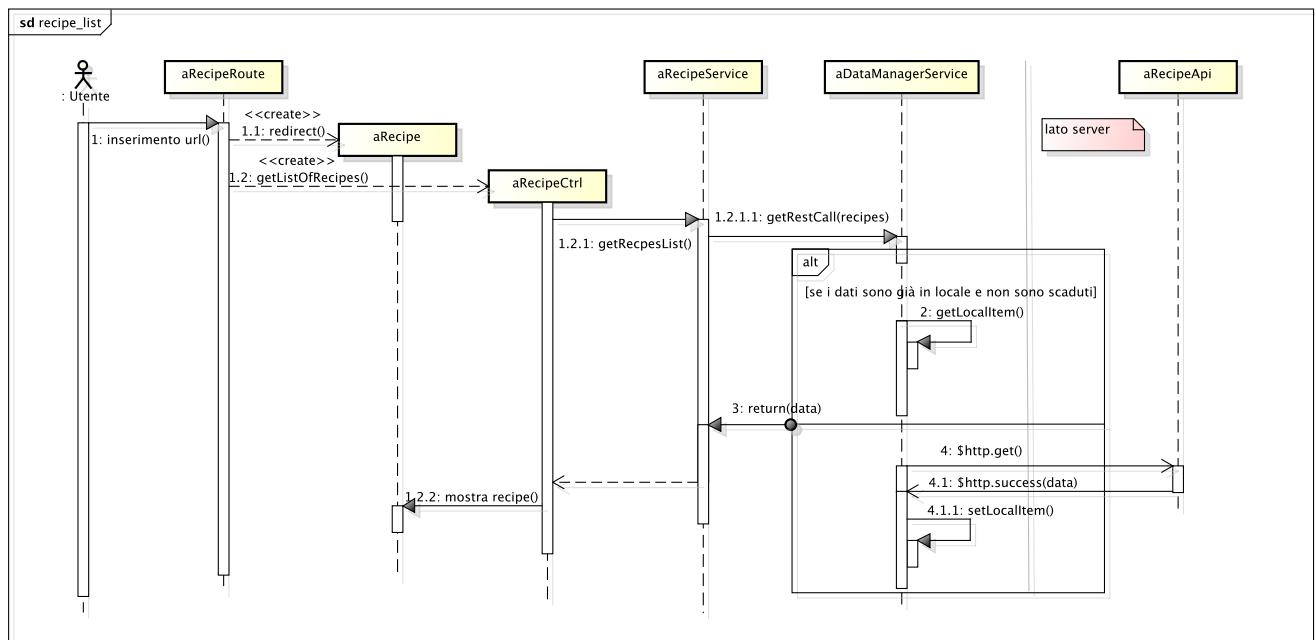


Figura 151: Diagramma di sequenza - Visualizzazione delle Recipe

Il diagramma descrive le interazioni tra gli oggetti che compongono il front-end quando un utente vuole visualizzare la lista delle Recipe $_G$ presenti nel sistema.

- **Utente autenticato:** l'oggetto Utente individua un utente che ha effettuato l'accesso al sistema;
- **aRecipeRoute:** l'oggetto aRecipeRoute re-indirizza l'utente alla corretta pagina, dalla quale potrà visualizzare la lista di tutte le recipe_G;
- **aRecipe (recipe_G.html):** l'oggetto aRecipe è la vera pagina che permette la visualizzazione di tutte le recipe presenti nel sistema. La schermata viene riempita grazie all'invocazione in automatico del metodo **getListOfRecipes()**;
- **aRecipeCtrl:** l'oggetto aRecipeCtrl è il controller che si occupa di gestire le funzioni disponibili in **aRecipe**, in particolare del metodo **getListOfRecipes()** che chiama ulteriormente il metodo **getRecipesList()** dell'oggetto **aRecipeService**;
- **aRecipeService:** l'oggetto aRecipeService attraverso la chiamata al metodo **getRestCall('recipes')** del servizio **aDataManagerService** fa ritornare una promise all'oggetto **aRecipeCtrl** che verrà risolta quando la chiamata sarà eseguita correttamente;
- **aDataManagerService:** l'oggetto aDataManagerService riceve le direttive da impostare per effettuare la chiamata al server. Prima di effettuare la chiamata vera e propria però controlla se i dati richiesti non siano già presenti nel localStorage o che non siano troppo vecchi, attraverso l'invocazione del metodo **getLocalItem(key)** dove il parametro key rappresenta l'indirizzo della chiamata http, che viene usato come chiave per salvare i dati in locale. Se i dati sono presenti gli restituisce incapsulandoli in una promise che verrà risolta immediatamente, altrimenti effettua la chiamata vera e proprio invocando il metodo **httpGetRequest** dell'oggetto stesso. Una volta ricevuti i dati, il metodo **setLocalItem(key, value)** provvederà a salvarli in locale;
- **aRecipeApi:** l'oggetto aRecipeApi viene usato in questo diagramma per rappresentare il Server, che riceve la chiamata REST_G dal client e restituisce in risposta i dati richiesti.

4.1.3 Generazione dei grafici

Il seguente diagramma di sequenza descrive le collaborazioni tra gli oggetti del front-end dell'applicazione durante la generazione di grafici da una metrica_G.

Il diagramma descrive le interazioni tra gli oggetti che compongono il front-end quando devono essere generati i grafici relativi ad una specifica metrica_G.

- **aChartCtrl:** l'oggetto aChartCtrl è il controller che si occupa di gestire le funzioni di visualizzazione di grafici, e che ne richiede la generazione quando è necessario;
- **aRecipeService:** l'oggetto aRecipeService attraverso la chiamata al metodo **getRestCall('metrics/metric_id')** del servizio **aDataManagerService** fa ritornare una promise all'oggetto **aRecipeCtrl** che verrà risolta quando la chiamata sarà eseguita correttamente. Successivamente tramite la chiamata al metodo **getViews('false', metric)** otterrà dall'oggetto **aViewTypeModel** quali tipi di grafici dovranno essere generati. Ottenuta questa informazione attraverso la chiamata **chartGeneration(data)** avente come parametro le

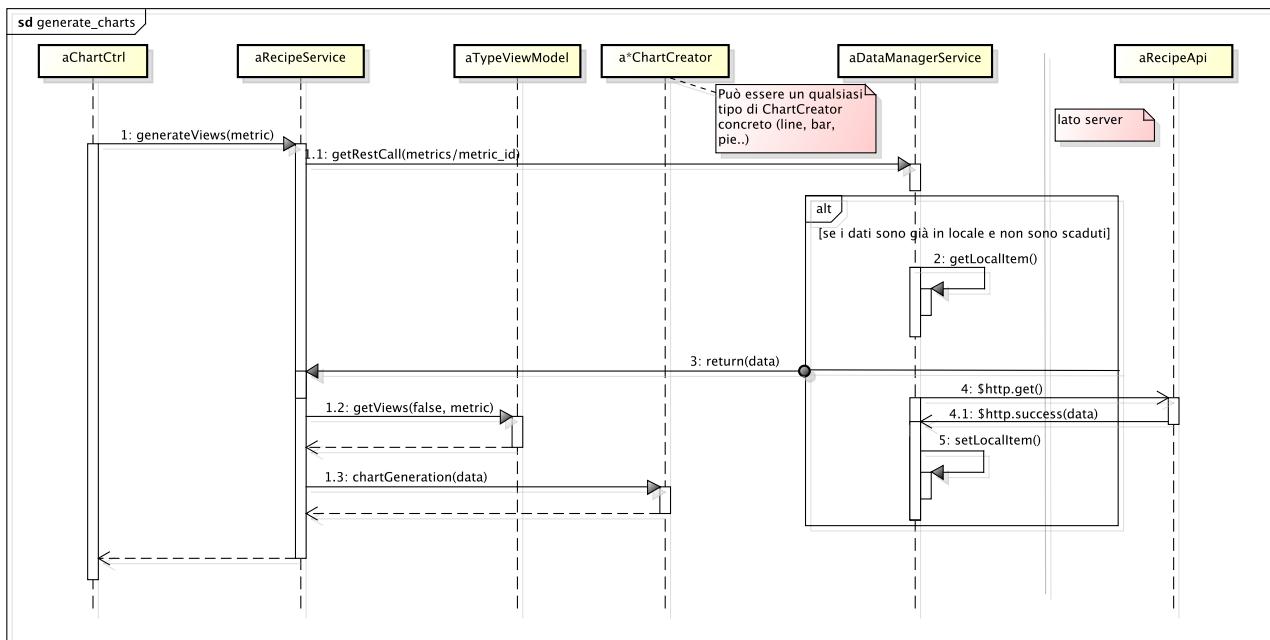


Figura 152: Diagramma di sequenza - Generazione dei grafici

informazioni della metrica_G e quelle ottenute dall'oggetto **aViewTypeModel** otterrà i grafici generati, che poi potrà restituire all'oggetto **aChartCtrl**;

- **aDataManagerService**: l'oggetto aDataManagerService riceve le direttive da impostare per effettuare la chiamata al server. Prima di effettuare la chiamata vera e propria però controlla se i dati richiesti non siano già presenti nel localStorage o che non siano troppo vecchi, attraverso l'invocazione del metodo **getLocalItem(key)** dove il parametro key rappresenta l'indirizzo della chiamata http, che viene usato come chiave per salvare i dati in locale. Se i dati sono presenti gli restituisce incapsulandoli in una promise che verrà risolta immediatamente, altrimenti effettua la chiamata vera e proprio invocando il metodo **httpGetRequest** dell'oggetto stesso. Una volta ricevuti i dati, il metodo **setLocalItem(key, value)** provvederà a salvarli in locale;
- **aTypeViewModel**: l'oggetto aTypeViewModel restituisce all'oggetto **aRecipeService** i tipi di grafici che devono essere generati a partire dalla metrica_G che riceve quando viene chiamato il suo metodo **getViews('false', metric)**;
- **a*ChartCreator**: l'oggetto a*ChartCreator è in realtà un'istanza di una delle classi figlie di ChartCreator. Questo restituisce i grafici generati a partire dai dati che riceve nella chiamata al suo metodo **chartGeneration(data)** effettuata da **aRecipeService**;
- **aRecipeApi**: l'oggetto aRecipeApi viene usato in questo diagramma per rappresentare il Server, che riceve la chiamata REST_G dal client e restituisce in risposta i dati richiesti.

4.2 Server

Quelli che seguono sono i diagrammi di sequenza che rappresentano le più rilevanti interazioni tra i vari moduli presenti nel livello server. Per quanto riguarda il client, fare riferimento alla sezione 4.1.

4.2.1 Cron task

Il seguente diagramma descrive il processo di avvio dell'aggiornamento dei dati grezzi relativi ad ogni Recipe_G presente nel sistema. Tale sequenza è attivata a cadenza regolare tramite le funzionalità di *task_G scheduling* offerte da Google App Engine_G. Per semplicità, è stato descritto in dettaglio esclusivamente l'aggiornamento dei dati relativi a Facebook, indicando solamente i metodi principali per i restanti social network.

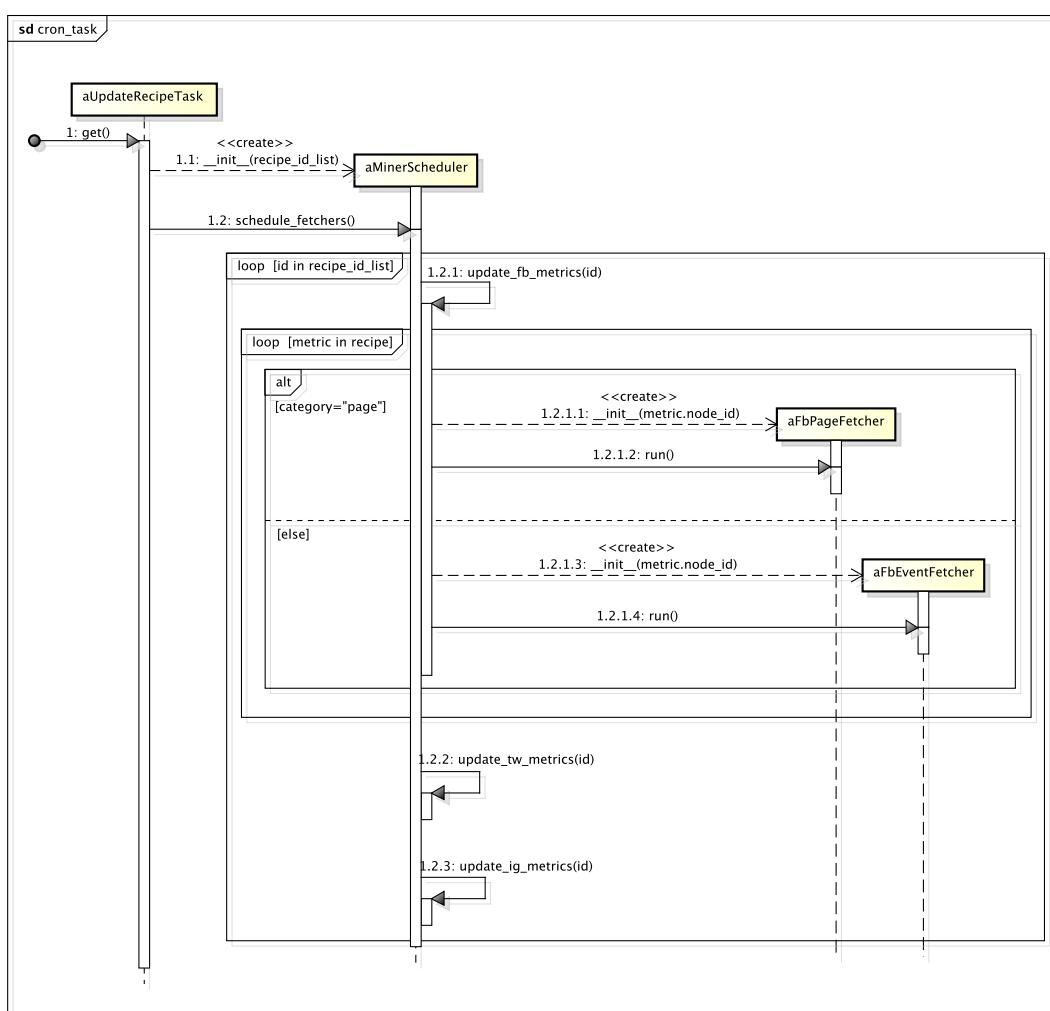


Figura 153: Diagramma di sequenza - Cron task

- **UpdateRecipeTask:** classe implementata secondo gli standard di Google App Engine_G, definisce il metodo statico `get()` attivato ad ogni avvio del componente Cron_G di Google. `UpdateRecipeTask` istanzia la classe `MinerManager` invocando il metodo `schedule_fetchers(recipeG_id_list)` per effettuare l'aggiornamento dei dati grezzi;

- **MinerScheduler:** definisce il metodo `schedule_fetchers(recipe_id_list)` che cicla la lista di id delle Recipe_G scadute invocando per ognuna il metodo `update_fb_metrics(id)` per ogni metrica_G Facebook contenuta nella Recipe in questione. Tale metodo avvia un fetcher differente a seconda del tipo di metrica ricavata. Tale procedimento avviene sequenzialmente anche per gli altri social tramite i metodi `update_tw_metrics(id)` e `update_ig_metrics(id)`;
- **FbPageFetcher:** classe che viene istanziata ed avviata tramite il metodo `run()` per avviare l'aggiornamento dei dati relativi ad una pagina Facebook;
- **FbEventFetcher:** classe che viene istanziata ed avviata tramite il metodo `run()` per avviare l'aggiornamento dei dati relativi ad un evento Facebook.

4.2.2 Miner Facebook

Il diagramma alla figura 154 descrive l'aggiornamento dei dati, relativi ad una metrica che descrive una pagina di Facebook, a seguito della chiamata del cron che richiede l'aggiornamento della Recipe a cui la metrica appartiene. Tale sequenza verrà attivata ogni qualvolta una metrica che descrive una pagina di Facebook dovrà essere aggiornata.

- **FbPageFetcher:** classe richiamata dalla classe `MinerScheduler` e contenente i metodi `get_data()` e `get_trend_data()`. Il primo metodo si occupa di recuperare i dati statici di una pagina Facebook se non ancora presente nel database_G; il secondo metodo si occupa di recuperare i dati dinamici della pagina;
- **RawFbPage:** classe utilizzata dal metodo `get_data()` della classe `FbPageFetcher` per salvare i dati statici di una pagina Facebook tramite il metodo `put()`;
- **RawFbPageTrend:** classe utilizzata dal metodo `get_trend_data()` della classe `FbPageFetcher` per salvare i dati dinamici di una pagina Facebook tramite il metodo `put()`;
- **PostCounter:** classe utilizzata dal metodo `get_trend_data()` della classe `FbPageFetcher` per effettuare il conteggio dei post di una pagina nell'arco di tre giorni e per calcolarsi i vari trend relativi ai post;
- **CommentCounter:** classe utilizzata dal metodo `action()` della classe `PostCounter` per effettuare il conteggio dei commenti in un post tramite il metodo `count()`;
- **RawFbPostTrend:** classe utilizzata dal metodo `get_trend_data()` della classe `FbPageFetcher` per salvare i dati dinamici relativi ai post all'interno di una pagina Facebook tramite il metodo `put()`.

4.2.3 Login API

Il diagramma nella figura 155 descrive il funzionamento del servizio REST di autenticazione offerto dal sistema. Il client tramite il metodo `login()` della classe `DataManagerService` utilizza il servizio REST di autenticazione definito nella classe `OAuthApi`.

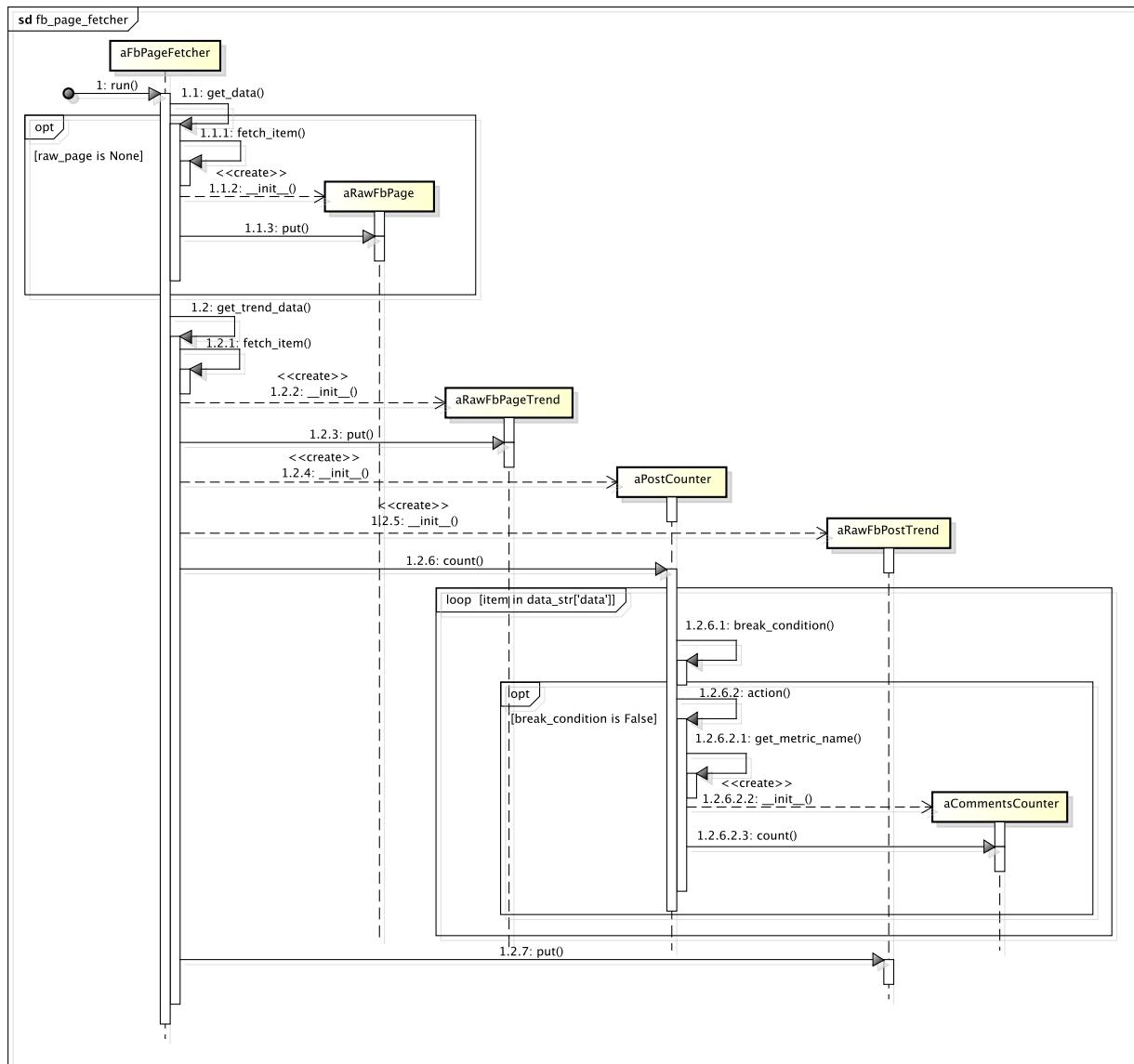


Figura 154: Diagramma di sequenza - Miner Facebook

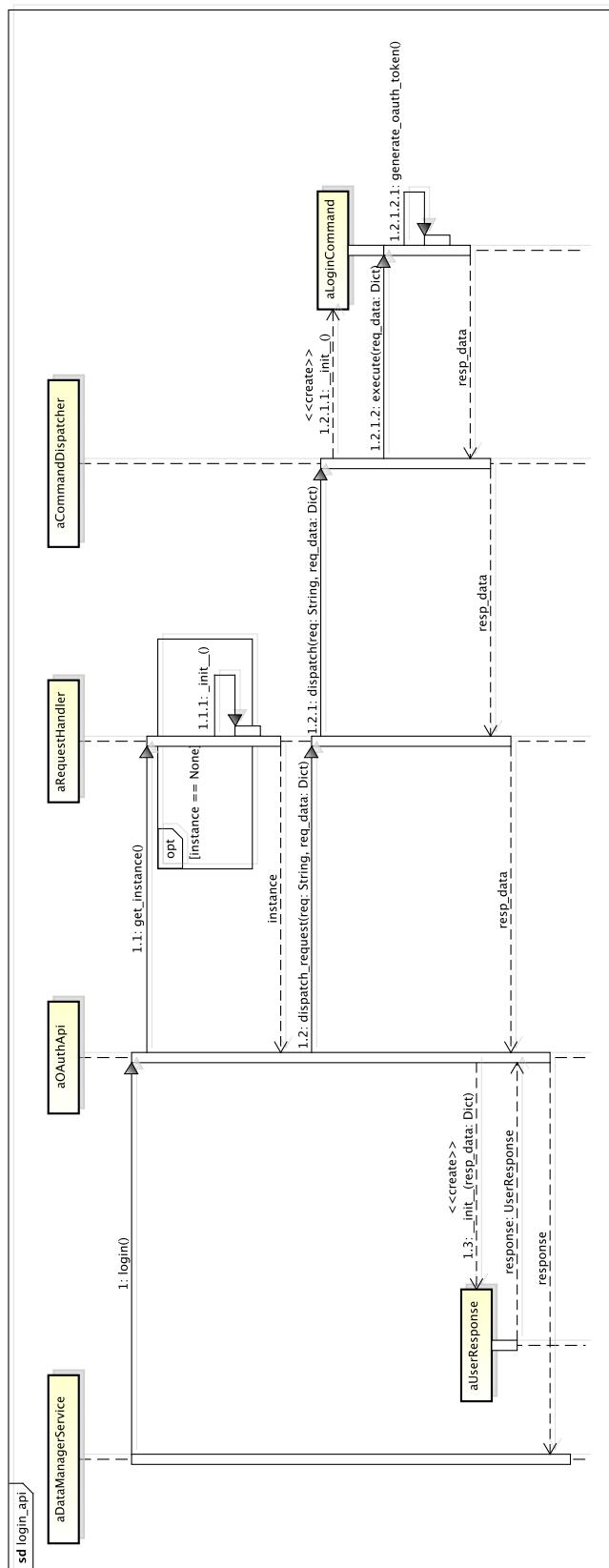


Figura 155: Diagramma di sequenza - Login API

- **OAuthApi**: classe che riceve i dati di autenticazione dal client attraverso il metodo `login()` della classe `DataManagerService`. Si occupa di creare un’istanza della classe `RequestHandler` dopodiché richiama il processore_G passandogli il tipo di richiesta e il dizionario per soddisfare tale richiesta;
- **RequestHandler**: classe che rappresenta un singleton che viene istanziato dal metodo `get_instance()` della classe `OAuthApi`. Attraverso il metodo `dispatch_request()` riceve il tipo di richiesta e un dizionario contenente i dati per soddisfare tale richiesta;
- **CommandDispatcher**: classe che riceve il tipo di richiesta e il dizionario per soddisfare tale richiesta dalla classe `RequestHandler`. Si occupa di reindirizzare tale richiesta al command che si occupa di effettuare l’autenticazione;
- **LoginCommand**: classe che si occupa di soddisfare la richiesta di login ricevuta dal metodo `execute()` della classe `CommandDispatcher` ritornando i dati dell’utente che ha effettuato l’autenticazione e un access token generato tramite l’utilizzo del metodo `generate_oauth_token()`;
- **UserResponse**: classe utilizzata dal metodo `login()` della classe `OAuthApi` per ritornare la risposta al client.

4.2.4 Recipe List API

Il diagramma alla figura 156 descrive il funzionamento del servizio REST offerto dal sistema e utile ad ottenere la lista di tutte le Recipe presenti nel sistema. Il client tramite il metodo `get_recipe_list()` della classe `DataManagerService` utilizza il servizio REST definito nella classe `MetricApi`.

- **MetricApi**: classe utilizzata dal client attraverso il metodo `get_recipe_list()` della classe `DataManagerService`. Si occupa di creare un’istanza della classe `RequestHandler` dopodiché richiama il processore_G passandogli il tipo di richiesta;
- **RequestHandler**: classe che rappresenta un singleton che viene istanziato dal metodo `get_instance()` della classe `MetricApi`. Attraverso il metodo `dispatch_request()` riceve il tipo di richiesta da soddisfare;
- **CommandDispatcher**: classe che riceve il tipo di richiesta da soddisfare dalla classe `RequestHandler`. Si occupa di reindirizzare tale richiesta al command che si occupa di prelevare tutte le Recipe_G dal sistema;
- **GetRecipeListCommand**: classe che si occupa di soddisfare la richiesta di prelievo di tutte le Recipe_G ricevuta dal metodo `execute()` della classe `CommandDispatcher` ritornando una lista contenente i dettagli di tutte le Recipe presenti nel sistema;
- **RecipeListResponse**: classe utilizzata dal metodo `get_recipe_list()` della classe `MetricApi` per ritornare la risposta al client.

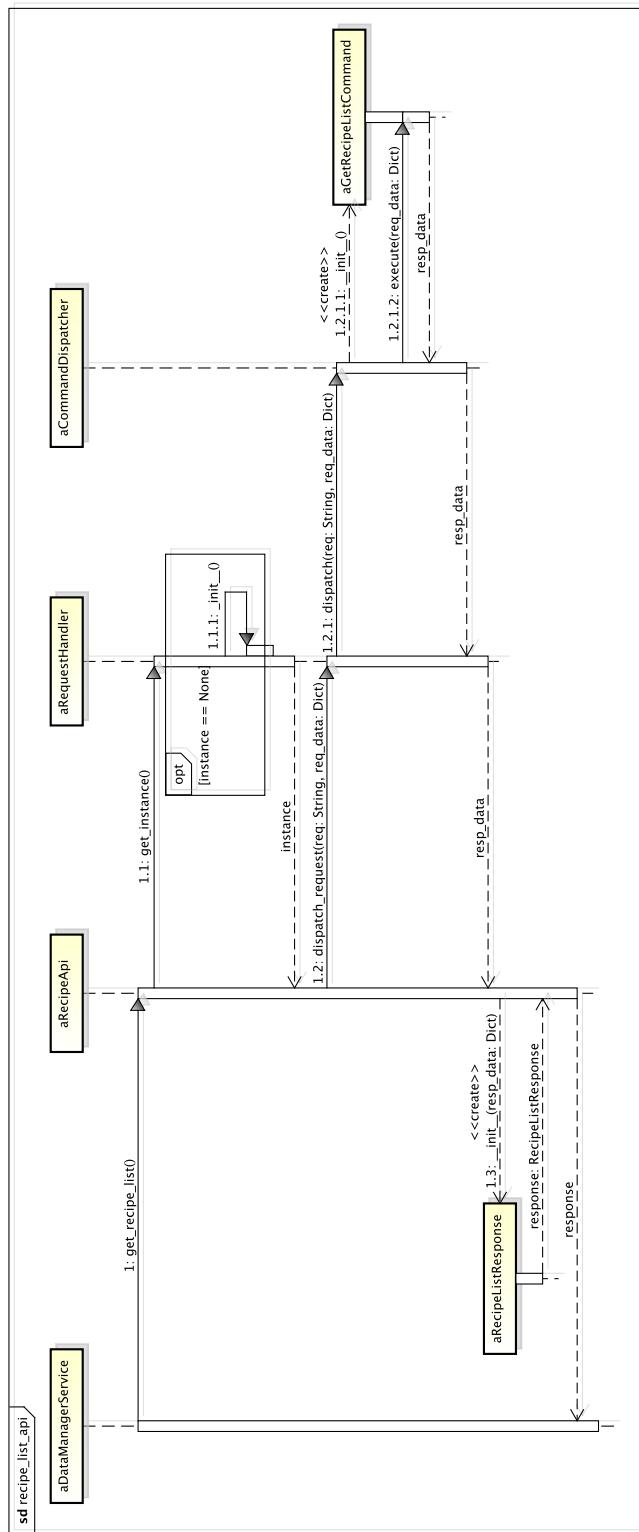


Figura 156: Diagramma di sequenza - Recipe List API

5 Tracciamento

5.1 Classi-Requisiti

Classi	Requisiti
client::model::data::ViewTypeModel	ROF5.3
client::model::data::ApiDocsModel	ROF11
client::model::services::AuthServiceTarget	ROF4.1.1.1 ROF4.1.2
client::model::services::RecipeService	ROF5.1 ROF5.1.1 ROF5.2 RDF6 RDF6.1 RDF6.2 RDF6.2.2 RDF6.3 RDF6.3.1 RFF7.1.1.4 RFF7.1.1.5 RFF7.1.1.6 RFF7.1.1.7 RFF8.4
client::model::services::RecipeAdminService	RFF10 RFF10.2
client::model::services::UserService	ROF3.3.1 ROF3.3.7.2.1
client::model::services::UserAdminService	RFF9.1.1 RFF9.2.1 ROF9.3.1 ROF9.3.2
client::model::services::ChartCreator	ROF5.3 ROF5.4 ROF5.5 ROF5.6.4.2

client::model::services::LineChartCreator	ROF5.3.1 ROF5.3.1.1 ROF5.4.1 ROF5.4.1.2 ROF5.4.1.4 ROF5.5.1 ROF5.5.1.2 ROF5.5.1.3 ROF5.5.1.4 ROF5.5.1.5 ROF5.5.1.7 ROF5.5.1.8 ROF5.6.4.1.2 ROF5.6.4.1.3 ROF5.6.4.1.4 ROF5.6.4.1.5 ROF5.6.4.1.6
client::model::services::BarChartCreator	ROF5.3.1 ROF5.3.1.6 ROF5.4.1 ROF5.4.1.1 ROF5.5.1 ROF5.5.1.1 ROF5.6.4.1.2 ROF5.6.4.1.3 ROF5.6.4.1.4 ROF5.6.4.1.5 ROF5.6.4.1.6
client::model::services::PieChartCreator	ROF5.3.1 ROF5.3.1.2 ROF5.3.1.3 ROF5.3.1.4 ROF5.4.1 ROF5.4.1.5 ROF5.4.1.7 ROF5.5.1 ROF5.5.1.10 ROF5.6.4.1.1 ROF5.6.4.1.2 ROF5.6.4.1.3 ROF5.6.4.1.4 ROF5.6.4.1.5 ROF5.6.4.1.6

client::model::services::MapChartCreator	ROF5.3.1 ROF5.3.1.5 ROF5.4.1 ROF5.4.1.3 ROF5.5.1 ROF5.5.1.9
client::model::services::RadarChartCreator	ROF5.4.1 ROF5.4.1.6 ROF5.5.1 ROF5.5.1.6
client::model::services::DataManagerService	ROF5.1 ROF5.2 RDF6.1 RDF6.2 RFF9.1.1 RFF9.2.1
client::view _G ::public::Login	ROF2 ROF2.1 ROF2.2 ROF2.3 ROF2.4 ROF2.4.1 ROF2.4.2 ROF3.3.7.2.2
client::view _G ::public::Register	ROF1 ROF1.1 ROF1.2 ROF1.3 ROF1.4 RDF1.5 ROF1.6 ROF1.7
client::view _G ::public::ApiDocs	ROF11 ROF12.1 ROF12.1.1 ROF12.1.2 ROF12.1.3 ROF12.2
client::view _G ::user::Home	ROF2.5 ROF4 ROF4.1.1.1 ROF4.1.2

client::view _G ::user::Recipe _G	ROF2.5 ROF5 ROF5.1 ROF5.1.1 ROF5.1.2 RFF5.1.3 RFF5.1.3.2 ROF8.3 ROF8.3.1
client::view _G ::user::Metrics	ROF5.2 ROF5.2.1 ROF5.2.1.2 ROF5.2.1.3
client::view _G ::user::Charts	ROF5.2.2 ROF5.3 ROF5.3.1.1 ROF5.3.1.2 ROF5.3.1.3 ROF5.3.1.4 ROF5.3.1.5 ROF5.3.1.6 ROF5.4 ROF5.4.1.1 ROF5.4.1.2 ROF5.4.1.3 ROF5.4.1.4 ROF5.4.1.5 ROF5.4.1.6 ROF5.4.1.7 ROF5.5 ROF5.5.1.1 ROF5.5.1.2 ROF5.5.1.3 ROF5.5.1.4 ROF5.5.1.5 ROF5.5.1.6 ROF5.5.1.7 ROF5.5.1.8 ROF5.5.1.9 ROF5.5.1.10 RDF6.2 RDF6.2.1 RDF6.2.2

client::view _G ::user::Compare	ROF5.6 ROF5.6.1 ROF5.6.2 ROF5.6.2.1 ROF5.6.2.2 ROF5.6.2.3 ROF5.6.3 ROF5.6.3.1 ROF5.6.4 ROF5.6.4.1 ROF5.6.4.1.1 ROF5.6.4.1.2 ROF5.6.4.1.3 ROF5.6.4.1.4 ROF5.6.4.1.5 ROF5.6.4.1.6 ROF5.6.4.2 ROF5.6.4.3
client::view _G ::user::RecipeRequest	RFF7 RFF7.1 RFF7.1.1 RFF7.1.1.1 RFF7.1.1.2 RFF7.1.1.3 RFF7.1.1.4 RFF7.1.1.5 RFF7.1.1.6 RFF7.1.1.7 RFF7.1.1.8 RFF7.1.1.9 RFF7.1.1.10 RFF7.1.1.11 RFF7.1.1.12 RFF7.1.2 RFF7.1.3
client::view _G ::user::Favourites	RDF6.1 RDF6.1.1 RDF6.3 RDF6.3.1

client::view _G ::user::TokenConfig	ROF11 ROF11.1.1 ROF11.1.2 ROF11.2.2 ROF11.2.3 ROF11.3.1 ROF11.3.2 ROF12 ROF12.1 ROF12.1.1 ROF12.1.2 ROF12.1.3 ROF12.2
client::view _G ::user::Settings	ROF3.1 ROF3.1.1 ROF3.1.2 ROF3.1.3 RFF3.1.4 ROF3.3 ROF3.3.1 ROF3.3.2 ROF3.3.2.1 ROF3.3.2.2 ROF3.3.3 ROF3.3.3.1 ROF3.3.3.2 ROF3.3.3.3 ROF3.3.4 ROF3.3.4.1 ROF3.3.4.2 ROF3.3.4.2.1 ROF3.3.4.3 ROF3.3.4.3.1 RDF3.3.5 ROF3.3.6 ROF3.3.6.2 ROF3.3.6.3 ROF3.3.7 ROF3.3.7.1 ROF3.3.7.2
client::view _G ::user::MenuMain	ROF3 ROF4 ROF4.1.1 ROF12
client::view _G ::user::MenuSecond	ROF3
client::view _G ::admin::RecipeConfig	ROF8.1

client::view _G ::admin::RequestList	RFF10 RFF10.1 RFF10.1.1 RFF10.1.2 RFF10.2 RFF10.3 RFF10.3.1 RFF10.3.2 RFF10.3.2.1 RFF10.4 RFF10.5 RFF10.5.1
client::view _G ::admin::Insert	ROF8.1 ROF8.1.1 ROF8.1.1.1 ROF8.1.1.2 ROF8.1.1.3 ROF8.1.1.4 ROF8.2 ROF8.2.1
client::view _G ::admin::Ratings	RFF8.4 RFF8.4.1 RFF8.4.2
client::view _G ::admin::MenuSecondAdmin	ROF8 ROF9
client::controller::public::PublicRoute	ROF1 ROF2 ROF3.3.7.2.2
client::controller::public::LoginCtrl	ROF2 ROF2.1 ROF2.2 ROF2.3 ROF2.4 ROF2.4.1 ROF2.4.2 ROF3.3.7.2.2
client::controller::public::RegisterCtrl	ROF1 ROF1.1 ROF1.2 ROF1.3 ROF1.4 ROF1.5 ROF1.6 ROF1.7

client::controller::user::MenuMainCtrl	ROF4 ROF4.1
client::controller::user::MenuSecondCtrl	ROF8
client::controller::user::HomeRoute	ROF2.5
client::controller::user::HomeCtrl	ROF4 ROF4.1
client::controller::user::RecipeRoute	ROF2.5
client::controller::user::RecipeCtrl	ROF5 ROF5.1 ROF5.1.1 RFF5.1.3.1 RFF5.1.3.2 ROF8.3
client::controller::user::MetricsCtrl	ROF5.2 ROF5.2.1.2
client::controller::user::ChartsCtrl	ROF5.2.2 ROF5.3.1.1 ROF5.3.1.2 ROF5.3.1.3 ROF5.3.1.4 ROF5.3.1.5 ROF5.3.1.6 ROF5.4 ROF5.4.1.1 ROF5.4.1.2 ROF5.4.1.3 ROF5.4.1.4 ROF5.4.1.5 ROF5.4.1.6 ROF5.4.1.7 ROF5.5 ROF5.5.1.1 ROF5.5.1.2 ROF5.5.1.3 ROF5.5.1.4 ROF5.5.1.5 ROF5.5.1.6 ROF5.5.1.7 ROF5.5.1.8 ROF5.5.1.9 ROF5.5.1.10 RDF6.2

client::controller::user::CompareCtrl	ROF5.6 ROF5.6.4 ROF5.6.4.1 ROF5.6.4.1.1 ROF5.6.4.1.2 ROF5.6.4.3
client::controller::user::RecipeRequestRoute	RFF7
client::controller::user::RecipeRequestCtrl	RFF7 RFF7.1 RFF7.1.1 RFF7.1.1.1 RFF7.1.1.2 RFF7.1.1.3 RFF7.1.1.4 RFF7.1.1.5 RFF7.1.1.6 RFF7.1.1.7 RFF7.1.1.8 RFF7.1.1.9 RFF7.1.1.10 RFF7.1.2 RFF7.1.3
client::controller::user::FavouritesRoute	RDF6
client::controller::user::FavouritesCtrl	RDF6.1 RDF6.1.1 RDF6.3 RDF6.3.1
client::controller::user::TokenConfigRoute	ROF12
client::controller::user::TokenConfigCtrl	ROF11.1 ROF11.1.1 ROF11.1.2 ROF11.2 ROF11.3.1 ROF11.3.2
client::controller::user::SettingsRoute	ROF3.1 ROF3.1.1 ROF3.1.2 ROF3.1.3 RFF3.1.4

client::controller::user::SettingsCtrl	ROF3.1 ROF3.1.1 ROF3.1.2 ROF3.1.3 RFF3.1.4 ROF3.3 ROF3.3.1 ROF3.3.2 ROF3.3.2.1 ROF3.3.2.2 ROF3.3.3 ROF3.3.3.1 ROF3.3.3.2 ROF3.3.3.3 ROF3.3.4 ROF3.3.4.1 ROF3.3.4.2 ROF3.3.4.2.1 ROF3.3.4.3 ROF3.3.4.3.1 RDF3.3.5 ROF3.3.6 ROF3.3.6.2 ROF3.3.6.3 ROF3.3.7 ROF3.3.7.1 ROF3.3.7.2
client::controller::admin::MenuSecondAdminCtrl	ROF8
client::controller::admin::RecipeConfigRoute	ROF8 ROF8.1
client::controller::admin::RecipeConfigCtrl	ROF8.1
client::controller::admin::RequestListRoute	ROF8 RFF10
client::controller::admin::RequestListCtrl	RFF10 RFF10.1 RFF10.1.1 RFF10.1.2 RFF10.2 RFF10.3 RFF10.3.1 RFF10.3.2 RFF10.3.2.1 RFF10.4 RFF10.5 RFF10.5.1

client::controller::admin::InsertRecipeCtrl	ROF8.1 ROF8.1.1 ROF8.1.1.1 ROF8.1.1.2 ROF8.1.1.3 ROF8.1.1.4 ROF8.2 ROF8.2.1
client::controller::admin::RatingsRoute	ROF8 RFF8.4
client::controller::admin::RatingsCtrl	RFF8.4 RFF8.4.2
client::controller::admin::UserConfigRoute	ROF8
client::controller::admin::UserConfigCtrl	ROF9.1 RFF9.1.1 ROF9.2 RFF9.2.1 ROF9.3 ROF9.3.1 ROF9.3.2
server::db::raw_data::AbsRawData	ROF5.2.1.1 ROF5.2.1.2
server::db::app_data::user::UserModel	ROF2.3 ROF3.1.1 ROF3.1.2 ROF3.1.3 RFF3.1.4 ROF3.3.6.1 ROF3.3.7.2.1 ROF4.1.1.1 ROF11.2.1
server::db::app_data::user::FavouritesModel	RDF6.4
server::db::app_data::recipe_G::RecipeRequestModel	RFF7 RFF10.3.1
server::db::app_data::recipe_G::RecipeModel	ROF5.1 ROF5.1.1 ROF8.2 ROF8.2.1.1 ROF8.2.1.2 RFF10.4 RFF10.5 ROF11.3.1.1

server::db::app_data::recipe _G ::RatingModel	RFF5.1.3.1 RFF8.4.1.1 RFF8.4.1.2
server::db::app_data::recipe _G ::AbsSocialMetricModel	ROF11.3.1.2
server::endpoints::api _G ::public::RecipeApi	ROF5.1 ROF5.1.1 ROF5.6.3 ROF5.6.4.1 ROF8.1.1.3 ROF8.1.1.4 ROF11.3.1.1 ROF11.3.1.2 ROF11.3.2
server::endpoints::api _G ::public::FbMetricsApi	ROF5.2.1.2 ROF5.2.2 ROF5.6.2 ROF5.6.4 ROF5.6.4.3 ROF11.3 ROF11.3.1
server::endpoints::api _G ::public::TwMetricsApi	ROF5.2.1.2 ROF5.2.2 ROF5.6.2 ROF5.6.4 ROF5.6.4.1.3 ROF5.6.4.1.4 ROF5.6.4.3 ROF11.3 ROF11.3.1
server::endpoints::api _G ::public::IgMetricsApi	ROF5.2.1.2 ROF5.2.2 ROF5.6.2 ROF5.6.4 ROF5.6.4.1.5 ROF5.6.4.1.6 ROF5.6.4.3 ROF11.3 ROF11.3.1
server::endpoints::api _G ::private::RecipePrivateApi	RFF5.1.3.1
server::endpoints::api _G ::private::UserApi	ROF3.1.1 ROF3.1.2 ROF3.1.3 RFF3.1.4 ROF3.3.6.1

server::endpoints::api _G ::private::RecipeRequestApi	RFF7 RFF7.2 RFF10
server::endpoints::api _G ::private::OAuthApi	ROF1.6 ROF2.3 ROF3.3.7.2.1 ROF4.1.1.1 ROF11.1 ROF11.1.1 ROF11.1.2 ROF11.2.1 ROF11.2.2 ROF11.2.3
server::endpoints::resp::public::RecipeResponse	ROF8.2.2 ROF11
server::endpoints::resp::public::RecipeListResponse	ROF11
server::endpoints::resp::private::RecipeReqResponse	ROF11
server::endpoints::resp::private::RecipeReqListResponse	ROF11
server::endpoints::resp::private::UserResponse	ROF11
server::endpoints::resp::private::UserListResponse	ROF11
server::processor _G ::commands::user::LoginCommand	ROF11.1.1 ROF11.1.2
server::processor _G ::commands::user::LogoutCommand	ROF11.2.1 ROF11.2.2 ROF11.2.3
server::processor _G ::commands::user::AddUserCommand	RDF1.8
server::processor _G ::commands::user::DeleteUserCommand	ROF3.3.7.2.1 ROF9.2.2
server::processor _G ::commands::user::EditUserCommand	ROF3.1.1 ROF3.1.2 ROF3.1.3 RFF3.1.4 ROF3.3.2.1 ROF3.3.2.2 ROF3.3.3 ROF3.3.3.1 ROF3.3.3.2 ROF3.3.3.3 ROF3.3.6.1
server::processor _G ::commands::user::EditPermissionCommand	ROF9.1.2

server::processor _G ::commands::recipe _G ::GetRecipeCommand	ROF11.3 ROF11.3.1 ROF11.3.1.2 ROF11.3.2
server::processor _G ::commands::recipe _G ::GetRecipeListCommand	ROF5.1 ROF5.1.1 ROF8.2.2
server::processor _G ::commands::recipe _G ::AddRecipeCommand	ROF8.2 ROF8.2.2 RFF10.3.1 RFF10.3.2 RFF10.3.2.1 RFF10.4 RFF10.5
server::processor _G ::commands::recipe _G ::DeleteRecipeCommand	ROF8.3.2 ROF8.3.3
server::processor _G ::commands::recipe _G ::RateRecipeCommand	RFF5.1.3.1
server::processor _G ::commands::requests::AddRequestCommand	RFF7
server::processor _G ::commands::requests::GetRequestListCommand	RFF10 RFF10.1.1
server::processor _G ::commands::social::AbsSocialCommand	ROF5.2.1.2 ROF5.2.2 ROF11.3.1.2
server::processor _G ::commands::social::AbsFbCommand	ROF5.3 ROF5.6.3
server::processor _G ::commands::social::AbsTwCommand	ROF5.4 ROF5.6.3
server::processor _G ::commands::social::AbsIgCommand	ROF5.5

5.2 Requisiti-Componenti

Requisito	Classi
ROF1	client::controller::public::PublicRoute client::controller::public::RegisterCtrl client::view _G ::public::Register
ROF1.1	client::controller::public::RegisterCtrl client::view _G ::public::Register
ROF1.1.1	server::processor _G ::controller::AddUserCommand
ROF1.1.2	server::processor _G ::controller::AddUserCommand
ROF1.1.3	server::processor _G ::controller::AddUserCommand
ROF1.2	client::view _G ::public::Register client::controller::public::RegisterCtrl
ROF1.3	client::view _G ::public::Register client::controller::public::RegisterCtrl
ROF1.3.1	server::processor _G ::controller::AddUserCommand
ROF1.3.2	server::processor _G ::controller::AddUserCommand
ROF1.3.3	server::processor _G ::controller::AddUserCommand
ROF1.4	client::view _G ::public::Register client::controller::public::RegisterCtrl
ROF1.4.1	server::processor _G ::controller::AddUserCommand
RDF1.5	client::view _G ::public::Register client::controller::public::RegisterCtrl
ROF1.6	client::view _G ::public::Register client::controller::public::RegisterCtrl server::processor _G ::controller::AddUserCommand server::endpoints::api _G ::private::OAuthApi
ROF1.7	client::view _G ::public::Register client::controller::public::RegisterCtrl
RDF1.8	server::processor _G ::commands::user::AddUserCommand
ROF2	client::controller::public::PublicRoute client::controller::public::LoginCtrl client::view _G ::public::Login
ROF2.1	client::controller::public::LoginCtrl client::view _G ::public::Login
ROF2.2	client::controller::public::LoginCtrl client::view _G ::public::Login

ROF2.3	client::controller::public::LoginCtrl client::view _G ::public::Login server::db::app_data::user::UserModel server::endpoints::api _G ::private::OAuthApi server::processor _G ::controller::LoginCommand
ROF2.4	client::controller::public::LoginCtrl client::view _G ::public::Login
ROF2.4.1	client::controller::public::LoginCtrl client::view _G ::public::Login
ROF2.4.2	client::controller::public::LoginCtrl client::view _G ::public::Login
ROF2.5	client::controller::user::HomeRoute client::controller::user::RecipeRoute client::view _G ::user::Home client::view::user::Recipe _G
ROF3	client::view _G ::user::MenuMain client::view::user::MenuSecond
ROF3.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl client::controller::user::SettingsRoute
ROF3.1.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl client::controller::user::SettingsRoute server::db::app_data::user::UserModel server::endpoints::api _G ::private::UserApi server::processor _G ::commands::user::EditUserCommand
ROF3.1.2	client::view _G ::user::Settings client::controller::user::SettingsCtrl client::controller::user::SettingsRoute server::db::app_data::user::UserModel server::endpoints::api _G ::private::UserApi server::processor _G ::commands::user::EditUserCommand
ROF3.1.3	client::view _G ::user::Settings client::controller::user::SettingsCtrl client::controller::user::SettingsRoute server::db::app_data::user::UserModel server::endpoints::api _G ::private::UserApi server::processor _G ::commands::user::EditUserCommand

RFF3.1.4	client::view _G ::user::Settings client::controller::user::SettingsCtrl client::controller::user::SettingsRoute server::db::app_data::user::UserModel server::endpoints::api _G ::private::UserApi server::processor _G ::commands::user::EditUserCommand
ROF3.3	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl client::model::services::UserService
ROF3.3.2	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.2.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl server::processor _G ::commands::user::EditUserCommand
ROF3.3.2.2	client::view _G ::user::Settings client::controller::user::SettingsCtrl server::processor _G ::commands::user::EditUserCommand
ROF3.3.3	client::view _G ::user::Settings client::controller::user::SettingsCtrl server::processor _G ::commands::user::EditUserCommand
ROF3.3.3.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl server::processor _G ::commands::user::EditUserCommand
ROF3.3.3.2	client::view _G ::user::Settings client::controller::user::SettingsCtrl server::processor _G ::commands::user::EditUserCommand
ROF3.3.3.3	client::view _G ::user::Settings client::controller::user::SettingsCtrl server::processor _G ::commands::user::EditUserCommand
ROF3.3.4	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.4.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.4.2	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.4.2.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.4.3	client::view _G ::user::Settings client::controller::user::SettingsCtrl

ROF3.3.4.3.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl
RDF3.3.5	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.6	client::view _G ::user::Settings client::controller::user::SettingsCtrl client::model::services::UserServices client::model::services::DataManagerServices
ROF3.3.6.1	server::db::app_data::user::UserModel server::endpoints::api _G ::private::UserApi server::processor _G ::commands::user::EditUserCommand
ROF3.3.6.2	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.6.3	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.7	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.7.1	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.7.2	client::view _G ::user::Settings client::controller::user::SettingsCtrl
ROF3.3.7.2.1	client::model::services::UserService client::model::services::DataManagerServices server::db::app_data::user::UserModel server::endpoints::api _G ::private::OAuthApi server::processor _G ::commands::user::DeleteUserCommand
ROF3.3.7.2.2	client::controller::public::PublicRoute client::controller::public::LoginCtrl client::view _G ::public::Login
ROF4	client::view _G ::user::Home client::view::user::MenuMain client::controller::user::HomeCtrl client::controller::user::MenuMainCtrl
ROF4.1	client::controller::user::HomeCtrl client::controller::user::MenuMainCtrl
ROF4.1.1	client::view _G ::user::MenuMain client::model::services::AuthServiceTargetAdapter

ROF4.1.1.1	server::db::app_data::user::UserModel server::endpoints::api _G ::private::OAuthApi client::model::services::AuthServiceTarget client::view _G ::public::Login client::view::user::Home
ROF4.1.2	client::model::services::AuthServiceTarget client::view _G ::public::Login client::view::user::Home
ROF5	client::view _G ::user::Recipe _G client::controller::user::RecipeCtrl
ROF5.1	server::db::app_data::recipe _G ::RecipeModel server::endpoints::api _G ::public::RecipeApi server::processor _G ::commands::recipe::GetRecipeListCommand client::model::services::RecipeService client::model::services::DataManagerService client::view _G ::user::Recipe _G client::controller::user::RecipeCtrl
ROF5.1.1	server::db::app_data::recipe _G ::RecipeModel server::endpoints::api _G ::public::RecipeApi server::processor _G ::commands::recipe::GetRecipeListCommand client::model::services::RecipeService client::view _G ::user::Recipe _G client::controller::user::RecipeCtrl
ROF5.1.2	client::view _G ::user::Recipe _G
RFF5.1.3	client::view _G ::user::Recipe _G
RFF5.1.3.1	server::db::app_data::recipe _G ::RatingModel server::endpoints::api _G ::private::RecipePrivateApi server::processor _G ::commands::recipe::RateRecipeCommand client::controller::user::RecipeCtrl
RFF5.1.3.2	client::view _G ::user::Recipe _G client::controller::user::RecipeCtrl
ROF5.2	client::model::services::RecipeService client::model::services::DataManagerService client::view _G ::user::Metrics client::controller::user::MetricsCtrl
ROF5.2.1	client::view _G ::user::Metrics
ROF5.2.1.1	server::db::raw_data::AbsRawData

ROF5.2.1.2	server::db::raw_data::AbsRawData server::endpoints::api _G ::public::TwMetricsApi server::endpoints::api::public::FbMetricsApi server::endpoints::api::public::IgMetricsApi server::processor _G ::commands::social::AbsSocialCommand client::view _G ::user::Metrics client::controller::user::MetricsCtrl
ROF5.2.1.3	client::view _G ::user::Metrics
ROF5.2.2	server::endpoints::api _G ::public::TwMetricsApi server::endpoints::api::public::FbMetricsApi server::endpoints::api::public::IgMetricsApi server::processor _G ::commands::social::AbsSocialCommand client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.3	client::model::data::ViewTypeModel server::processor _G ::commands::social::AbsFbCommand client::model::services::ChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.3.1	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::MapChartCreator client::model::services::PieChartCreator
ROF5.3.1.1	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.3.1.2	client::model::services::PieChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.3.1.3	client::model::services::PieChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.3.1.4	client::model::services::PieChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.3.1.5	client::model::services::MapChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.3.1.6	client::model::services::BarChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl

ROF5.4	server::processor _G ::commands::social::AbsTwCommand client::model::services::ChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.4.1	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::MapChartCreator client::model::services::PieChartCreator client::model::services::RadarChartCreator
ROF5.4.1.1	client::model::services::BarChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.4.1.2	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.4.1.3	client::model::services::MapChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.4.1.4	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.4.1.5	client::model::services::PieChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.4.1.6	client::model::services::RadarChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.4.1.7	client::model::services::PieChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5	server::processor _G ::commands::social::AbsIgCommand client::model::services::ChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::MapChartCreator client::model::services::PieChartCreator client::model::services::RadarChartCreator
ROF5.5.1.1	client::model::services::BarChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl

ROF5.5.1.2	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.3	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.4	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.5	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.6	client::model::services::RadarChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.7	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.8	client::model::services::LineChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.9	client::model::services::MapChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.5.1.10	client::model::services::PieChartCreator client::view _G ::user::Charts client::controller::user::ChartsCtrl
ROF5.6	client::view _G ::user::Compare client::controller::user::CompareCtrl
ROF5.6.1	client::view _G ::user::Compare
ROF5.6.2	server::endpoints::api _G ::public::TwMetricsApi server::endpoints::api::public::FbMetricsApi server::endpoints::api::public::IgMetricsApi client::view _G ::user::Compare
ROF5.6.2.1	client::view _G ::user::Compare client::model::service::RecipeService server::db::raw_data::fb::raw_FbPage server::db::raw_data::fb::raw_FbEvent
ROF5.6.2.2	client::view _G ::user::Compare client::model::service::RecipeService server::db::raw_data::tw::raw_TwUser server::db::raw_data::tw::raw_TwHashtag

ROF5.6.2.3	client::view _G ::user::Compare client::model::service::RecipeService server::db::raw_data::ig::raw_IgUser server::db::raw_data::ig::raw_IgHashtag
ROF5.6.3	client::view _G ::user::Compare client::model::service::RecipeService client::model::service::DataManagerService server::endpoints::api _G ::public::RecipeApi server::processor _G ::commands::social::AbsFbCommand server::processor::commands::social::AbsTwCommand server::processor::commands::social::AbsIgCommand
ROF5.6.3.1	client::view _G ::user::Compare
ROF5.6.4	client::controller::user::CompareCtrl client::view _G ::user::Compare server::endpoints::api _G ::public::FbMetricsApi server::endpoints::api::public::TwMetricsApi server::endpoints::api::public::IgMetricsApi
ROF5.6.4.1	client::view _G ::user::Compare client::controller::user::CompareCtrl client::model::service::ChartCreator client::model::service::DataManagerService server::endpoints::api _G ::public::RecipeApi
ROF5.6.4.1.1	client::model::services::PieChartCreator client::view _G ::user::Compare client::controller::user::CompareCtrl
ROF5.6.4.1.2	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::PieChartCreator client::view _G ::user::Compare client::controller::user::CompareCtrl
ROF5.6.4.1.3	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::PieChartCreator client::view _G ::user::Compare server::endpoints::api _G ::public::TwMetricsApi
ROF5.6.4.1.4	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::PieChartCreator client::view _G ::user::Compare server::endpoints::api _G ::public::TwMetricsApi

ROF5.6.4.1.5	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::PieChartCreator client::view _G ::user::Compare server::endpoints::api _G ::public::IgMetricsApi
ROF5.6.4.1.6	client::model::services::LineChartCreator client::model::services::BarChartCreator client::model::services::PieChartCreator client::view _G ::user::Compare server::endpoints::api _G ::public::IgMetricsApi
ROF5.6.4.2	client::model::services::ChartCreator client::view _G ::user::Compare
ROF5.6.4.3	client::controller::user::CompareCtrl client::view _G ::user::Compare server::endpoints::api _G ::public::FbMetricsApi server::endpoints::api::public::TwMetricsApi server::endpoints::api::public::IgMetricsApi
RDF6	client::controller::user::FavouritesRoute client::model::services::RecipeService server::endpoints::api _G ::private::FavouritesApi server::processor _G ::commands::user::AddFavouritesCommand server::processor::commands::user::DeleteFavouritesCommand
RDF6.1	client::model::services::RecipeService client::model::services::DataManagerService client::view _G ::user::Favourites client::controller::user::FavouritesCtrl
RDF6.1.1	client::controller::user::FavouritesCtrl client::view _G ::user::Favourites
RDF6.2	client::model::services::RecipeService client::model::services::DataManagerService client::view _G ::user::Charts client::controller::user::ChartsCtrl
RDF6.2.1	client::view _G ::user::Charts
RDF6.2.2	client::model::services::RecipeService client::view _G ::user::Charts client::view::controller::ChartsCtrl server::endpoints::api _G ::private::FavouritesApi
RDF6.3	client::model::services::RecipeService client::view _G ::user::Favourites client::controller::user::FavouritesCtrl server::processor _G ::commands::user::DeleteFavoriteCommand

RDF6.3.1	client::model::services::RecipeService client::view _G ::user::Favourites client::controller::user::FavouritesCtrl
RDF6.4	server::db::app_data::user::FavouritesModel
RFF7	client::model::service::RecipeService client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestRoute client::controller::user::RecipeRequestCtrl server::db::app_data::recipe _G ::RecipeRequestModel server::endpoints::api _G ::private::RecipeRequestApi server::processor _G ::commands::requests::AddRequestCommand
RFF7.1	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1.1	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1.2	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1.3	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1.4	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl client::model::services::RecipeService
RFF7.1.1.5	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl client::model::services::RecipeService
RFF7.1.1.6	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl client::model::services::RecipeService
RFF7.1.1.7	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl client::model::services::RecipeService
RFF7.1.1.8	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1.9	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1.10	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.1.11	client::view _G ::user::RecipeRequest

RFF7.1.1.12	client::view _G ::user::RecipeRequest
RFF7.1.2	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.1.3	client::view _G ::user::RecipeRequest client::controller::user::RecipeRequestCtrl
RFF7.2	server::endpoints::api _G ::private::RecipeRequestApi
ROF8	client::controller::user::MenuSecondCtrl client::controller::admin::RatingsRoute client::controller::admin::UserConfigRoute client::controller::admin::RequestListRoute client::controller::admin::RecipeConfigRoute client::controller::admin::MenuSecondAdminCtrl client::view _G ::admin::MenuSecondAdmin server::db::app_data::user::AdminModel server::endpoints::api _G ::private::UserApi server::processor _G ::commands::recipe _G ::AddRecipeCommand server::processor::commands::recipe::DeleteRecipeCommand
ROF8.1	client::controller::admin::RecipeConfigRoute client::view _G ::admin::RecipeConfig client::view::admin::Insert client::controller::admin::InsertRecipeCtrl client::controller::admin::RecipeConfigCtrl
ROF8.1.1	client::controller::admin::InsertRecipeCtrl client::view _G ::admin::Insert
ROF8.1.1.1	client::controller::admin::InsertRecipeCtrl client::view _G ::admin::Insert
ROF8.1.1.2	client::controller::admin::InsertRecipeCtrl client::view _G ::admin::Insert
ROF8.1.1.3	client::controller::admin::InsertRecipeCtrl client::view _G ::admin::Insert server::endpoints::api _G ::public::RecipeApi
ROF8.1.1.4	client::controller::admin::InsertRecipeCtrl client::view _G ::admin::Insert server::endpoints::api _G ::public::RecipeApi
ROF8.2	client::controller::admin::InsertRecipeCtrl client::view _G ::admin::Insert server::db::app_data::recipe _G ::RecipeModel server::endpoints::api _G ::private::RecipeApi server::processor _G ::commands::recipe::AddRecipeCommand
ROF8.2.1	client::controller::admin::InsertRecipeCtrl client::view _G ::admin::Insert
ROF8.2.1.1	server::db::app_data::recipe _G ::RecipeModel

ROF8.2.1.2	server::db::app_data::recipe _G ::RecipeModel
ROF8.2.2	client::model::service::RecipeAdminService client::model::service::DataManagerService server::endpoints::resp::public::RecipeResponse server::processor _G ::commands::recipe _G ::AddRecipeCommand server::processor::commands::recipe::GetRecipeListCommand
ROF8.3	client::view _G ::user::Recipe _G client::controller::user::RecipeCtrl
ROF8.3.1	client::view _G ::user::Recipe _G
ROF8.3.2	server::processor _G ::commands::recipe _G ::DeleteRecipeCommand
ROF8.3.3	server::processor _G ::commands::recipe _G ::DeleteRecipeCommand
RFF8.4	client::model::services::RecipeService client::view _G ::admin::Ratings client::controller::admin::RatingsCtrl client::controller::admin::RatingsRoute
RFF8.4.1	client::view _G ::admin::Ratings
RFF8.4.1.1	server::db::app_data::recipe _G ::RatingModel
RFF8.4.1.2	server::db::app_data::recipe _G ::RatingModel
RFF8.4.2	client::view _G ::admin::Ratings client::controller::admin::RatingsCtrl
ROF9	client::view _G ::admin::MenuSecondAdmin
ROF9.1	client::view _G ::admin::UserConfig client::controller::admin::UserConfigCtrl
RFF9.1.1	client::model::services::DataManagerService client::controller::admin::UserConfigCtrl client::model::services::UserAdminService
ROF9.1.2	server::processor _G ::commands::user::EditPermissionCommand
ROF9.2	client::view _G ::admin::UserConfig client::controller::admin::UserConfigCtrl
RFF9.2.1	client::model::services::DataManagerService client::controller::admin::UserConfigCtrl client::model::services::UserAdminService
ROF9.2.2	server::processor _G ::commands::user::DeleteUserCommand
ROF9.3	client::view _G ::admin::UserConfig client::controller::admin::UserConfigCtrl
ROF9.3.1	client::view _G ::admin::UserConfig client::controller::admin::UserConfigCtrl client::model::services::UserAdminService

ROF9.3.2	client::view _G ::admin::UserConfig client::controller::admin::UserConfigCtrl client::model::services::UserAdminService
RFF10	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl client::controller::admin::RequestListRoute client::model::services::RecipeAdminService server::endpoints::api _G ::private::RecipeRequestApi server::processor _G ::commands::requests::GetRequestListCommand
RFF10.1	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl
RFF10.1.1	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl server::processor _G ::commands::requests::GetRequestListCommand
RFF10.1.2	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl
RFF10.2	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl client::model::services::RecipeAdminService
RFF10.3	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl
RFF10.3.1	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl server::db::app_data::recipe _G ::RecipeRequestModel server::processor _G ::commands::recipe::AddRecipeCommand
RFF10.3.2	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl server::processor _G ::commands::recipe _G ::AddRecipeCommand
RFF10.3.2.1	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl server::processor _G ::commands::recipe _G ::AddRecipeCommand
RFF10.4	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl client::model::service::RecipeAdminService client::model::service::DataManagerService server::db::app_data::recipe _G ::RecipeModel server::processor _G ::commands::recipe::AddRecipeCommand
RFF10.5	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl client::model::service::RecipeAdminService client::model::service::DataManagerService server::db::app_data::recipe _G ::RecipeModel server::processor _G ::commands::recipe::AddRecipeCommand

RFF10.5.1	client::view _G ::admin::RequestList client::controller::admin::RequestListCtrl client::model::service::RecipeAdminService client::model::service::DataManagerService
RFF10.5.1.1	server::endpoints::api _G ::private::RecipeRequestModel server::processor _G ::commands::requests::AddRequestModel
ROF11	client::model::data::ApiDocsModel client::view _G ::public::ApiDocs client::view::user::TokenConfig server::endpoints::resp::private::RecipeReqListResponse server::endpoints::resp::private::FavouriteListResponse server::endpoints::resp::private::UserListResponse server::endpoints::resp::private::RecipeReqResponse server::endpoints::resp::private::FavouritesResponse server::endpoints::resp::private::UserResponse server::endpoints::resp::public::RecipeListResponse server::endpoints::resp::public::RecipeResponse server::endpoints::resp::public::MetricResponse
ROF11.1	client::controller::user::TokenConfigCtrl client::model::service::UserService server::endpoints::api _G ::private::OAuthApi
ROF11.1.1	client::controller::user::TokenConfigCtrl client::view _G ::user::TokenConfig server::endpoints::api _G ::private::OAuthApi server::processor _G ::commands::user::LoginCommand
ROF11.1.2	client::controller::user::TokenConfigCtrl client::view _G ::user::TokenConfig server::endpoints::api _G ::private::OAuthApi server::processor _G ::commands::user::LoginCommand
ROF11.2	client::view _G ::user::TokenConfig client::controller::user::TokenConfigCtrl
ROF11.2.1	server::db::app_data::user::UserModel server::endpoints::api _G ::private::OAuthApi server::processor _G ::commands::user::LogoutCommand
ROF11.2.2	client::view _G ::user::TokenConfig server::endpoints::api _G ::private::OAuthApi server::processor _G ::commands::user::LogoutCommand
ROF11.2.3	client::view _G ::user::TokenConfig server::endpoints::api _G ::private::OAuthApi server::processor _G ::commands::user::LogoutCommand

ROF11.3	server::endpoints::api _G ::public::FbMetricsApi server::endpoints::api::public::TwMetricsApi server::endpoints::api::public::IgMetricsApi server::processor _G ::commands::recipe _G ::GetRecipeCommand
ROF11.3.1	client::view _G ::user::TokenConfig client::controller::user::TokenConfigCtrl client::model::service::UserService client::model::service::DataManagerService server::endpoints::api _G ::public::FbMetricsApi server::endpoints::api::public::TwMetricsApi server::endpoints::api::public::IgMetricsApi server::processor _G ::commands::recipe _G ::GetRecipeCommand
ROF11.3.1.1	server::db::app_data::recipe _G ::RecipeModel server::endpoints::api _G ::public::RecipeApi server::processor _G ::commands::recipe::GetRecipeListCommand
ROF11.3.1.2	server::db::app_data::recipe _G ::AbsSocialMetricModel server::endpoints::api _G ::public::RecipeApi server::processor _G ::commands::social::AbsSocialCommand server::processor::commands::recipe::GetRecipeCommand
ROF11.3.1.3	server::db::raw_data::AbsFbRawData server::db::raw_data::AbsTwRawData server::db::raw_data::AbsIgRawData server::endpoints::api _G ::public::RespApi server::processor _G ::commands::social::AbsSocialCommand
ROF11.3.2	client::view _G ::user::TokenConfig client::controller::user::TokenConfigCtrl server::endpoints::api _G ::public::RecipeApi server::processor _G ::commands::recipe _G ::GetRecipeCommand
ROF12	client::controller::user::TokenConfigRoute client::view _G ::user::TokenConfig client::view::user::MenuMain
ROF12.1	client::view _G ::user::TokenConfig client::view::public::ApiDocs
ROF12.1.1	client::view _G ::user::TokenConfig client::view::public::ApiDocs
ROF12.1.2	client::view _G ::user::TokenConfig client::view::public::ApiDocs
ROF12.1.3	client::view _G ::user::TokenConfig client::view::public::ApiDocs
ROF12.2	client::view _G ::user::TokenConfig client::view::public::ApiDocs