

MashUp

Software Engineering Group

info@mashup-unipd.it

Informazioni Documento

Nome documento	<i>Specifica Tecnica</i>
Versione	<i>v1.0.0</i>
Data redazione	2015-02-20
Redattori	Carnovalini Filippo Ceccon Lorenzo Cusinato Giacomo Faccin Nicola Roetta Marco Tesser Paolo
Verificatori	Ceccon Lorenzo Roetta Marco
Approvazione	Santacatterina Luca
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 vuol definire l'architettura generale del prodotto che verrà sviluppato dal team *MashUp*.

Diario Revisioni

Modifica	Autore & Ruolo	Data	Versione
<i>Approvazione documento</i>	Santacatterina Luca <i>Responsabile di Progetto</i>	2015-03-30	v1.0.0
<i>Verifica documento</i>	Ceccon Lorenzo <i>Verificatore</i>	2015-03-26	v0.3.0
<i>Correzioni errori relativi alla sezione client::model::data</i>	Tesser Paolo <i>Progettista</i>	2015-03-23	v0.2.3
<i>Correzioni errori relativi alla sezione server::miner</i>	Faccin Nicola <i>Progettista</i>	2015-03-23	v0.2.2
<i>Correzioni errori relativi alla sezione server::db</i>	Cusinato Giacomo <i>Progettista</i>	2015-03-22	v0.2.1
<i>Verifica documento</i>	Roetta Marco <i>Verificatore</i>	2015-03-20	v0.2.0
<i>Verifica documento</i>	Ceccon Lorenzo <i>Verificatore</i>	2015-03-20	v0.1.0
<i>Stesura capitolo Interfaccia REST</i>	Cusinato Giacomo <i>Progettista</i>	2015-03-18	v0.0.22
<i>Stesura appendice Mockup</i>	Tesser Paolo <i>Progettista</i>	2015-03-18	v0.0.21
<i>Stesura appendice Design Pattern</i>	Roetta Marco <i>Progettista</i>	2015-03-16	v0.0.20
<i>Stesura sezione Design Pattern: Comportamentali e Creazionali</i>	Carnovalini Filippo <i>Progettista</i>	2015-03-14	v0.0.19
<i>Stesura sezione Design Pattern: Architettonicali e Strutturali</i>	Roetta Marco <i>Progettista</i>	2015-03-11	v0.0.18
<i>Stesura sezione server::processor nel capitolo Componenti e Classi</i>	Ceccon Lorenzo <i>Progettista</i>	2015-03-09	v0.0.17
<i>Stesura sezione server::miner nel capitolo Componenti e Classi</i>	Faccin Nicola <i>Progettista</i>	2015-03-05	v0.0.16
<i>Stesura sezione client::model nel capitolo Componenti e Classi</i>	Tesser Paolo <i>Progettista</i>	2015-03-03	v0.0.15
<i>Stesura sezione server::endpoints nel capitolo Componenti e Classi</i>	Cusinato Giacomo <i>Progettista</i>	2015-03-03	v0.0.14
<i>Stesura sezione server::db nel capitolo Componenti e Classi</i>	Ceccon Lorenzo <i>Progettista</i>	2015-03-02	v0.0.13
<i>Rimossa sezione sui diagrammi di attività perché più conformi al documento di Analisi</i>	Tesser Paolo <i>Progettista</i>	2015-03-02	v0.0.12
<i>Stesura sezione client::controller nel capitolo Componenti e Classi</i>	Roetta Marco <i>Progettista</i>	2015-02-27	v0.0.11
<i>Stesura sezione client::view nel capitolo Componenti e Classi</i>	Carnovalini Filippo <i>Progettista</i>	2015-02-27	v0.0.10
<i>Creata scheletro per sezione Client nel capitolo Componenti e Classi</i>	Carnovalini Filippo <i>Progettista</i>	2015-02-26	v0.0.9

<i>Creata scheletro per sezione Server nel capitolo Componenti e Classi</i>	Cusinato Giacomo <i>Progettista</i>	2015-02-26	v0.0.8
<i>Stesura sezione Descrizione architettura generale: dettaglio dei tre livelli di architettura e protocolli di comunicazione</i>	Roetta Marco <i>Progettista</i>	2015-02-25	v0.0.7
<i>Stesura sezione Descrizione architettura generale: Metodo e formalismo di specifica e Architettura Generale</i>	Roetta Marco <i>Progettista</i>	2015-02-24	v0.0.6
<i>Aggiunte le librerie per l'utilizzo delle API dei diversi social in maniera più semplice e astratta</i>	Tesser Paolo <i>Progettista</i>	2015-02-23	v0.0.5
<i>Stesura sezione sullo schema della base di dati</i>	Faccin Nicola <i>Progettista</i>	2015-02-23	v0.0.4
<i>Prima stesura sulle tecnologie utilizzate per lo sviluppo dell'applicativo</i>	Tesser Paolo <i>Progettista</i>	2015-02-21	v0.0.3
<i>Stesura introduzione</i>	Tesser Paolo <i>Progettista</i>	2015-02-21	v0.0.2
<i>Creazione struttura del documento</i>	Tesser Paolo <i>Progettista</i>	2015-02-20	v0.0.1

Indice

1 Introduzione	1
1.1 Scopo del documento	1
1.2 Scopo del prodotto	1
1.3 Glossario	1
1.4 Riferimenti	1
1.4.1 Normativi	1
1.4.2 Informativi	1
2 Tecnologie utilizzate	3
2.1 Linguaggi	3
2.1.1 CSS3	3
2.1.2 HTML5	3
2.1.3 JavaScript	3
2.1.4 JSON	4
2.1.5 Python	4
2.1.6 YAML	4
2.2 Framework	4
2.2.1 AngularJS	4
2.2.2 Twitter Bootstrap	5
2.3 Librerie	5
2.3.1 Chart.js	5
2.3.2 Google Chart	5
2.3.3 facebook-sdk	6
2.3.4 tweepy.api	6
2.3.5 python-instagram	6
2.3.6 ng-token-auth	6
2.4 Database	7
2.4.1 Google Cloud Datastore	7
3 Descrizione Architettura	8
3.1 Metodo e formalismo di specifica	8
3.2 Architettura generale	8
3.2.1 Architettura Client Tier	9
3.2.2 Architettura Server Tier	9
3.2.3 Architettura Database Tier	9
3.2.4 Protocollo di comunicazione Client-Server	10
3.2.5 Protocollo di comunicazione Server-Database	10
4 Componenti e Classi	11
4.1 Client (Front-end)	11
4.1.1 client	11
4.1.2 client::model	13
4.1.3 client::model::data	14
4.1.3.1 Classi	14
4.1.4 client::model::services	17
4.1.4.1 Classi	18
4.1.5 client::view	23
4.1.6 client::view::public	25
4.1.6.1 Classi	25

4.1.7	client::view::user	26
4.1.7.1	Classi	27
4.1.8	client::view::admin	30
4.1.8.1	Classi	30
4.1.9	client::controller	34
4.1.10	client::controller::public	35
4.1.10.1	Classi	35
4.1.11	client::controller::user	37
4.1.11.1	Classi	37
4.1.12	client::controller::admin	43
4.1.12.1	Classi	44
4.2	Server (Back-end)	48
4.2.1	server	48
4.2.2	server::db	50
4.2.3	server::db::raw_data	50
4.2.3.1	Classi	51
4.2.4	server::db::raw_data::fb	51
4.2.4.1	Classi	53
4.2.5	server::db::raw_data::tw	54
4.2.5.1	Classi	54
4.2.6	server::db::raw_data::ig	58
4.2.6.1	Classi	58
4.2.7	server::db::app_data	59
4.2.8	server::db::app_data::user	60
4.2.8.1	Classi	60
4.2.9	server::db::app_data::recipe	61
4.2.9.1	Classi	61
4.2.10	server::processor	64
4.2.10.1	Classi	64
4.2.11	server::processor::commands	65
4.2.11.1	Classi	66
4.2.12	server::processor::commands::user	66
4.2.12.1	Classi	67
4.2.13	server::processor::commands::recipe	69
4.2.13.1	Classi	69
4.2.14	server::processor::commands::requests	70
4.2.14.1	Classi	71
4.2.15	server::processor::commands::social	72
4.2.15.1	Classi	72
4.2.16	server::miner	78
4.2.16.1	Classi	78
4.2.17	server::miner::fb	79
4.2.17.1	Classi	79
4.2.18	server::miner::tw	83
4.2.18.1	Classi	83
4.2.19	server::miner::ig	85
4.2.19.1	Classi	85
4.2.20	server::endpoints	87
4.2.21	server::endpoints::api	87
4.2.22	server::endpoints::api::public	88

4.2.22.1	Classi	89
4.2.23	server::endpoints::api::private	90
4.2.23.1	Classi	90
4.2.24	server::endpoints::resp	91
4.2.25	server::endpoints::resp::public	93
4.2.25.1	Classi	93
4.2.26	server::endpoints::resp::public::fb	93
4.2.26.1	Classi	93
4.2.27	server::endpoints::resp::public::tw	96
4.2.27.1	Classi	96
4.2.28	server::endpoints::resp::public::ig	98
4.2.28.1	Classi	98
4.2.29	server::endpoints::resp::private	100
4.2.29.1	Classi	100
4.3	Database	103
5	Interfaccia REST	104
5.1	Lista servizi REST	104
5.1.1	API pubbliche	104
5.1.2	API private	106
6	Design Pattern	108
6.1	Design pattern architetturali	109
6.1.1	Three-Tier	109
6.1.2	MVW	109
6.1.2.1	MVC	109
6.1.2.2	MVVM	109
6.1.3	Dependency Injection	110
6.2	Design pattern creazionali	111
6.2.1	Constructor Pattern	111
6.2.2	Prototype Pattern	111
6.2.3	Module Pattern	111
6.2.4	Singleton	111
6.3	Design pattern strutturali	113
6.3.1	Adapter (object)	113
6.3.2	Façade	113
6.3.3	Front controller	114
6.4	Design pattern comportamentali	115
6.4.1	Page Controller	115
6.4.2	Template View	115
6.4.3	Command	116
6.4.4	Template Method	116
7	Stime di fattibilità e bisogno di risorse	118
8	Tracciamento	119
8.1	Componenti-Requisiti	119
8.2	Requisiti-Componenti	126

A Mockup	134
A.1 Login	134
A.1.1 Descrizione generale	134
A.1.2 Vista desktop	134
A.1.3 Vista mobile	135
A.2 Register	136
A.2.1 Descrizione generale	136
A.2.2 Vista desktop	136
A.2.3 Vista mobile	136
A.3 Recipe	137
A.3.1 Descrizione generale	137
A.3.2 Vista desktop	137
A.3.3 Vista mobile	138
A.4 Metrics	139
A.4.1 Descrizione generale	139
A.4.2 Vista desktop	139
A.4.3 Vista mobile	139
A.5 Settings	140
A.5.1 Descrizione generale	140
A.5.2 Vista desktop	140
A.5.3 Vista mobile	141
A.6 RecipeConfig	142
A.6.1 Descrizione generale	142
A.6.2 Vista desktop	142
A.6.3 Vista mobile	143
B Descrizione Design Pattern	144
B.1 Design pattern architetturali	144
B.1.1 Three-Tier	144
B.1.2 MVC	145
B.1.3 MVVM	147
B.1.4 Dependency Injection	148
B.2 Design pattern creazionali	149
B.2.1 Prototype Pattern	149
B.2.2 Module Pattern	150
B.2.3 Singleton	151
B.3 Design pattern strutturali	152
B.3.1 Façade	152
B.3.2 Adapter	153
B.4 Design pattern comportamentali	154
B.4.1 Page Controller	154
B.4.2 Template View	155
B.4.3 Template Method	156
B.4.4 Command	157

Elenco delle figure

1	Three-Tier	8
2	Package - client	11
3	Package - client::model	13
4	Package - client::model::data	14
5	Package - client::model	17
6	Package - client::view	23
7	Package - client::view::public	25
8	Package - client::view::user	26
9	Package - client::view::admin	31
10	Package - client::controller	34
11	Package - client::controller::public	35
12	Package - client::controller::user	37
13	Package - client::controller::admin	44
14	Package - server	48
15	Package - server::db	50
16	Package - server::db::raw_data	51
17	Package - server::db::raw_data::fb	52
18	Package - server::db::raw_data::tw	55
19	Package - server::db::raw_data::ig	57
20	Package - server::db::app_data	59
21	Package - server::processor	64
22	Package - server::processor::commands	65
23	Package - server::processor::commands::user	66
24	Package - server::processor::commands::recipe	69
25	Package - server::processor::commands::requests	71
26	Package - server::processor::commands::social	73
27	Package - server::miner	78
28	Package - server::miner::fb	80
29	Package - server::miner::tw	83
30	Package - server::miner::ig	85
31	Package - server::endpoints	87
32	Package - server::endpoints::api	88
33	Package - server::endpoints::api::public	88
34	Package - server::endpoints::api::private	90
35	Package - server::endpoints::resp	92
36	Package - server::endpoints::resp::public	92
37	Package - server::endpoints::resp::public::fb	94
38	Package - server::endpoints::resp::public::tw	96
39	Package - server::endpoints::resp::public::ig	98
40	Package - server::endpoints::resp::private	100
41	Contestualizzazione Singleton - Server	112
42	Contestualizzazione Adapter - Client	113
43	Contestualizzazione Front Controller - Server	114
44	Contestualizzazione Template View - Client	115
45	Contestualizzazione Command - Server	116
46	Contestualizzazione Template Method - Client	117
47	Contestualizzazione Template Method - Server	117
48	Page - Login (vista desktop)	134
49	Page - Login (vista mobile)	135

50	Page - Register (vista desktop)	136
51	Page - Register (vista mobile)	136
52	Page - Recipe (vista desktop)	137
53	Page - Recipe (vista mobile)	138
54	Page - Metrics (vista desktop)	139
55	Page - Metrics (vista mobile)	139
56	Page - Settings (vista desktop)	140
57	Page - Settings (vista mobile)	141
58	Page - Recipe Config (vista desktop)	142
59	Page - Recipe Config (vista mobile)	143
60	Design pattern architettonale - Three-Tier	144
61	Design pattern architettonale - MVC	145
62	Design pattern architettonale - MVVM	147
63	Design pattern architettonale - Dependency Injection	148
64	Design pattern creazionale - Prototype Pattern	149
65	Design pattern creazionale - Module Pattern	150
66	Design pattern creazionale - Singleton	151
67	Design pattern strutturale - Façade	152
68	Design pattern strutturale - Adapter	153
69	Design pattern comportamentale - Page Controller	154
70	Design pattern comportamentale - Template View	155
71	Design pattern comportamentale - Template Method	156
72	Design pattern comportamentale - Command	157

1 Introduzione

1.1 Scopo del documento

Questo documento ha come scopo quello di definire la progettazione ad alto livello per il prodotto BDSMApp. Verrà presentata l'architettura generale secondo la quale saranno organizzate le varie componenti software e i Design Pattern utilizzati nella creazione del prodotto. Verrà inoltre dettagliato il tracciamento tra le componenti software individuate ed i requisiti.

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 v3.0.0*. 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

- **Norme di Progetto:** *Norme di Progetto v3.0.0*

1.4.2 Informativi

- **Progettazione Software:** <http://www.math.unipd.it/~tullio/IS-1/2014/Dispense/P09.pdf>
- **Introduzione ai Design Pattern:** vengono utilizzate tutte le slide fornite dal professore Cardin presenti ai seguenti indirizzi:
 - <http://www.math.unipd.it/~tullio/IS-1/2014/>
 - **Design Pattern MVC, MVP e MVVM:** <http://www.math.unipd.it/~rcardin/sweb.html>
- **Python Design Pattern:** <https://github.com/faif/python-patterns>
- **Google Endpoints:** <https://cloud.google.com/appengine/docs/java/endpoints/>
- **Google Datastore:** <https://cloud.google.com/appengine/docs/python/datastore/datamodeling>
- **AngularJS Pattern:** <https://github.com/mgechev/angularjs-in-patterns>

- **Javascript OOP:** https://developer.mozilla.org/it/docs/Web/JavaScript/Introduzione_al_carattere_Object-Oriented_di_JavaScript
- **Template View Pattern:** <http://martinfowler.com/eaaCatalog/templateView.html>
- **ChartJS:** <http://www.chartjs.org/docs/>
- **tc-angular-chartjs:** <https://github.com/carlraig/tc-angular-chartjs>
- **Google Chart:** <https://developers.google.com/chart/interactive/docs/gallery/geochart>
- **angular-google-chart:** <https://github.com/bouil/angular-google-chart>
- **Bootstrap:** <https://angular-ui.github.io/bootstrap/>

2 Tecnologie utilizzate

In questa sezione verranno descritte le tecnologie su cui si basa lo sviluppo del progetto. Per ognuna di esse, verrà indicato l'ambito di utilizzo della tecnologia ed i vantaggi/svantaggi che ne derivano.

2.1 Linguaggi

2.1.1 CSS3

CSS (Cascading Style Sheets) è il linguaggio che verrà utilizzato per la formattazione delle pagine web offerte dal sistema qualora non bastassero i costrutti forniti dal framework Bootstrap 2.2.2.

Pro:

- permette una buona separazione dal contenuto della pagina rispetto a come verrà visualizzata. Questo garantisce un maggiore controllo sull'aspetto grafico e una più facile manutenzione.

2.1.2 HTML5

HTML5_G è il linguaggio di markup_G che verrà utilizzato per la strutturazione delle pagine web che l'applicazione andrà ad offrire sia per gli utenti che per gli amministratori del sistema.

Pro:

- permette di definire in maniera semplice la struttura delle pagine web;
- permette una maggiore semantica della pagina web, garantendo così una migliore indicizzazione da parte dei motori di ricerca;
- presenza di una vasta documentazione a supporto per i membri del team.

2.1.3 JavaScript

JavaScript_G è il linguaggio di scripting_G che verrà utilizzato lato client per la creazione e l'interazione con l'interfaccia utente.

Pro:

- permette alle pagine di essere dinamiche, reagendo a eventi scaturiti dall'utente;
- permette di validare in prima istanza i dati inseriti dagli utenti.

Contro:

- l'assenza di tipizzazione potrebbe ostacolare la valutazione della correttezza del codice;
- assenza dei costrutti tipici della programmazione ad oggetti come ad esempio le classi. Sarà compito del *Progettista* progettare, dove necessario, soluzioni implementativi per sopprimere a queste mancanze. I *Programmatori* dovranno seguire tali decisioni attraverso l'attuazione di norme di codifica presenti nelle *Norme di Progetto v3.0.0*;
- il codice è visibile e può essere letto da chiunque, mettendo quindi a rischio la sicurezza dell'utente nel caso venisse eseguito del codice malevolo.

2.1.4 JSON

JSON_G (JavaScript_G Object Notation) è il formato di dati scelto per lo scambio di informazioni tra il client e il server dell'applicazione.

Pro:

- utilizzo semplice tramite Javascript;
- formato utilizzato dai Google Endpoints che si andranno a sviluppare per i servizi REST_G resi disponibili dall'applicazione.

Contro:

- supporta meno tipi di dati nativi rispetto a JavaScript_G;
- non permette di specificare il tipo dei campi numerici salvati al suo interno;

2.1.5 Python

Python è il linguaggio utilizzato per gestire il back-end dell'applicazione. Viene scelta la versione 2.7.9 in quanto è quella che la Google App Engine_G mette a disposizione.

Pro:

- notevole velocità di apprendimento del linguaggio;
- presenza di una code convention molto forte;
- permette una potente manipolazione dei dati.

Contro:

- assenza di costrutti come lo switch;
- assenza di tipizzazione statica. Sarà compito del programmatore rendere esplicito il tipo delle variabili nel codice seguendo delle norme stabilite nel *Norme di Progetto v3.0.0*.

2.1.6 YAML

YAML (YAML Ain't a Markup_G Language) è il formato per la serializzazione di dati utilizzabile da esseri umani.

La sua scelta è stata vincolata all'utilizzo della Google App Engine_G con Python.

Pro:

- formato di una facile comprensione per un umano.

2.2 Framework

2.2.1 AngularJS

AngularJS_G è il framework_G JavaScript_G open source utilizzato per gestire il lato client dell'applicazione sia per quanto riguarda la parte HTML sia per la parte algorithmica. Permette di sviluppare più facilmente tutto ciò che riguarda l'applicazione web.

Pro:

- presenza di una vasta comunità a supporto;
- presenza di numerosi moduli per far fronte a diversi compiti non presenti naturalmente nel framework_G o che richiederebbero altrimenti un maggiore lavoro per la loro realizzazione;
- design pattern_G MVC e Dependency Injection già presente all'interno del framework_G.

Contro:

- si è molto vincolati all'ambiente, che non permette facili implementazioni con librerie esterne.

2.2.2 Twitter Bootstrap

Bootstrap è un insieme di elementi grafici, scritti in JavaScript_G e CSS3_G, che viene utilizzato per permettere di avere una grafica moderna e responsive dell'applicazione.

Pro:

- permette all'applicativo di essere responsive applicando facilmente delle regole già pronte;
- le diverse componenti vengono già incorporate direttamente nel framework_G AngularJS_G;
- presenza di una vasta comunità a supporto.

Contro:

- molto oneroso personalizzare pesantemente gli elementi grafici integrati

2.3 Librerie

2.3.1 Chart.js

Chart.js è la libreria di grafici in JavaScript utilizzata per la generazione della maggior parte dei grafici che l'utente andrà a vedere. La libreria è reperibile al seguente indirizzo: <http://www.chartjs.org/>.

Pro:

- permette ai grafici di essere responsive;
- presenza di diversi moduli implementativi per il framework_G AngularJS_G.

Contro:

- assenza di un Map Chart, che andrà cercato in un'altra libreria.

2.3.2 Google Chart

Google Chart è la libreria di grafici in JavaScript utilizzata per la generazione dei Geo Chart (Map Chart) reperibile al seguente indirizzo <https://developers.google.com/chart/interactive/docs/gallery/geochart>.

Pro:

- grosso supporto da parte della comunità;

- presenza di diversi moduli implementativi per il framework_G AngularJS_G in particolare di **angular-google-chart**, il più diffuso e supportato dalla comunità.

Contro:

- i grafici offerti, compresi quindi i Geo Chart utilizzati, non sono responsive. Bisognerà quindi che il team provveda a renderli tali.

2.3.3 facebook-sdk

Facebook-sdk è la libreria scelta per effettuare in maniera più semplice e astratta le chiamate alle API di Facebook. La libreria è reperibile al seguente indirizzo: <https://facebook-sdk.readthedocs.org/en/latest/>.

Pro:

- è la libreria maggiormente consigliata dalla comunità di Python.

2.3.4 tweepy.api

Tweepy.api è la libreria scelta per effettuare in maniera più semplice e astratta le chiamate alle API di Twitter. La libreria è reperibile al seguente indirizzo: <http://www.tweepy.org/>.

Pro:

- è la libreria maggiormente consigliata dalla comunità di Python.

Contro:

- alcune librerie utilizzate dalla suddetta, hanno causato qualche problema. In particolare la libreria **requests** con la versione 2.5.3 ha avuto dei problemi con la Google Cloud Platform, per questo si è deciso di utilizzare la versione 2.3.0. Anche la libreria **six** ha riscontrato dei problemi, ma sono stati risolti posizionando il file six.py nella root della libreria tweepy.api_G.

2.3.5 python-instagram

Python-instagram è la libreria scelta per effettuare in maniera più semplice e astratta le chiamate alle API di Instagram . La libreria è reperibile al seguente indirizzo: <https://github.com/Instagram/python-instagram>.

Pro:

- è la libreria maggiormente consigliata dalla comunità di Python.

2.3.6 ng-token-auth

ng-token-auth è un modulo di autenticazione basato sui token per AngularJS. Fornisce anche un supporto per la registrazione tramite altri sistemi attraverso il protocollo OAuth2. Il modulo è reperibile al seguente indirizzo: <https://github.com/lynndylanhurley/ng-token-auth>.

Pro:

- permette di effettuare tutte le operazioni relative alla gestione di un account utente andando a configurare in maniera semplice le API_G che dovranno essere chiamate nei diversi metodi offerti;

- fornisce numerosi eventi in risposta alle chiamate precedentemente citate per andare a gestire l'interfaccia utente e le diverse azioni che gli saranno messe;
- la libreria ha una coverage del 96%.

Contro:

- la libreria ha pochi contributori rispetto ad un'altra che si era valutata per eseguire lo stesso compito e cioè Satellizer (<https://github.com/sahat/satellizer>), quest'ultima però ha una coverage molto inferiore a quella scelta. Inoltre, per come si è deciso di progettare l'integrazione, non sarà particolarmente problematico decidere di sostituirla in caso non fosse più adeguata alle esigenze dell'applicativo.

2.4 Database

2.4.1 Google Cloud Datastore

Google Cloud Datastore è il database_G NoSQL che viene utilizzato per l'immagazzinamento di tutti i dati. È quello fornito dalla Google App Engine_G.

Pro:

- permette un forte livello di astrazione che nasconde la complessità dei database_G schema-less_G;
- presenza di un linguaggio (GQL) utilizzato per l'effettuazione di query con la stessa sintassi di MySQL.

Contro:

- forte accoppiamento tra le classi che modellano i dati e la comunicazione con il DBMS stesso.

3 Descrizione Architettura

3.1 Metodo e formalismo di specifica

La descrizione dell'architettura dell'applicazione viene fornita con un approccio di tipo top-down: si descrive l'architettura partendo dal generale e si analizzano via via in dettaglio tutti i moduli che le compongono. Vengono forniti degli esempi d'utilizzo dei design pattern_G scelti. I diagrammi delle componenti, delle classi e i diagrammi di attività rispettano il formalismo di UML_G 2.0.

Librerie esterne usate dall'applicazione verranno marcate con colori diversi per facilitare la distinzione durante la consultazione.

L'architettura generale non tratta l'intero insieme delle sottoclassi del sistema. Di conseguenza viene specificato lo scopo di una gerarchia e vengono individuate le relazioni con le varie componenti del sistema. Il resto dei dettagli viene rimandato alle diverse fasi di Progettazione di Dettaglio.

3.2 Architettura generale

L'architettura dell'applicazione segue il design pattern_G Three-Tier. Viene quindi suddivisa in tre livelli:

1. Client Tier
2. Server Tier
3. DataBase Tier

Di seguito viene proposto un diagramma rappresentante le relazioni tra i vari livelli. Vengono individuate le componenti che permettono ai vari livelli di interagire senza dover esporre la struttura interna degli stessi.

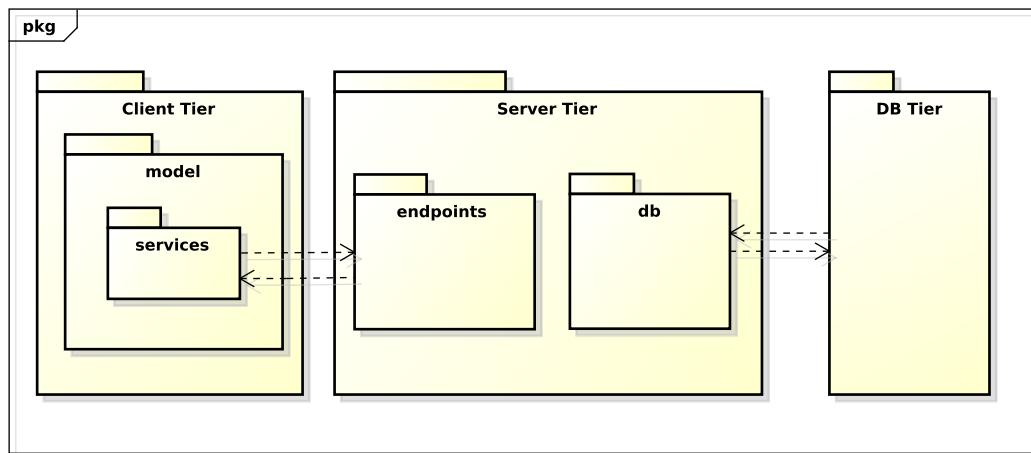


Figura 1: Three-Tier

Come indicato nel diagramma, il livello del client interagisce con il server tramite gli Endpoints forniti dal Server. Esso riceverà e analizzerà la richiesta. Una volta pronta, la risposta verrà fornita al Client.

Il client si occuperà di analizzarla per determinare se la richiesta ha avuto un esito positivo o negativo e nel primo caso utilizzerà i dati per effettuare le opportune operazioni.

3.2.1 Architettura Client Tier

Per il livello client è stato adottato il design pattern_G MVW offerto dal framework_G AngularJS_G. Questo livello implementa l’interfaccia grafica con cui utenti autenticati e amministratori possono interagire con il sistema in modo completo. Il diagramma seguente illustra l’interazione delle varie parti.

Gli utenti invocheranno determinate azioni utilizzando i pulsanti forniti dall’interfaccia e il sistema risponderà fornendo pagine di configurazione interattive oppure visualizzando una serie di grafici che verranno aggiornanti costantemente utilizzando i servizi REST_G forniti dall’applicazione. In dettaglio:

- Model: contiene il modello dei dati e la *business logic*, che comprende le definizioni delle chiamate che può fare l’applicazione e i servizi che offrono. Inserire la logica dell’applicazione nel modello anziché nel *controller* permette di ridurre la duplicazione del codice;
- View_G: contiene tutte le viste utente come astrazione del modello dei dati;
- Controller: questo modulo, in AngularJS_G, espone tutti i metodi forniti dalla logica dell’applicazione e permette l’associazione a due vie dei dati tra il livello astratto nelle viste e il livello concreto del modello.

Una illustrazione più dettagliata viene fornita nella sezione 4.1.

3.2.2 Architettura Server Tier

Il livello server si occupa di raccogliere i dati grezzi tramite le API_G fornite dai vari social network, elaborare e salvare tali dati nel database_G, fornire dei servizi REST_G ed elaborare le richieste in arrivo dal client. Il back-end è stato suddiviso nei seguenti moduli:

- db: contiene le definizioni del modello dei dati, sia per quanto riguarda i dati grezzi ricavati dai social network, sia per i dati necessari all’applicativo finale in sé;
- miner_G: contiene le classi ed i package_G che estraggono i dati grezzi a cadenza regolare e costruiscono nuove entità secondo il modello dei dati che andrà a popolare il database_G;
- processor_G: rappresenta il punto centrale del back-end. Si occupa di ricevere ed elaborare le richieste provenienti dal client oltre ad avviare il processo di aggiornamento delle Recipe_G scadute e recupero dei dati grezzi grazie alle funzionalità di *task_G scheduling* offerte da Google App Engine_G;
- endpoints: contengono le classi che definiscono la struttura dei servizi REST_G offerti tramite l’utilizzo del servizio Cloud Endpoints reso disponibile dalla piattaforma di Google.

Una illustrazione più dettagliata viene fornita nella sezione 4.2.

3.2.3 Architettura Database Tier

Il livello database_G è gestito dal sistema Datastore integrato nella Google Cloud Platform che fornisce la funzionalità di persistenza dei dati. Ad esso spetta il compito di memorizzazione sia tutti i risultati recuperati dal Miner_G riguardanti i dati grezzi dei social, sia di memorizzare le configurazioni del programma necessarie al Processor_G e lo stato di lavoro dei Miner ad esso collegati.

3.2.4 Protocollo di comunicazione Client-Server

La comunicazione tra il livello Client e quello Server avviene tramite i Google Cloud Endpoints. Questi permettono di creare delle API_G da un'applicazione generata tramite la Google App Engine_G. Gli Endpoints garantiscono un modo semplice di sviluppare un back-end condiviso, riducendo di molto il lavoro che sarebbe necessario per implementare una simile infrastruttura.

3.2.5 Protocollo di comunicazione Server-Database

La comunicazione tra il livello Server e quello del DBMS avviene tramite apposite funzionalità messe a disposizione dal servizio Datastore della piattaforma. Per poterle utilizzare il Server dovrà utilizzare il modulo: **google.appengine.ext.db**.

4 Componenti e Classi

4.1 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 riportare le classi di seguito identificate nelle strutture fornite dallo specifico linguaggio.

4.1.1 client

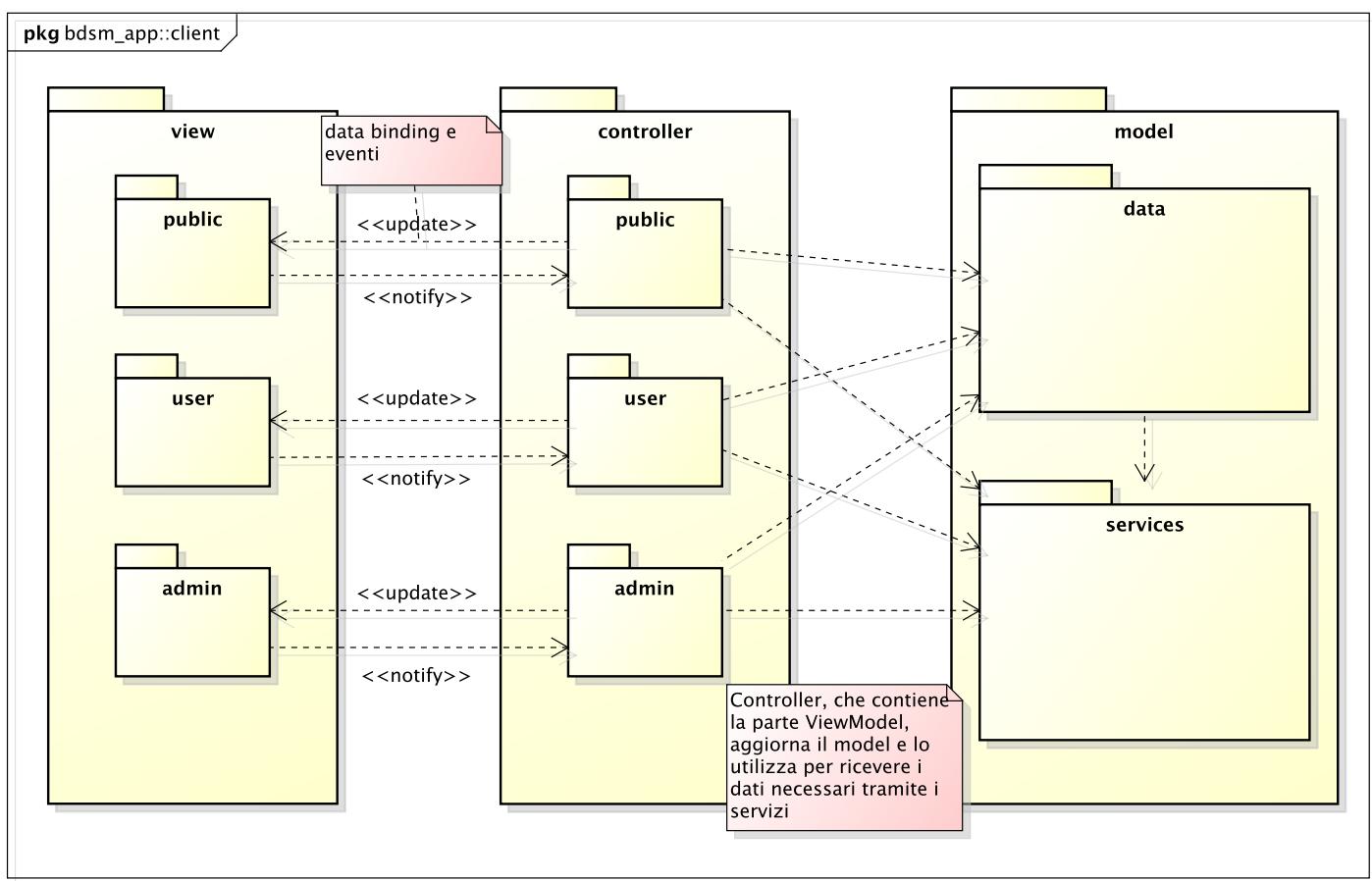


Figura 2: Package - client

- **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

- **Interazione con altri componenti:** interagisce con il server definito nella sezione 4.2 effettuando delle chiamate ai servizi REST esposti;

4.1.2 client::model

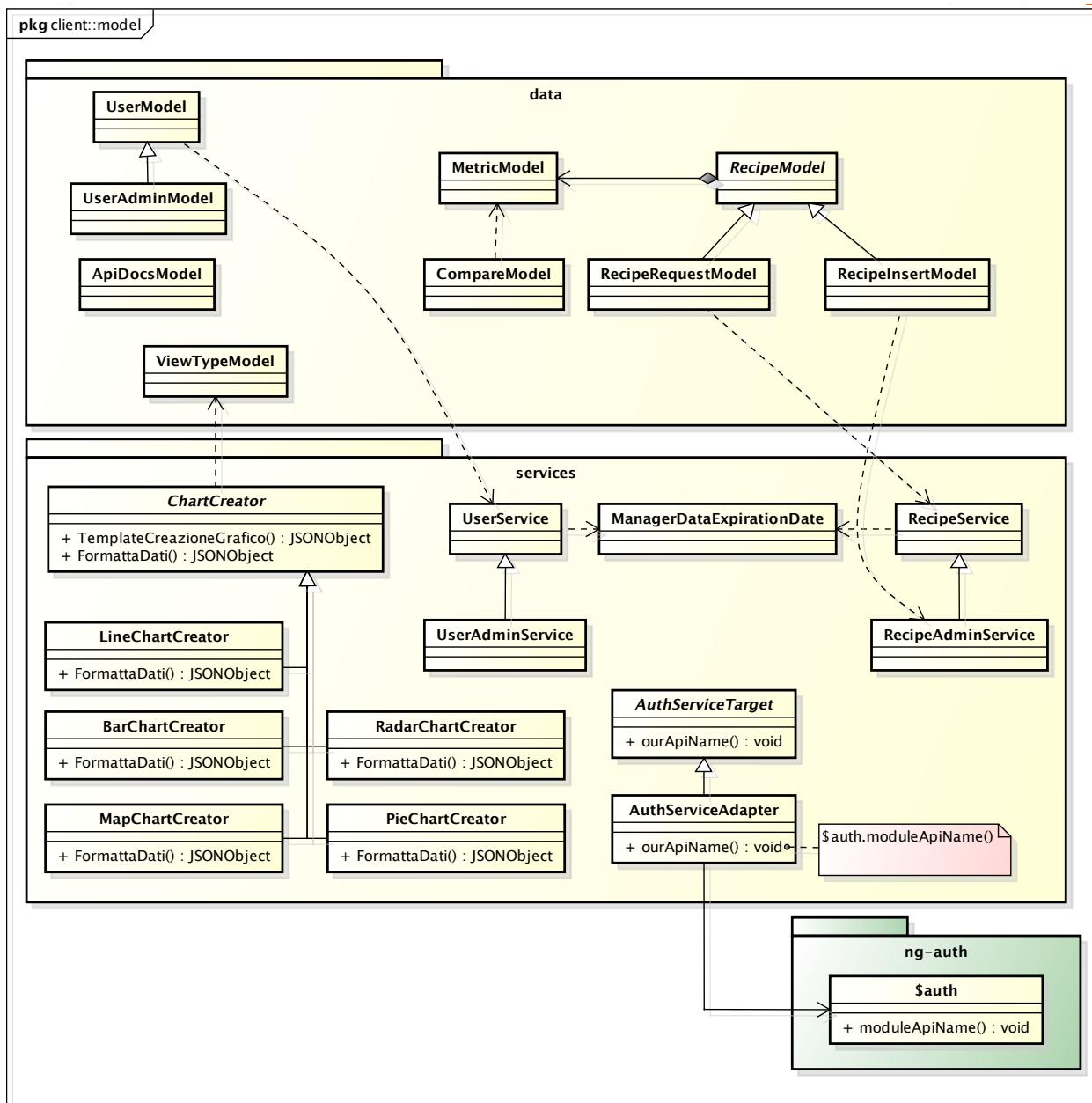


Figura 3: Package - client::model

- **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

4.1.3 client::model::data

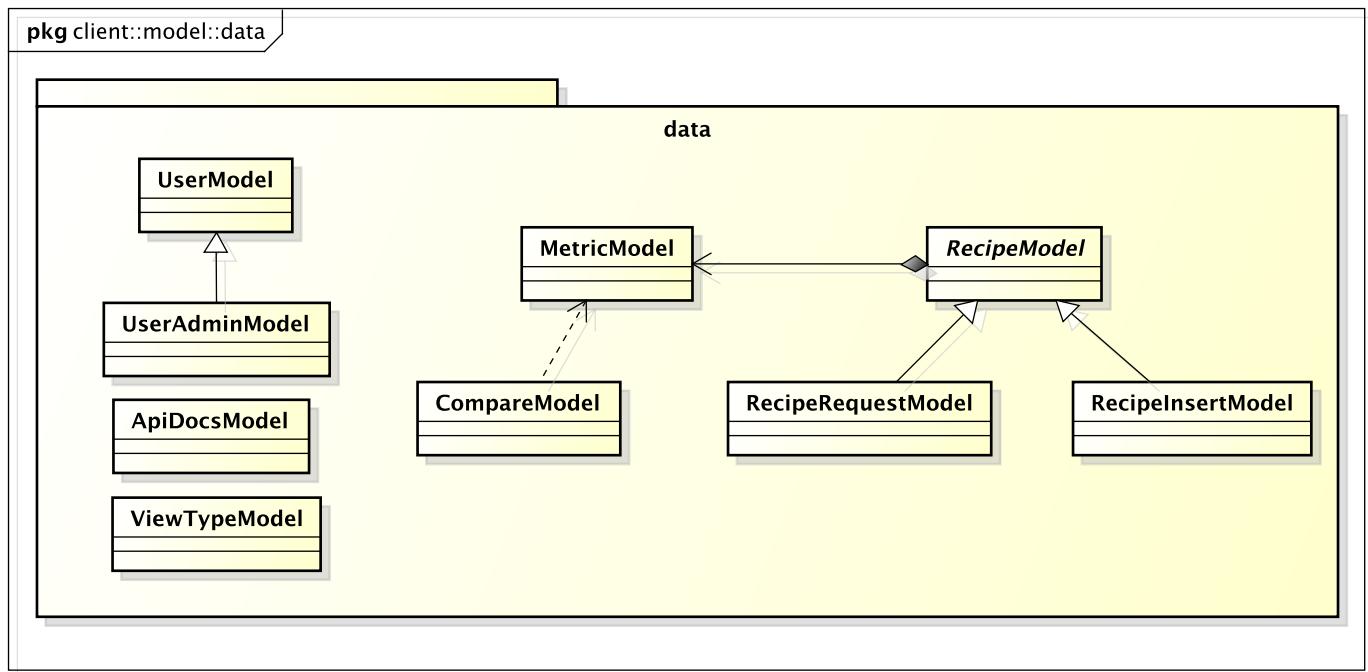


Figura 4: Package - client::model::data

- **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

4.1.3.1 Classi

4.1.3.1.1 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

4.1.3.1.2 client::model::data::ApiDocsModel

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

4.1.3.1.3 client::model::data::UserModel

- **Descrizione:** è la classe che rappresenta la struttura dati relativa all’utente normale;
- **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::controller::user::RecipeRequestCtrl
 - client::controller::user::SettingsCtrl
 - client::controller::user::TokenConfigCtrl

4.1.3.1.4 client::model::data::UserAdminModel

- **Descrizione:** è la classe che rappresenta la struttura dati relativa all’utente amministratore;
- **Utilizzo:** viene utilizzata da diversi servizi e controller per accedere agli attributi associati all’amministratore;
- **Classi ereditate:** client::model::data::UserModel;
- **Relazioni con altre classi:**
 - client::model::services::AuthService
 - client::model::services::UserAdminService
 - client::controller::admin::RecipeConfigCtrl
 - client::controller::admin::InsertRecipeCtrl

4.1.3.1.5 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

4.1.3.1.6 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

4.1.3.1.7 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

4.1.3.1.8 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

4.1.3.1.9 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

4.1.4 client::model::services

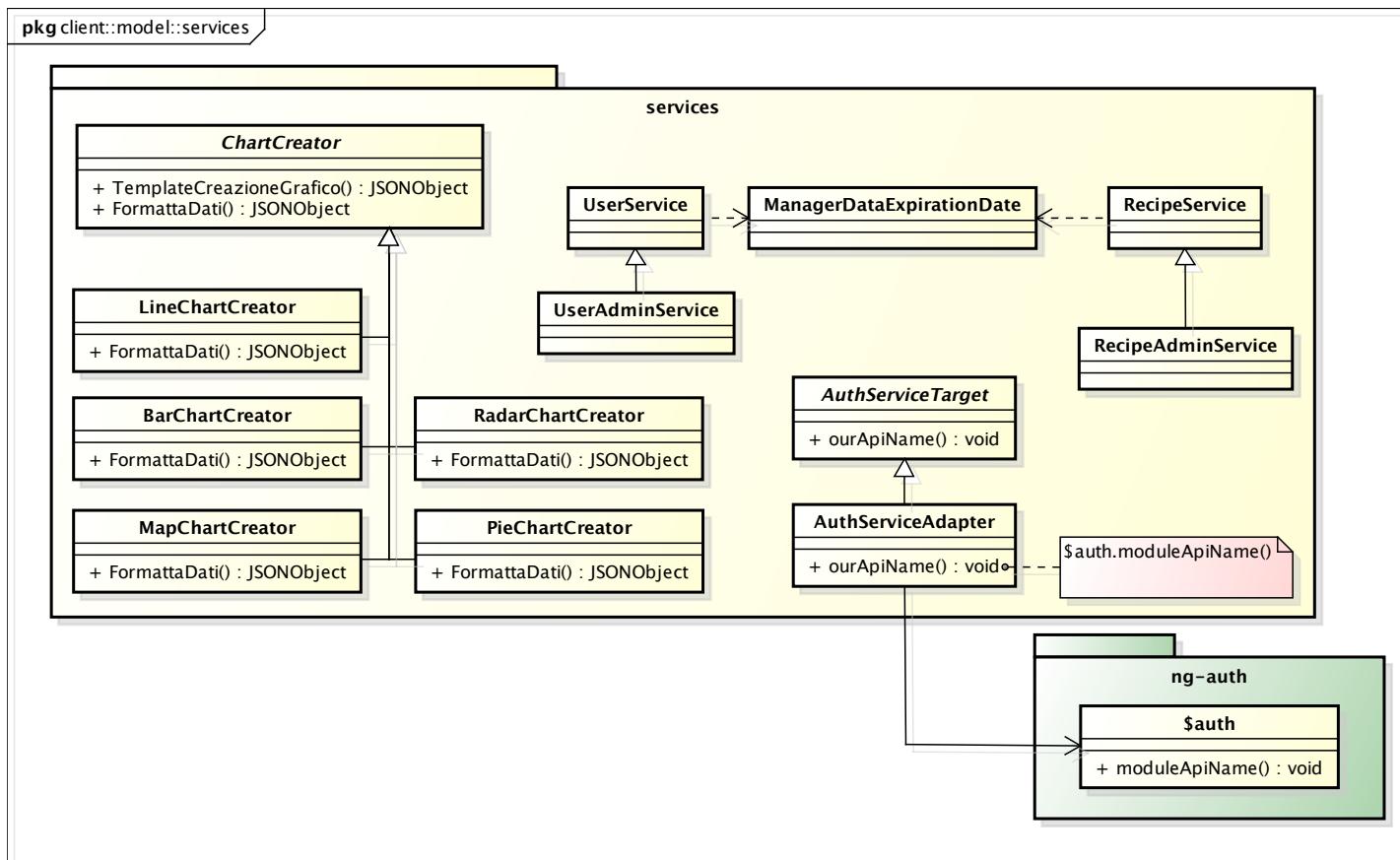


Figura 5: Package - client::model

- **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

4.1.4.1 Classi

4.1.4.1.1 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:** i metodi di questa classe saranno invocati quando un utente effettua la registrazione al servizio o effettua un login/logout;
- **Relazioni con altre classi:**
 - client::controller::public::LoginCtrl
 - client::controller::public::RegisterCtrl
 - client::controller::user::LogoutCtrl

4.1.4.1.2 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 AuthService-Target;
- **Utilizzo:** i metodi di questa classe saranno invocati quando un utente effettua la registrazione al servizio o effettua un login/logout;
- **Classi ereditate:**
 - client::model::services::AuthServiceTarget
- **Relazioni con altre classi:**
 - client::controller::public::LoginCtrl
 - client::controller::public::RegisterCtrl
 - client::controller::user::LogoutCtrl
 - ng-auth::\$auth

4.1.4.1.3 client::model::services::RecipeService

- **Descrizione:** p 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

4.1.4.1.4 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

4.1.4.1.5 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

4.1.4.1.6 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

4.1.4.1.7 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

4.1.4.1.8 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

4.1.4.1.9 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

4.1.4.1.10 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

4.1.4.1.11 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

4.1.4.1.12 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

4.1.4.1.13 client::model::services::ManagerDataExpirationDate

- **Descrizione:** questa classe 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

4.1.5 client::view

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

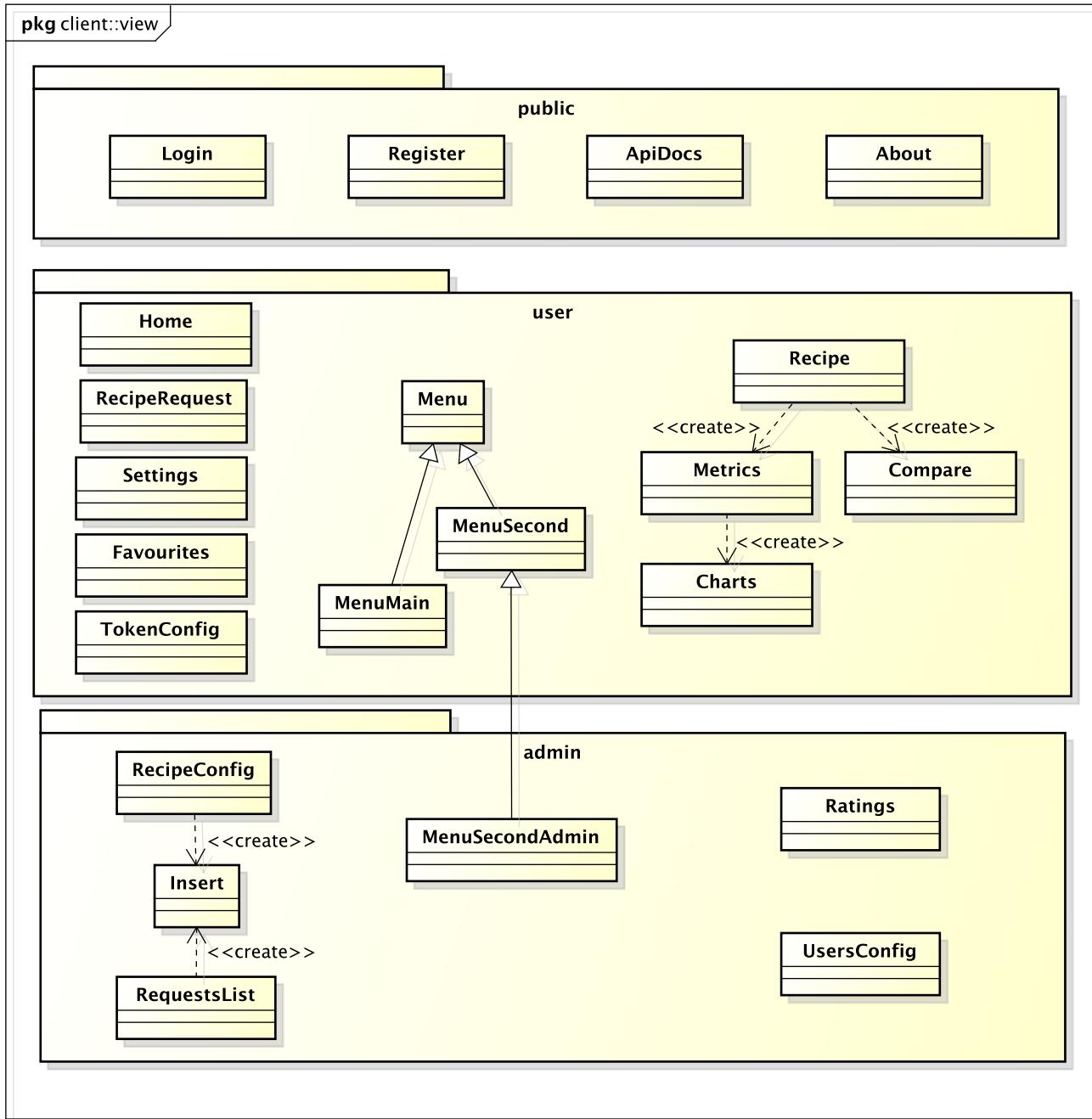


Figura 6: Package - client::view

- **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

4.1.6 client::view::public

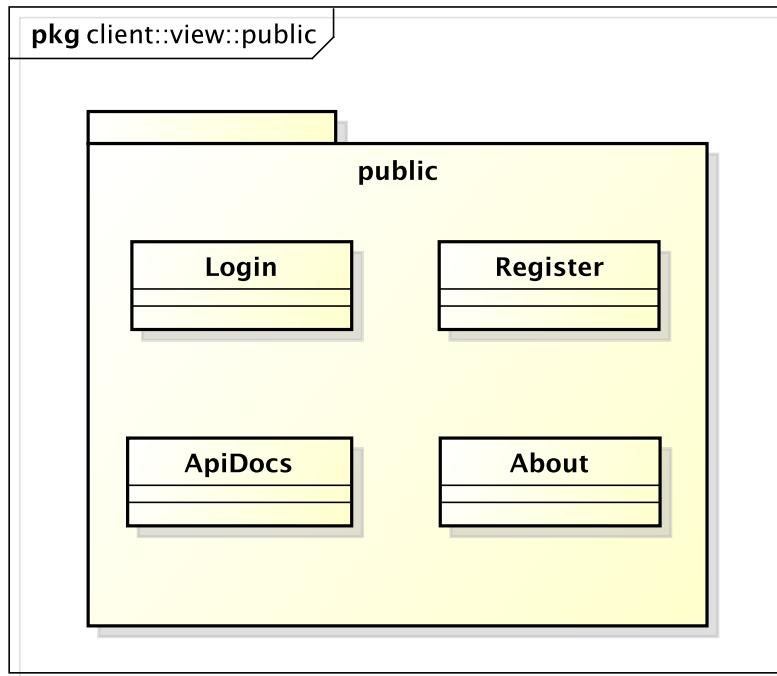


Figura 7: Package - client::view::public

- **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

4.1.6.1 Classi

4.1.6.1.1 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

4.1.6.1.2 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

4.1.6.1.3 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

4.1.6.1.4 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

4.1.7 client::view::user

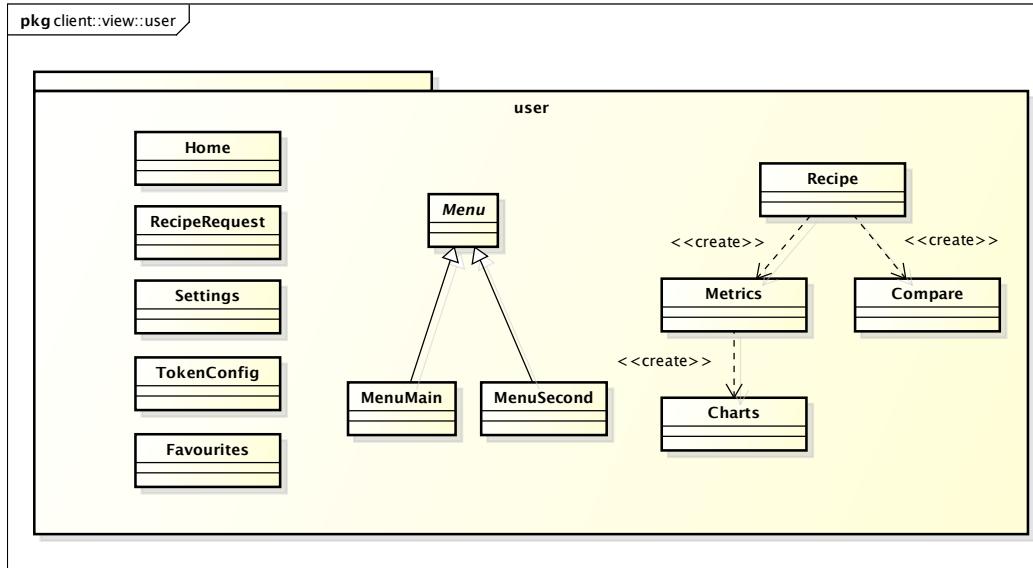


Figura 8: Package - client::view::user

- **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

4.1.7.1 Classi

4.1.7.1.1 client::view::user::Home

- **Descrizione:** la classe rappresenta un template_G HTML per la visualizzazione dell’interfaccia della home dell’applicazione;
- **Utilizzo:** viene utilizzata per generare la pagina HTML della home;
- **Relazioni con altre classi:**
 - client::view_G::user::MenuMain
 - client::view_G::user::MenuSecond
 - client::controller::public::HomeRoute
 - client::controller::user::HomeCtrl

4.1.7.1.2 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

4.1.7.1.3 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

4.1.7.1.4 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

4.1.7.1.5 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

4.1.7.1.6 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

4.1.7.1.7 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

4.1.7.1.8 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

4.1.7.1.9 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

4.1.7.1.10 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

4.1.7.1.11 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

4.1.7.1.12 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

4.1.8 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

4.1.8.1 Classi

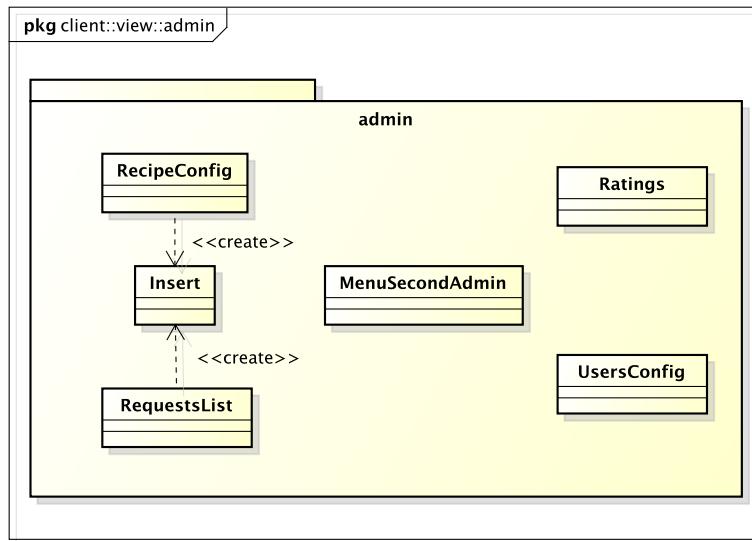


Figura 9: Package - client::view::admin

4.1.8.1.1 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

4.1.8.1.2 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

4.1.8.1.3 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

4.1.8.1.4 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

4.1.8.1.5 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

4.1.8.1.6 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

4.1.9 client::controller

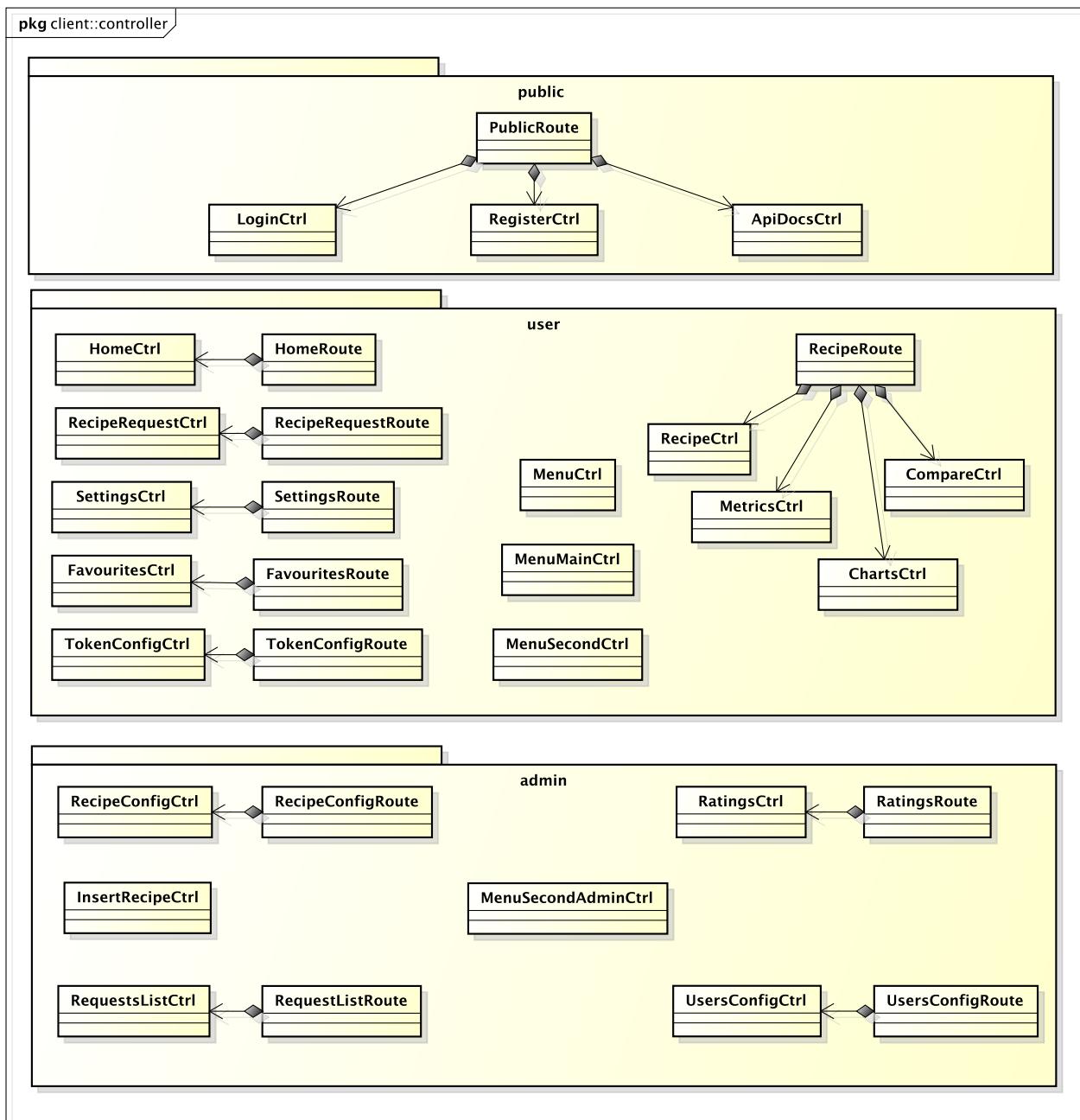


Figura 10: Package - client::controller

- **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

4.1.10 client::controller::public

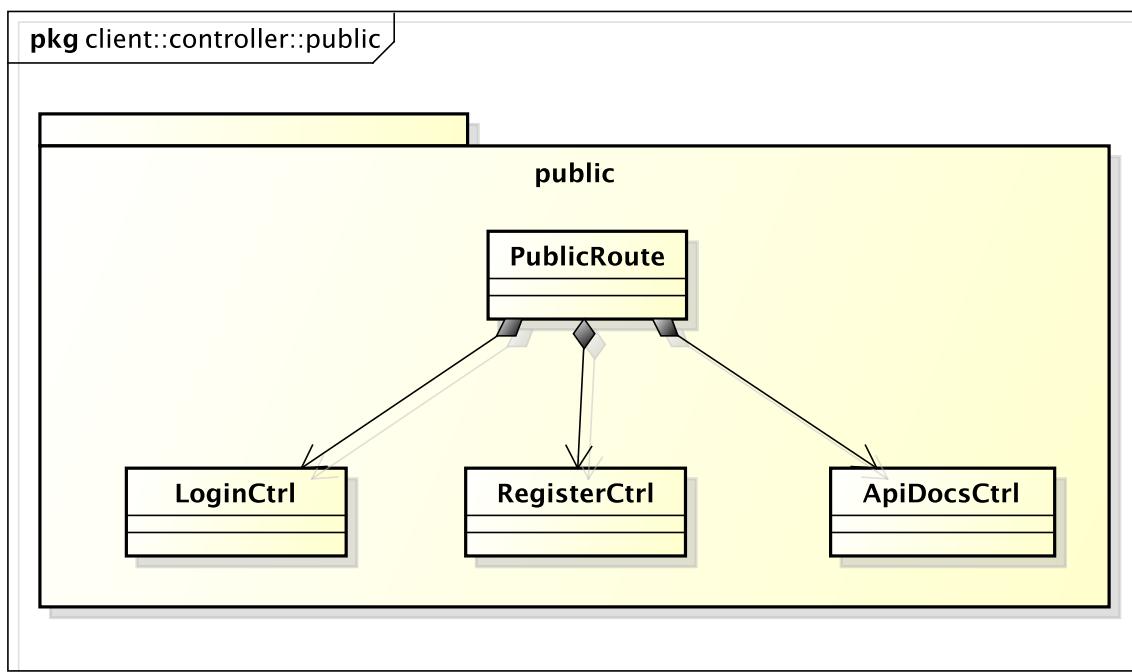


Figura 11: Package - client::controller::public

- **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

4.1.10.1 Classi

4.1.10.1.1 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

4.1.10.1.2 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

4.1.10.1.3 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

4.1.10.1.4 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

4.1.11 client::controller::user

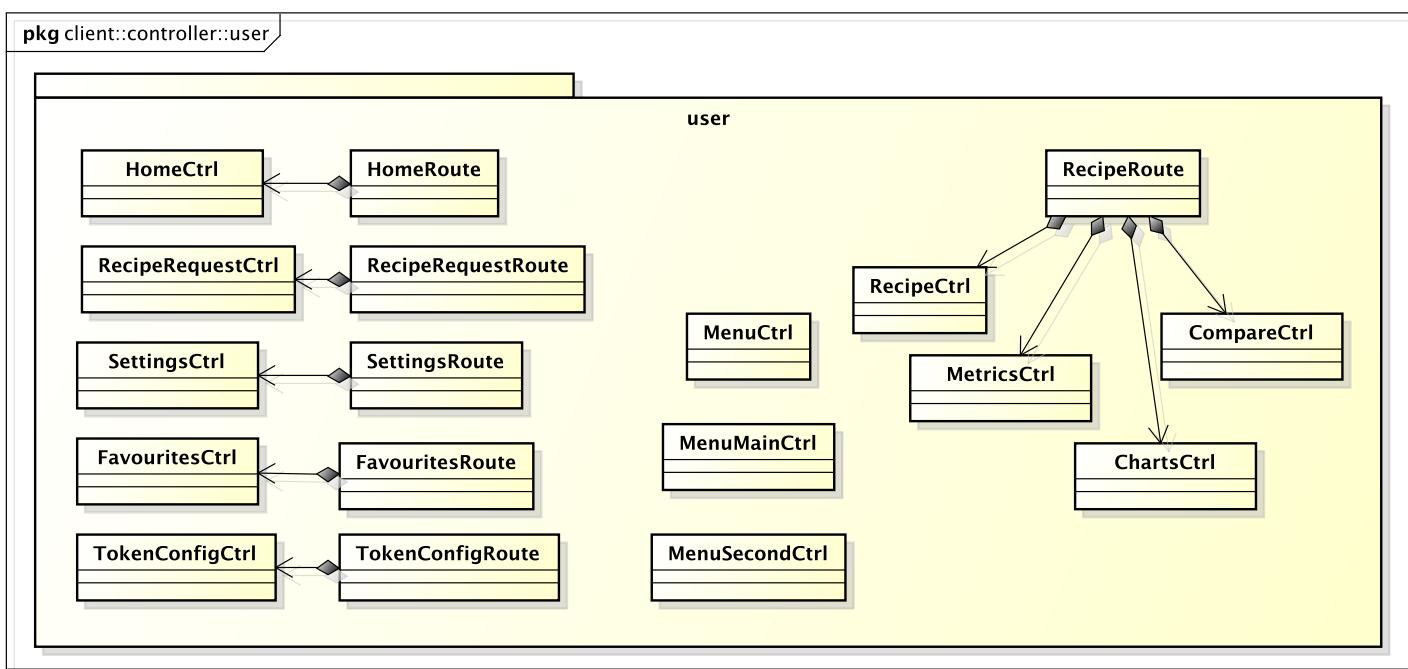


Figura 12: Package - client::controller::user

- **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

4.1.11.1 Classi

4.1.11.1.1 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

4.1.11.1.2 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

4.1.11.1.3 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

4.1.11.1.4 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

4.1.11.1.5 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

4.1.11.1.6 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

4.1.11.1.7 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

4.1.11.1.8 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;
- **Relazioni con altre classi:**
 - client::model::services::RecipeService
 - client::view_G::user::metrics
 - client::controller::user::RecipeRoute

4.1.11.1.9 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

4.1.11.1.10 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:**
 - client::model::data::CompareModel
 - client::view_G::user::Compare
 - client::controller::RecipeRoute

4.1.11.1.11 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

4.1.11.1.12 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

4.1.11.1.13 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

4.1.11.1.14 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

4.1.11.1.15 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

4.1.11.1.16 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

4.1.11.1.17 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 impostazione 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

4.1.11.1.18 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

4.1.11.1.19 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 reindirizzamento alla pagina di Login;
- **Relazioni con altre classi:**
 - client::model::services::AuthService
 - client::view_G::user::MenuMain
 - client::controller::public::PublicRoute

4.1.12 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

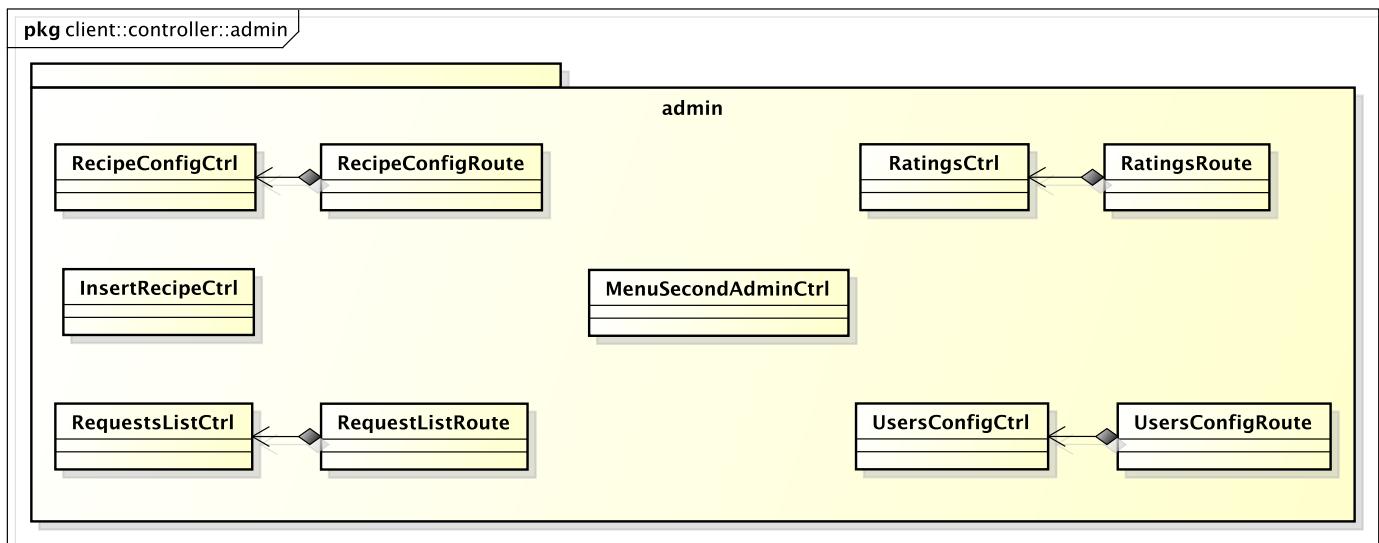


Figura 13: Package - `client::controller::admin`

4.1.12.1 Classi

4.1.12.1.1 `client::controller::admin::MenuSecondAdminController`

- **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`

4.1.12.1.2 `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

4.1.12.1.3 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

4.1.12.1.4 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

4.1.12.1.5 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

4.1.12.1.6 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

4.1.12.1.7 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

4.1.12.1.8 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

4.1.12.1.9 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

4.1.12.1.10 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

4.2 Server (Back-end)

4.2.1 server

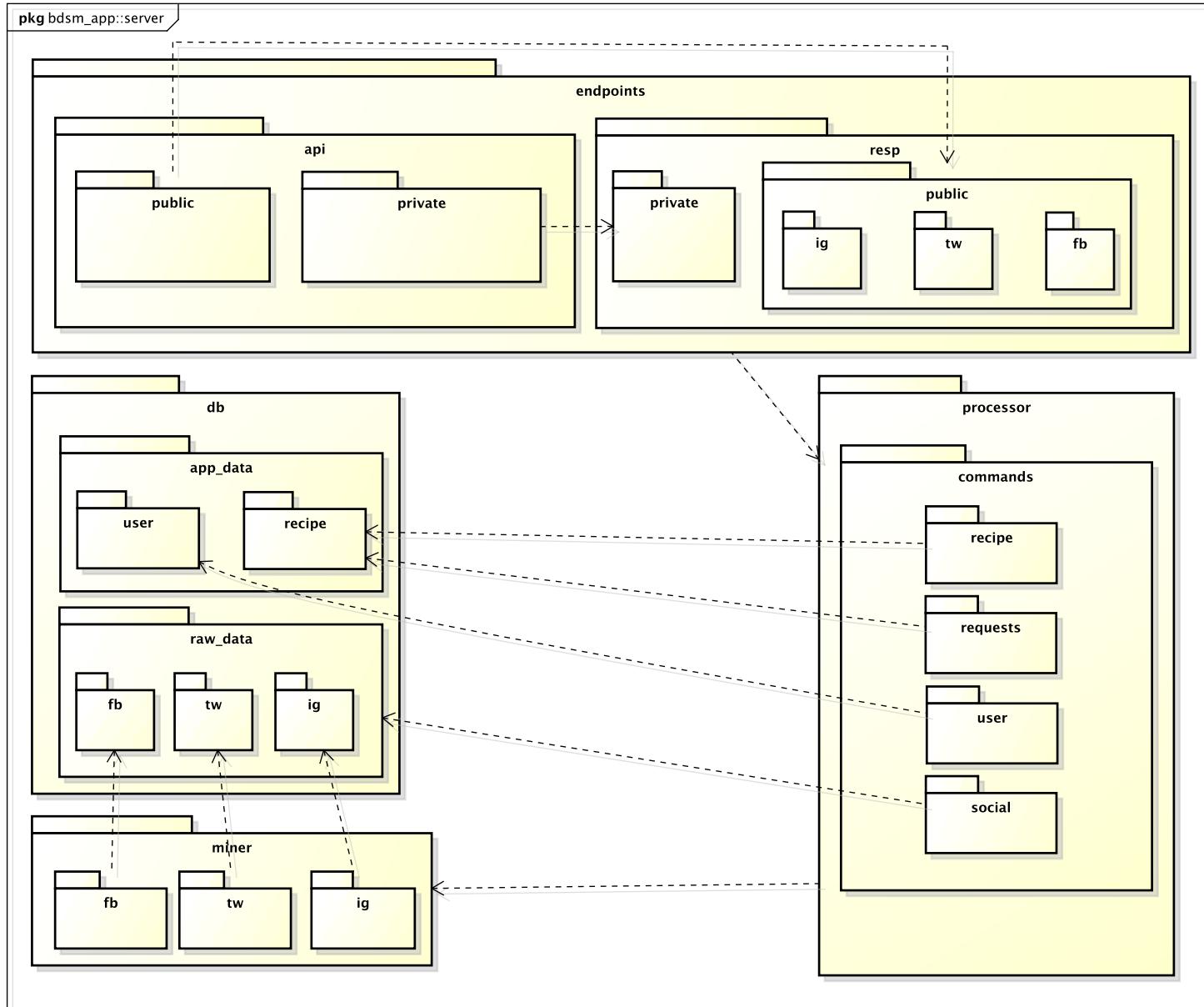


Figura 14: Package - server

- **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 4.1 tramite i servizi REST offerti dal modulo in questione;

4.2.2 server::db

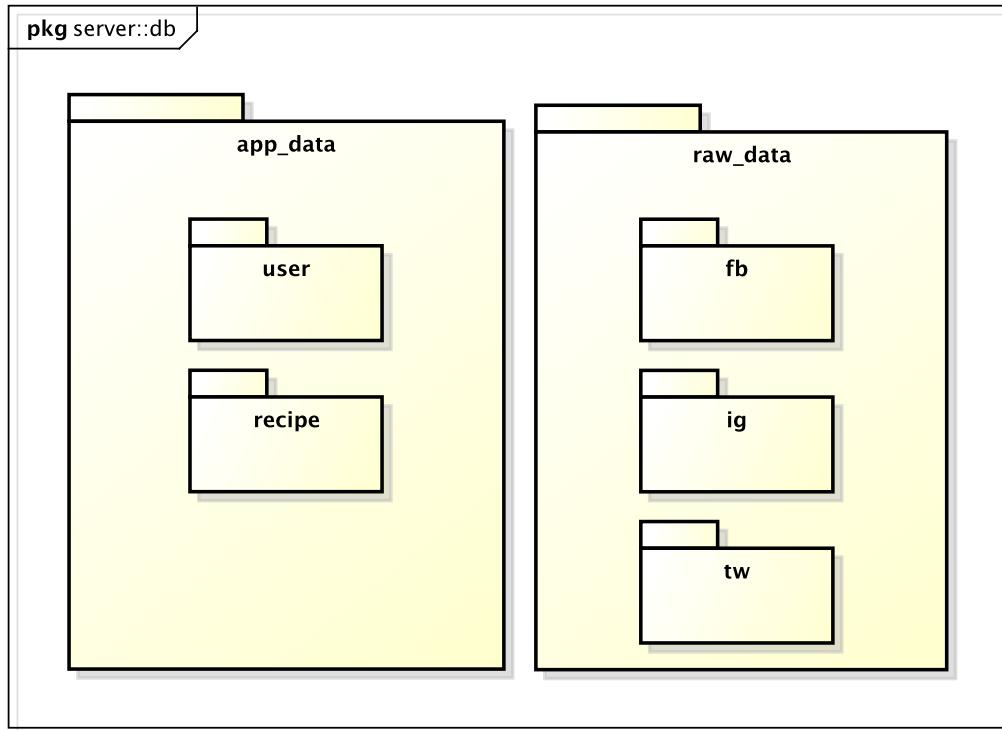


Figura 15: Package - server::db

- **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 dei parametri del software e degli utenti;
- **Padre:** server;
- **Package_G contenuti:**
 - server::app_data.
 - server::raw_data;

4.2.3 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

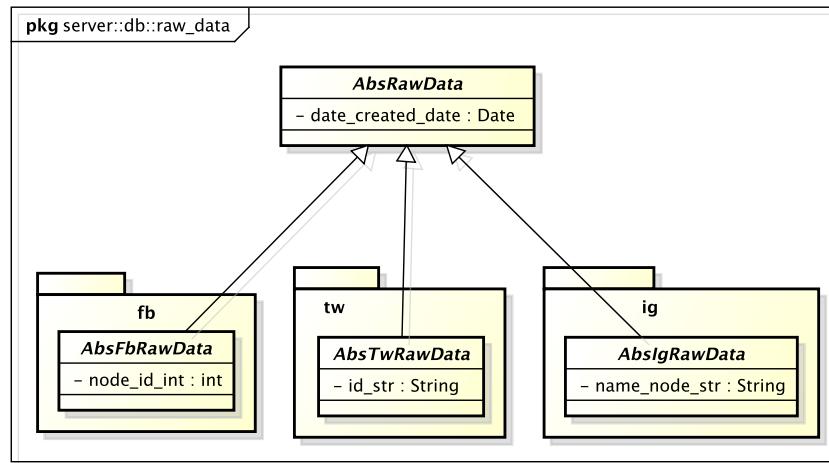


Figura 16: Package - server::db::raw_data

4.2.3.1 Classi

4.2.3.1.1 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::RawFbPostTrend
- server::db::raw_data::tw::RawTwUserTrend
- server::db::raw_data::tw::RawTwHashtagTrend
- server::db::raw_data::tw::RawTwUserTweet
- server::db::raw_data::ig::RawIgUserTrend
- server::db::raw_data::ig::RawIgHashtagTrend
- server::db::raw_data::ig::RawIgMedia

4.2.4 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

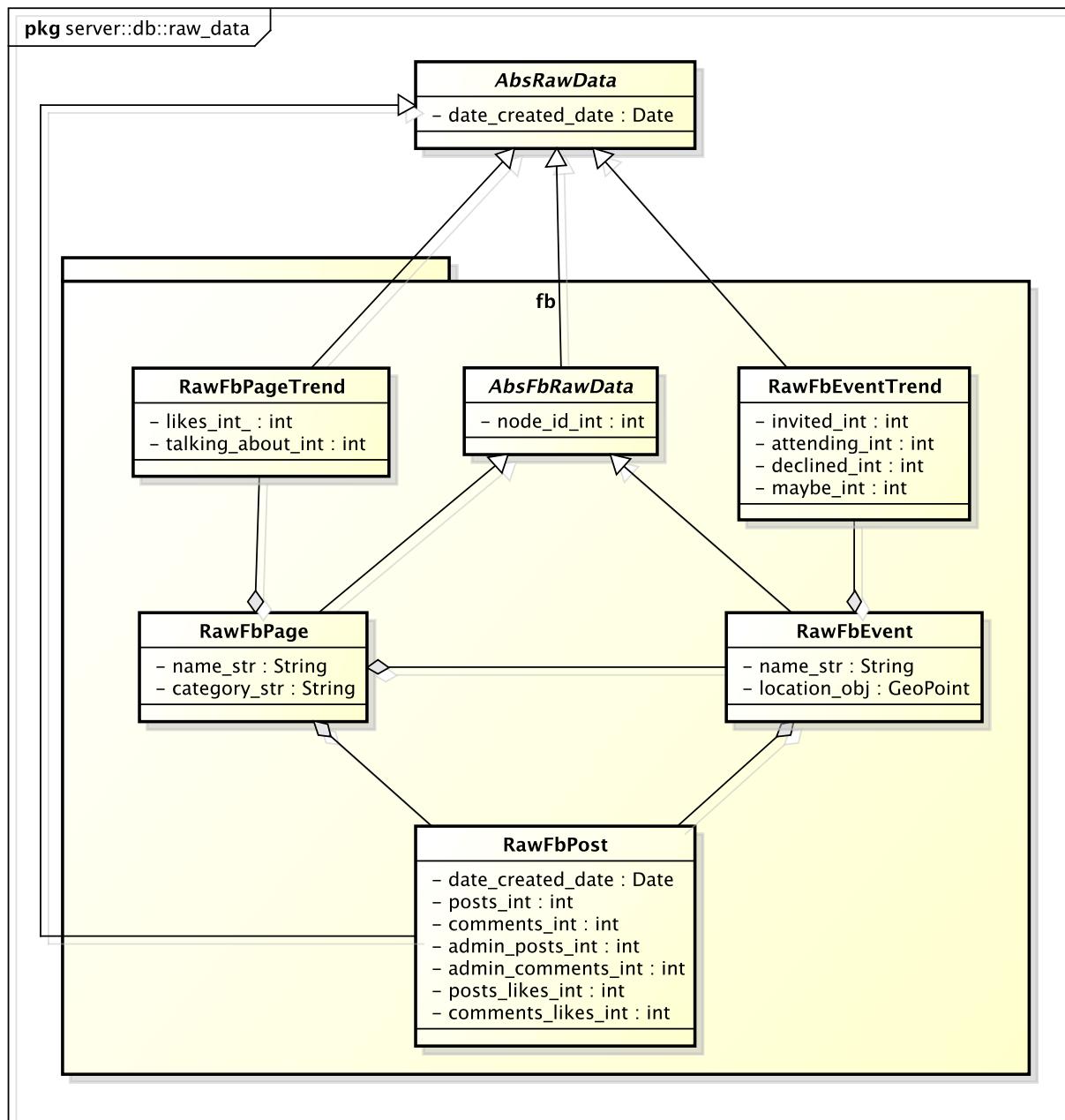


Figura 17: Package - `server::db::raw_data::fb`

4.2.4.1 Classi

4.2.4.1.1 server::db::raw_data::fb::AbsFbRawData

- **Descrizione:** classe astratta che definisce il modello 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

4.2.4.1.2 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::RawFbPost

4.2.4.1.3 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;
- **Classi ereditate:** server::db::raw_data::AbsRawData

4.2.4.1.4 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::RawFbPost

4.2.4.1.5 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;
- **Classi ereditate:** server::db::raw_data::AbsRawData

4.2.4.1.6 server::db::raw_data::fb::RawFbPost

- **Descrizione:** classe che rappresenta il modello del trend dei post di una pagina o di un evento su Facebook;
- **Utilizzo:** la classe fornisce metodi per sommare i dati statici sui post di Facebook;

4.2.5 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

4.2.5.1 Classi

4.2.5.1.1 server::db::raw_data::tw::AbsTwRawData

- **Descrizione:** classe astratta che definisce il modello 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

4.2.5.1.2 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

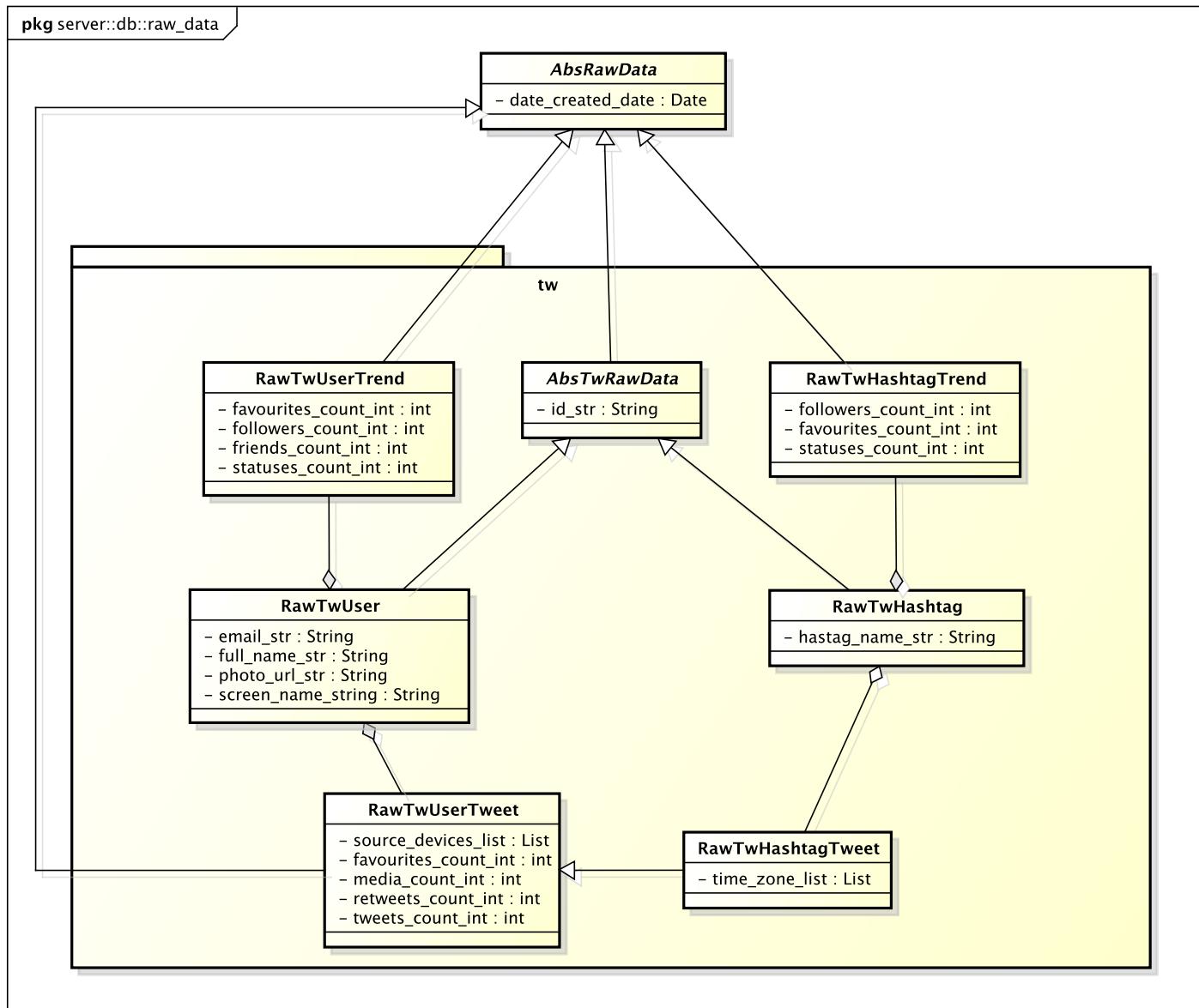


Figura 18: Package - `server::db::raw_data::tw`

4.2.5.1.3 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 una determinata persona;
- **Classi ereditate:** server::db::raw_data::AbsTwRawData

4.2.5.1.4 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

4.2.5.1.5 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. 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::RawTwHashtagTrend
 - server::db::raw_data::tw::RawTwHashtagTweet

4.2.5.1.6 server::db::raw_data::tw::RawTwHashtagTrend

- **Descrizione:** classe che definisce il modello dei dati del trend di un hashtag su Twitter;
- **Utilizzo:** la classe viene utilizzata per memorizzare il numero di favoriti, di followers e statuses di un determinato hashtag;
- **Classi ereditate:** server::db::raw_data::AbsTwRawData

4.2.5.1.7 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;
- **Classi ereditate:** server::db::raw_data::RawTwUserTweet

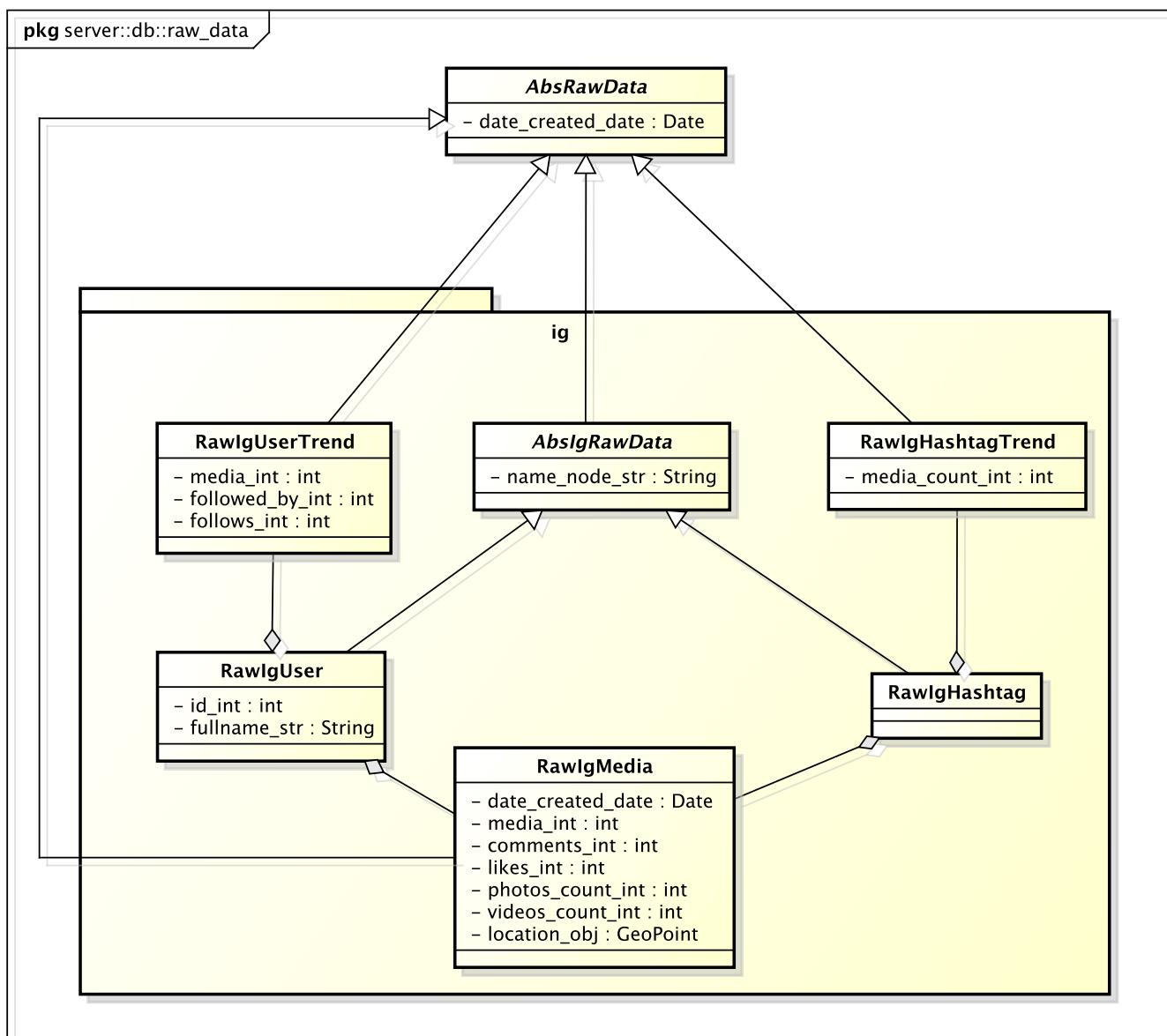


Figura 19: Package - `server::db::raw_data::ig`

4.2.6 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

4.2.6.1 Classi

4.2.6.1.1 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

4.2.6.1.2 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 le 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

4.2.6.1.3 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;
- **Classi ereditate:** server::db::raw_data::AbsRawData

4.2.6.1.4 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;
- **Classi ereditate:** server::db::raw_data::AbsIgRawData
- **Relazioni con altre classi:**
 - server::db::raw_data::ig::RawIgHashtagTrend
 - server::db::raw_data::ig::RawIgMedia

4.2.6.1.5 server::db::raw_data::ig::RawIgHashtagTrend

- **Descrizione:** classe che definisce il modello dei dati del trend di un hastag Instagram;
- **Utilizzo:** la classe viene utilizzata per memorizzare il numero di media caricati di un determinato hashtag;
- **Classi ereditate:** server::db::raw_data::AbsRawData

4.2.6.1.6 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 di un media caricato da un utente specifico su Instagram. Vengono forniti metodi automatici per il conteggio dei parametri che verranno utilizzati per seguire un trend;
- **Classi ereditate:** server::db::raw_data::AbsRawData

4.2.7 server::db::app_data

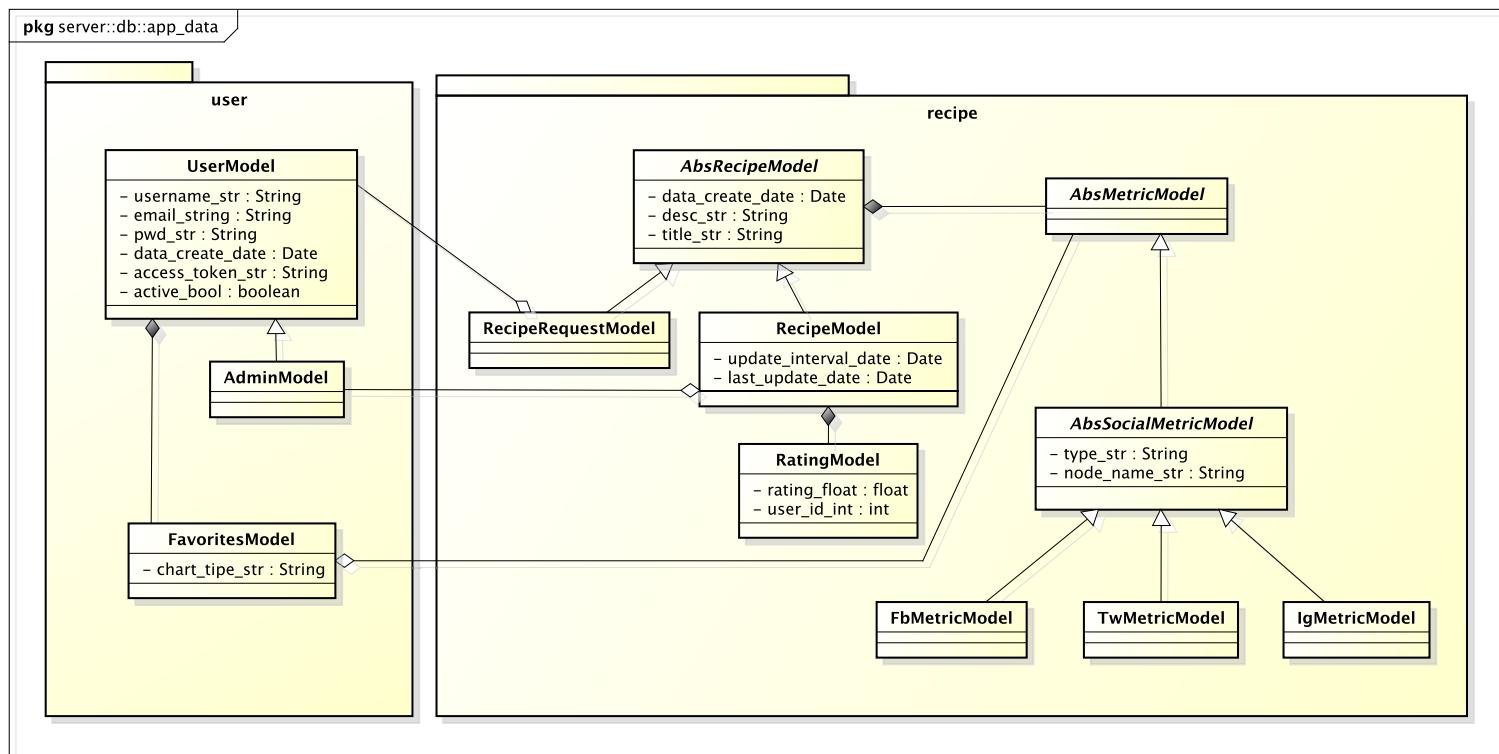


Figura 20: Package - `server::db::app_data`

- **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

4.2.8 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

4.2.8.1 Classi

4.2.8.1.1 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 conterrà i metodi per la creazione, eliminazione, modifica degli account utente;
- **Relazioni con altre classi:**
 - server::db::app_data::user::FavouritesModel
 - server::db::app_data::user::AdminModel
 - server::db::app_data::recipe_G::RecipeRequestModel

4.2.8.1.2 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 implementata con dei metodi per l'aggiunta la modifica e la rimozione;
- **Classi ereditate:** server::db::app_data::user::UserModel;
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::RecipeModel

4.2.8.1.3 server::db::app_data::user::FavouritesModel

- **Descrizione:** classe che definisce il modello dei dati relativo ai preferiti dell'utente;
- **Utilizzo:** la classe permette all'utente di aggiungere dei modelli di view_G nei preferiti. Contiene metodi per l'inserimento e rimozione di preferiti;
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::AbsMetricModel

4.2.9 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

4.2.9.1 Classi

4.2.9.1.1 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

4.2.9.1.2 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;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsRecipeModel

4.2.9.1.3 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
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::RatingModel

4.2.9.1.4 server::db::app_data::recipe::RatingModel

- **Descrizione:** classe che rappresenta il modello del rating di una Recipe_G;
- **Utilizzo:** viene utilizzata risalire al voto di ogni utente per una determinata Recipe_G;

4.2.9.1.5 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 ha il compito di inglobare tutte le sotto-metriche e non solo quelle derivate dai social media;
- **Relazioni con altre classi:**
 - server::db::app_data::recipe_G::AbsSocialMetricModel

4.2.9.1.6 server::db::app_data::recipe::AbsSocialMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative ai social media;
- **Utilizzo:** la classe ha il compito di raggruppare i social media da rappresentare e identificarli univocamente;
- **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

4.2.9.1.7 server::db::app_data::recipe::FbMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative a Facebook;
- **Utilizzo:** la classe ha il compito di identificare univocamente l'id della pagina o l'id dell'evento da analizzare. La classe fornisce metodi per l'aggiunta e la rimozione ricorsiva in caso di cancellazione;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsSocialMetricModel;

4.2.9.1.8 server::db::app_data::recipe::IgMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative a Instagram;
- **Utilizzo:** la classe ha il compito di identificare univocamente l'id della pagina o l'hashtag da analizzare. La classe fornisce metodi per l'aggiunta e la rimozione ricorsiva in caso di cancellazione;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsSocialMetricModel;

4.2.9.1.9 server::db::app_data::recipe::TwMetricModel

- **Descrizione:** classe che definisce il modello dei dati delle metriche relative a Twitter;
- **Utilizzo:** la classe ha il compito di identificare univocamente l'id della pagina o l'hashtag da analizzare. La classe fornisce metodi per l'aggiunta e la rimozione ricorsiva in caso di cancellazione;
- **Classi ereditate:** server::db::app_data::recipe_G::AbsSocialMetricModel;

4.2.10 server::processor

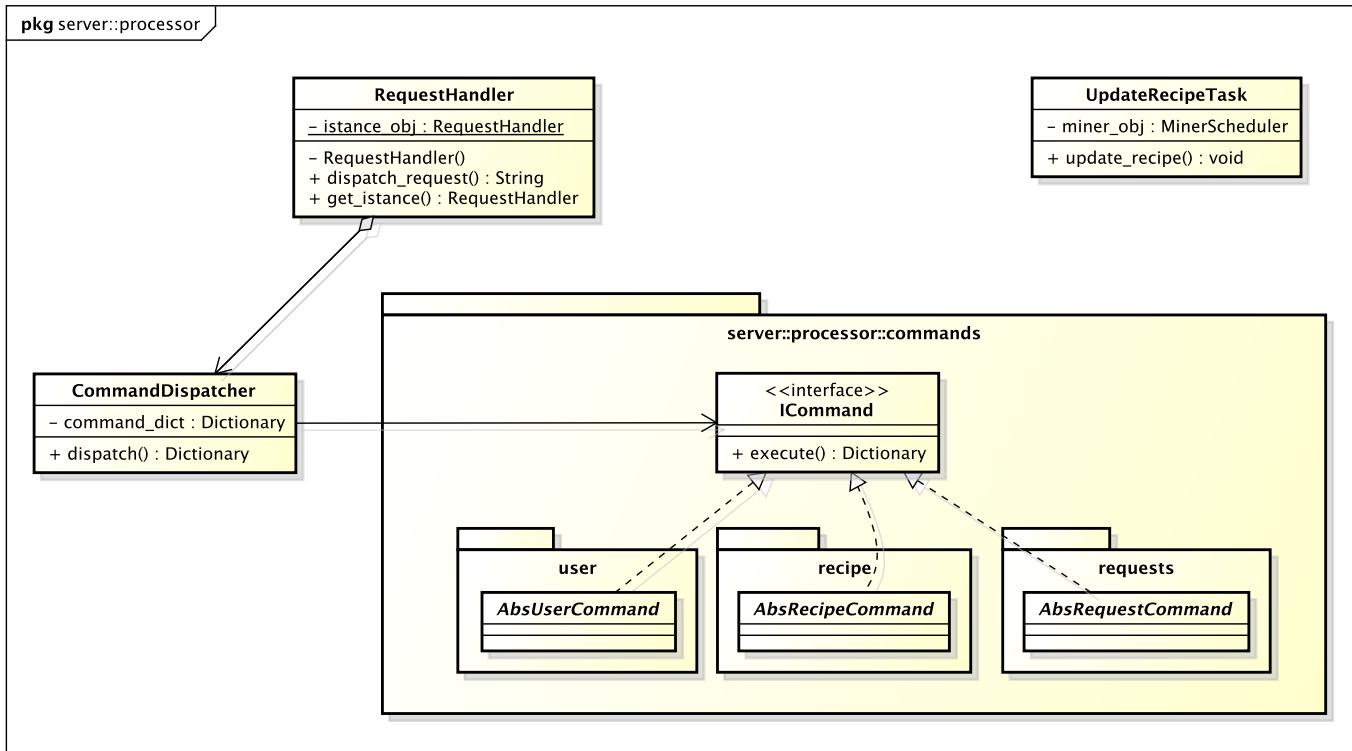


Figura 21: Package - `server::processor`

- **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`

4.2.10.1 Classi

4.2.10.1.1 `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::processor_G::CommandDispatcher

4.2.10.1.2 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

4.2.10.1.3 server::processor::UpdateRecipeTask

- **Descrizione:** è la classe che riceve la notifica dal Cron_G (funzionalità integrata nella Google Cloud Platform) per aggiornare le Recipe_G memorizzate nel sistema interagendo con la classe server::miner_G::MinerScheduler a tale scopo;
- **Utilizzo:** si occupa di invocare il MinerScheduler avviando il processo di aggiornamento delle Recipe_G passando al modulo Miner_G la lista delle ricette da aggiornare;
- Relazioni con altre classi:
 - server::miner_G::MinerScheduler

4.2.11 server::processor::commands

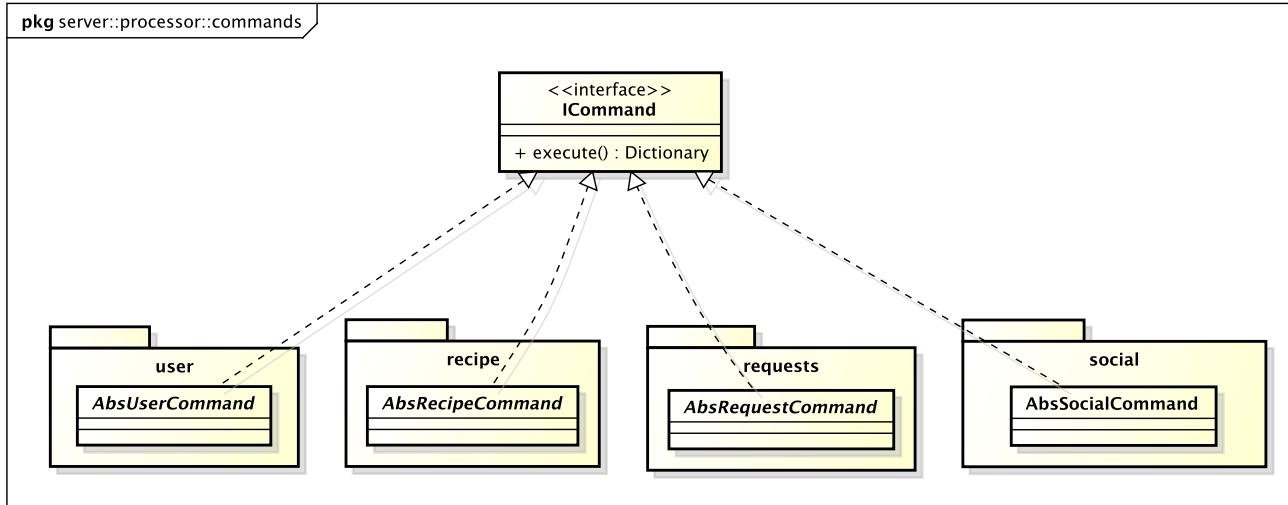


Figura 22: Package - server::processor::commands

- **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

4.2.11.1 Classi

4.2.11.1.1 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;

4.2.12 server::processor::commands::user

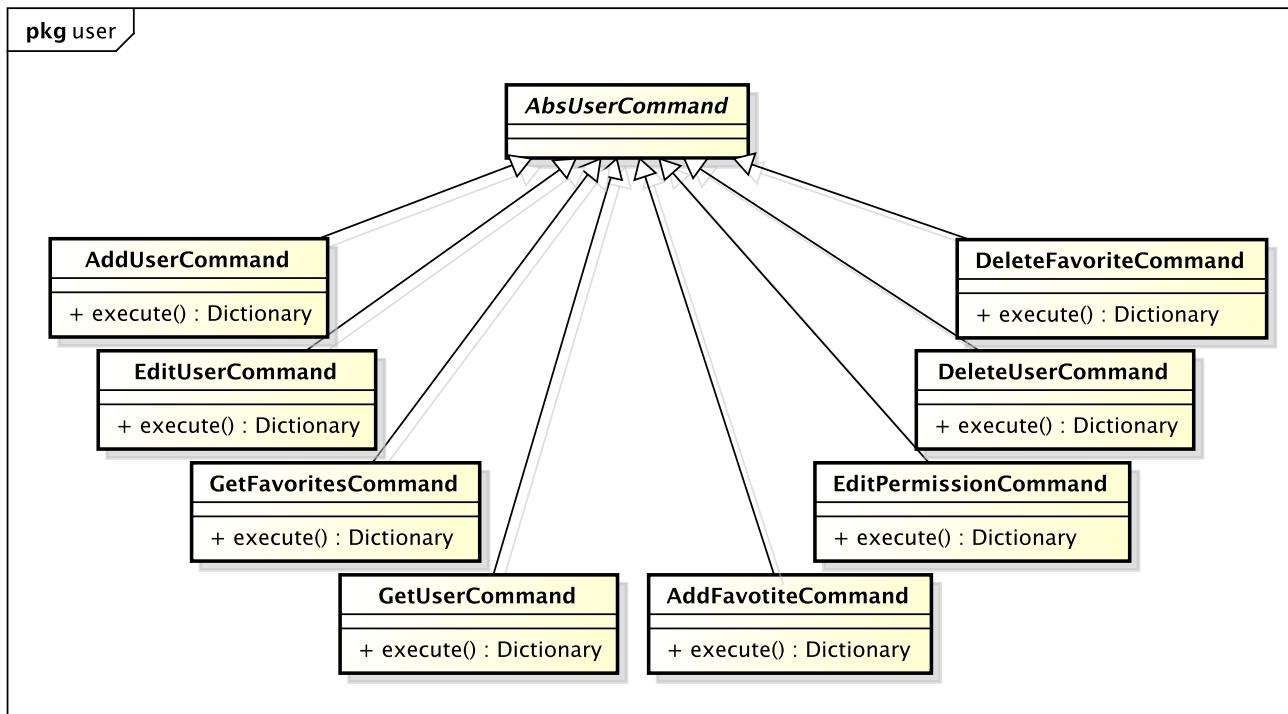


Figura 23: Package - server::processor::commands::user

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

4.2.12.1 Classi

4.2.12.1.1 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

4.2.12.1.2 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

4.2.12.1.3 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

4.2.12.1.4 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

4.2.12.1.5 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

4.2.12.1.6 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

4.2.12.1.7 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

4.2.12.1.8 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

4.2.12.1.9 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

4.2.13 server::processor::commands::recipe

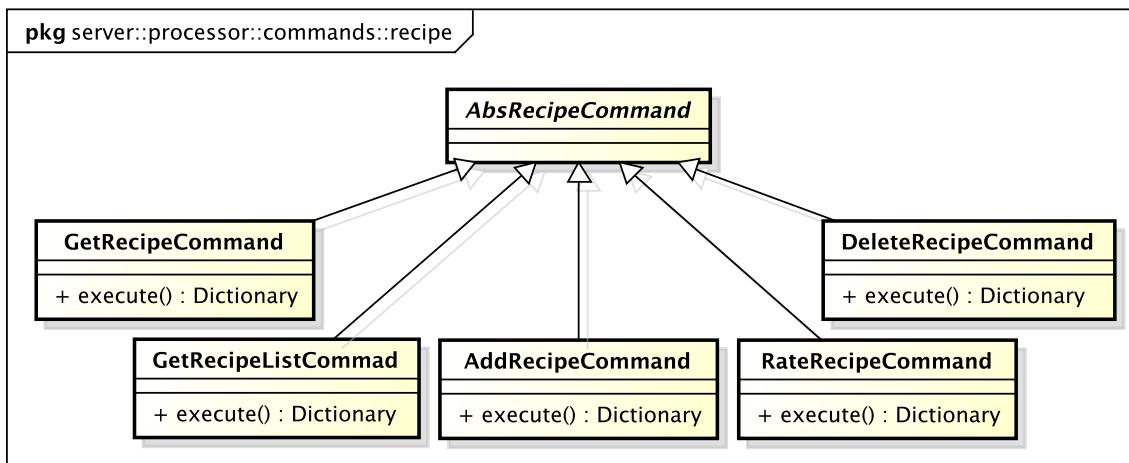


Figura 24: Package - server::processor::commands::recipe

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

4.2.13.1 Classi

4.2.13.1.1 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;

4.2.13.1.2 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

4.2.13.1.3 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

4.2.13.1.4 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

4.2.13.1.5 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

4.2.13.1.6 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

4.2.14 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

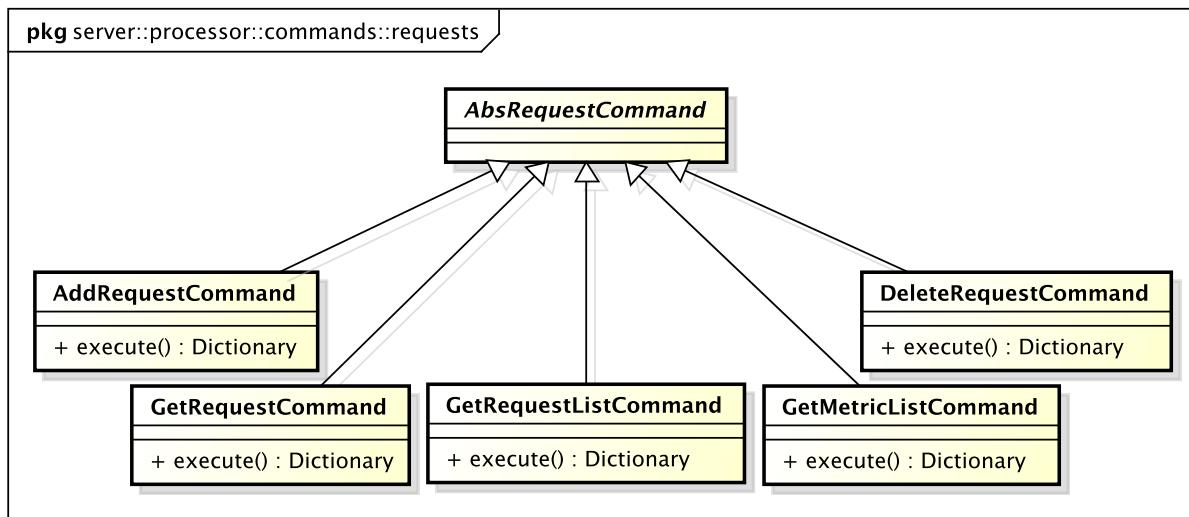


Figura 25: Package - server::processor::commands::requests

4.2.14.1 Classi

4.2.14.1.1 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

4.2.14.1.2 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

4.2.14.1.3 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

4.2.14.1.4 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

4.2.14.1.5 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

4.2.14.1.6 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

4.2.15 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

4.2.15.1 Classi

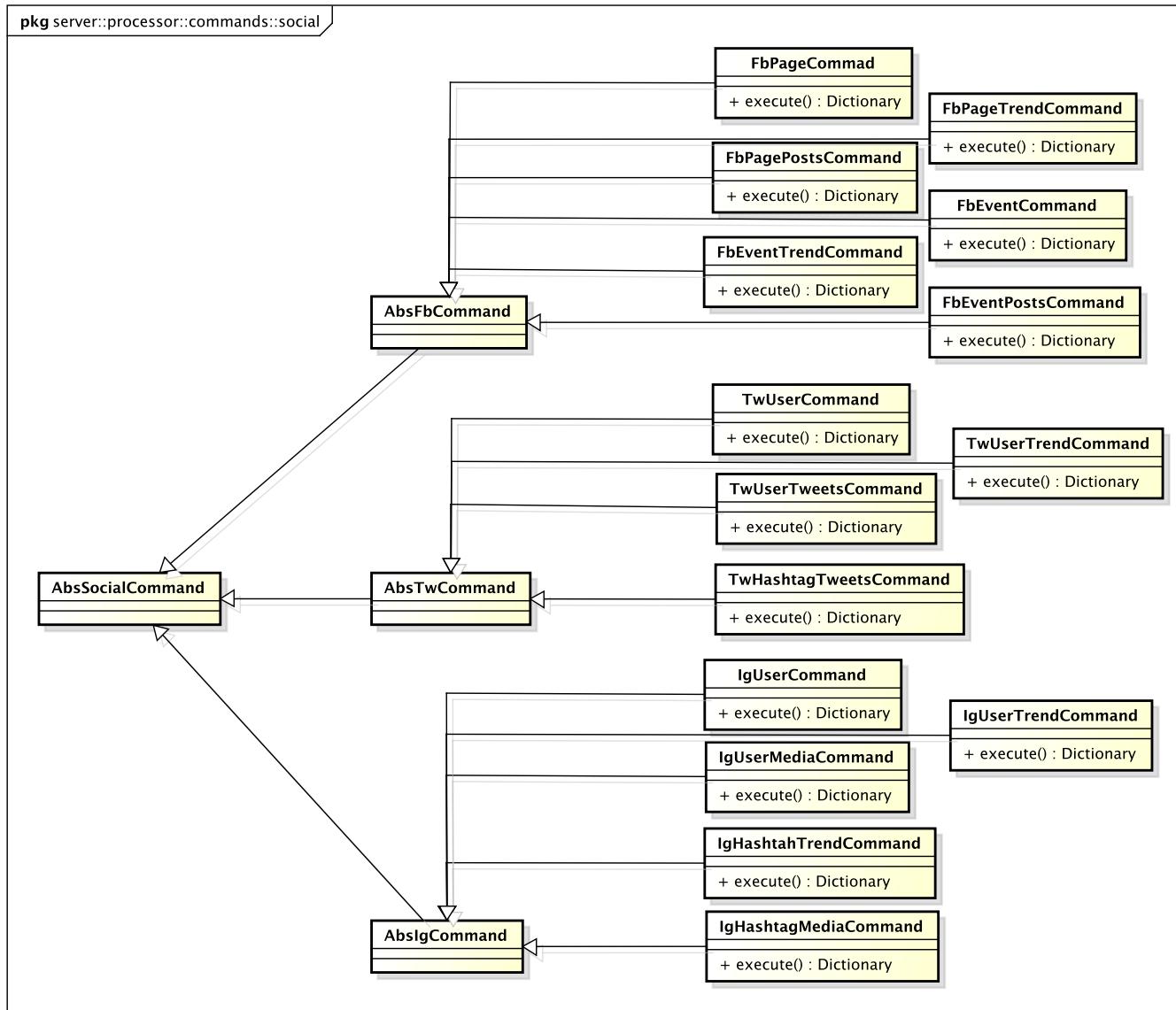


Figura 26: Package - server::processor::commands::social

4.2.15.1.1 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

4.2.15.1.2 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

4.2.15.1.3 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

4.2.15.1.4 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::processor_G::commands::social::AbsFbCommand

4.2.15.1.5 server::processor::commands::social::FbPostsCommand

- **Descrizione:** definisce la logica per ottenere i dati grezzi relativi al trend dei post di una pagina Facebook;
- **Utilizzo:** implementa il metodo `execute()` che conterrà la logica per ottenere i dati grezzi relativi al trend dei post di una pagina Facebook;
- **Relazioni con altre classi:**
 - server::processor_G::commands::social::AbsFbCommand

4.2.15.1.6 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`

4.2.15.1.7 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`

4.2.15.1.8 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`

4.2.15.1.9 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`

4.2.15.1.10 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`

4.2.15.1.11 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`

4.2.15.1.12 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`

4.2.15.1.13 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::processorG::commands::social::AbsTwCommand`

4.2.15.1.14 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::processorG::commands::social::AbsSocialCommand`

4.2.15.1.15 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::processorG::commands::social::AbsIgCommand`

4.2.15.1.16 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`

4.2.15.1.17 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`

4.2.15.1.18 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`

4.2.15.1.19 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`

4.2.16 server::miner

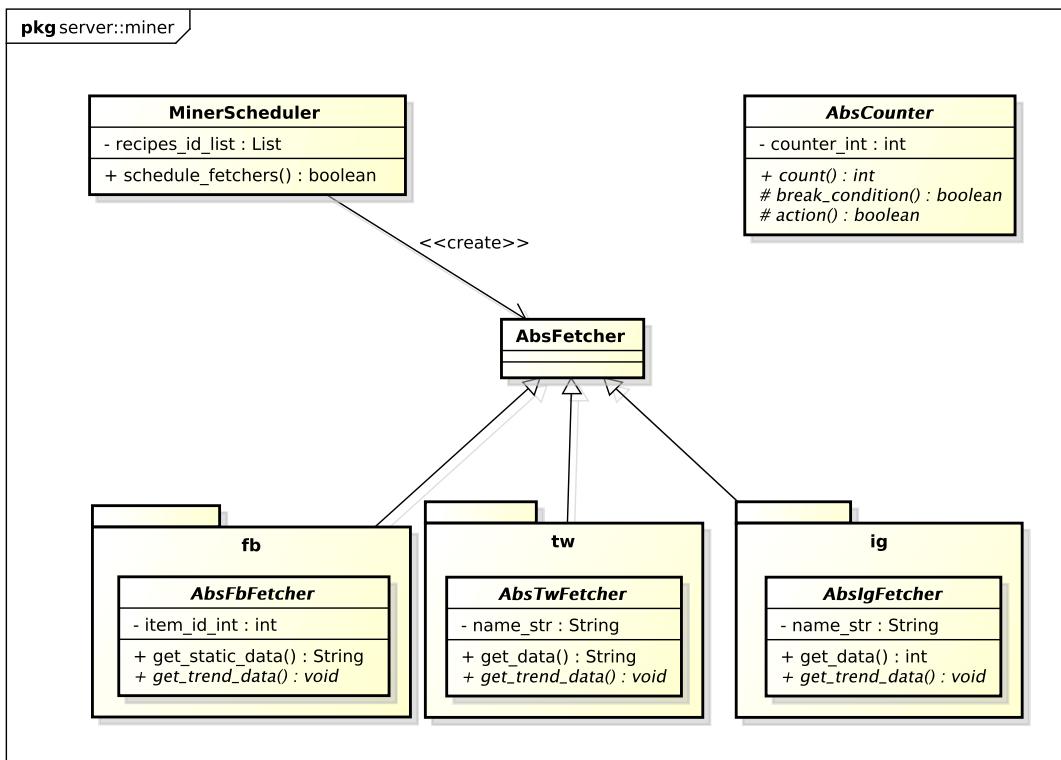


Figura 27: Package - server::miner

- **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

4.2.16.1 Classi

4.2.16.1.1 server::miner::MinerScheduler

- **Descrizione:** classe che si occupa di creare i fetcher che preleveranno i dati per ogni metrica_G di ogni social network;
- **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

4.2.16.1.2 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

4.2.16.1.3 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

4.2.17 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

4.2.17.1 Classi

4.2.17.1.1 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

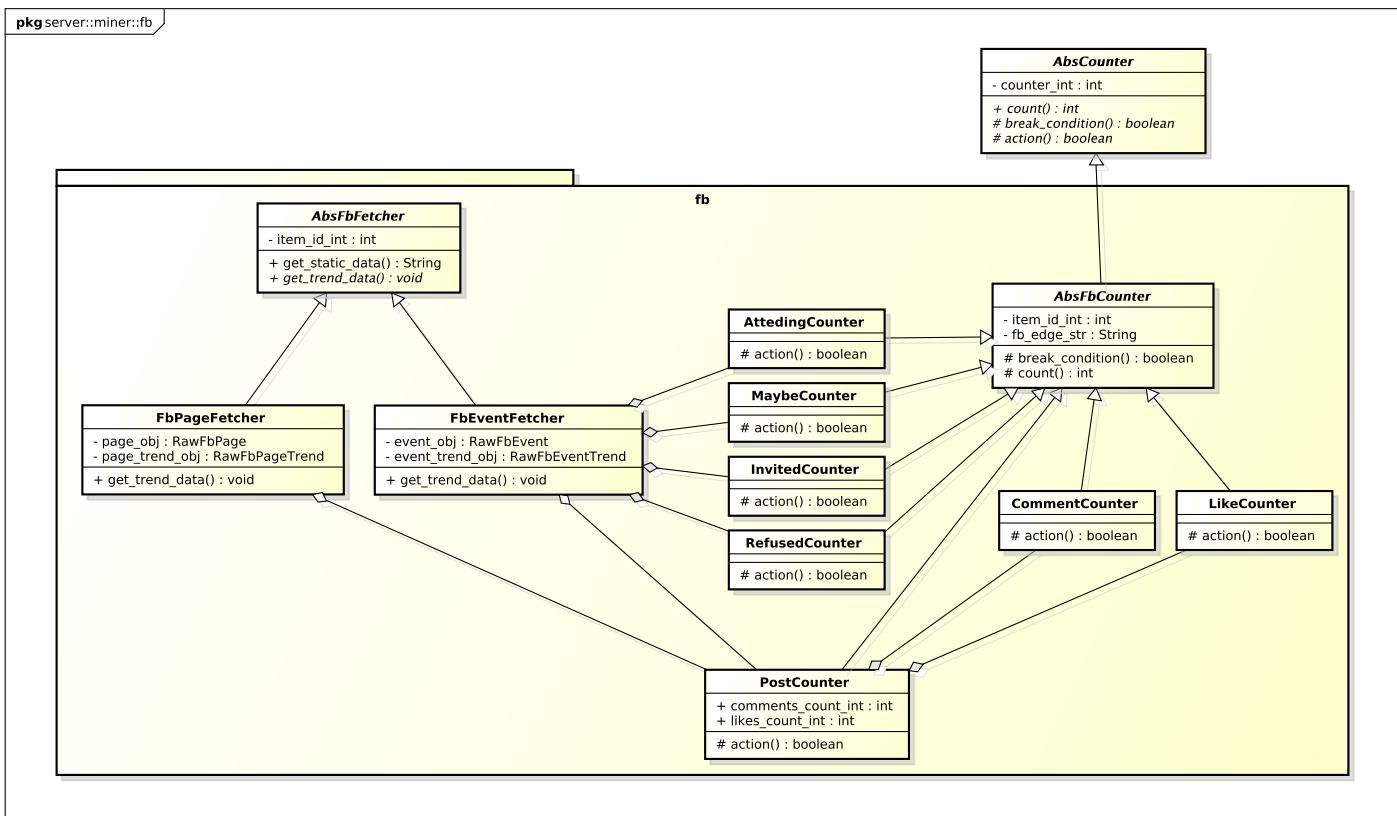


Figura 28: Package - server::miner::fb

- Relazioni con altre classi:

- server::miner_G::fb::FbPageFetcher
- server::miner_G::fb::FbEventFetcher

4.2.17.1.2 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 PostCounter;
- **Classi ereditate:** server::miner_G::fb::AbsFbFetcher
- **Relazioni con altre classi:**
 - server::miner_G::fb::PostCounter

4.2.17.1.3 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 PostCounter;
- **Classi ereditate:** server::miner_G::fb::AbsFbFetcher

- **Relazioni con altre classi:**

- server::miner_G::fb::PostCounter
- server::miner_G::fb::AttendingCounter
- server::miner_G::fb::MaybeCounter
- server::miner_G::fb::InvitedCounter
- server::miner_G::fb::RefusedCounter

4.2.17.1.4 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::AttendingCounter
- server::miner_G::fb::MaybeCounter
- server::miner_G::fb::InvitedCounter
- server::miner_G::fb::RefusedCounter
- server::miner_G::fb::PostCounter
- server::miner_G::fb::CommentCounter
- server::miner_G::fb::LikeCounter

4.2.17.1.5 server::miner::fb::AttendingCounter

- **Descrizione:** classe che descrive l'algoritmo per il counting degli invitati che non hanno ancora risposto in un evento Facebook;
- **Utilizzo:** classe che ricava il numero di persone invitate che non hanno ancora risposto di un evento Facebook;
- **Classi ereditate:** server::miner_G::fb::AbsFbCounter

4.2.17.1.6 server::miner::fb::MaybeCounter

- **Descrizione:** classe che descrive l'algoritmo per il counting dei maybe in un evento Facebook;
- **Utilizzo:** classe che ricava il numero di persone in forse per un evento Facebook;
- **Classi ereditate:** server::miner_G::fb::AbsFbCounter

4.2.17.1.7 server::miner::fb::InvitedCounter

- **Descrizione:** classe che descrive l'algoritmo per il counting degli invited in un evento Facebook;
- **Utilizzo:** classe che ricava il numero di persone invitate ad un evento Facebook;
- **Classi ereditate:** server::miner_G::fb::AbsFbCounter

4.2.17.1.8 server::miner::fb::RefusedCounter

- **Descrizione:** classe che descrive l'algoritmo per il counting dei refused in un evento Facebook;
- **Utilizzo:** classe che ricava il numero di persone che hanno rifiutato l'invito per un evento Facebook;
- **Classi ereditate:** server::miner_G::fb::AbsFbCounter

4.2.17.1.9 server::miner::fb::PostCounter

- **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::CommentCounter
 - server::miner_G::fb::LikeCounter

4.2.17.1.10 server::miner::fb::CommentCounter

- **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

4.2.17.1.11 server::miner::fb::LikeCounter

- **Descrizione:** classe che descrive l'algoritmo per calcolare il numero di like per ogni post;
- **Utilizzo:** classe che ricava il numero di like per ogni post;
- **Classe ereditate:** server::miner_G::fb::AbsFbCounter

4.2.18 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

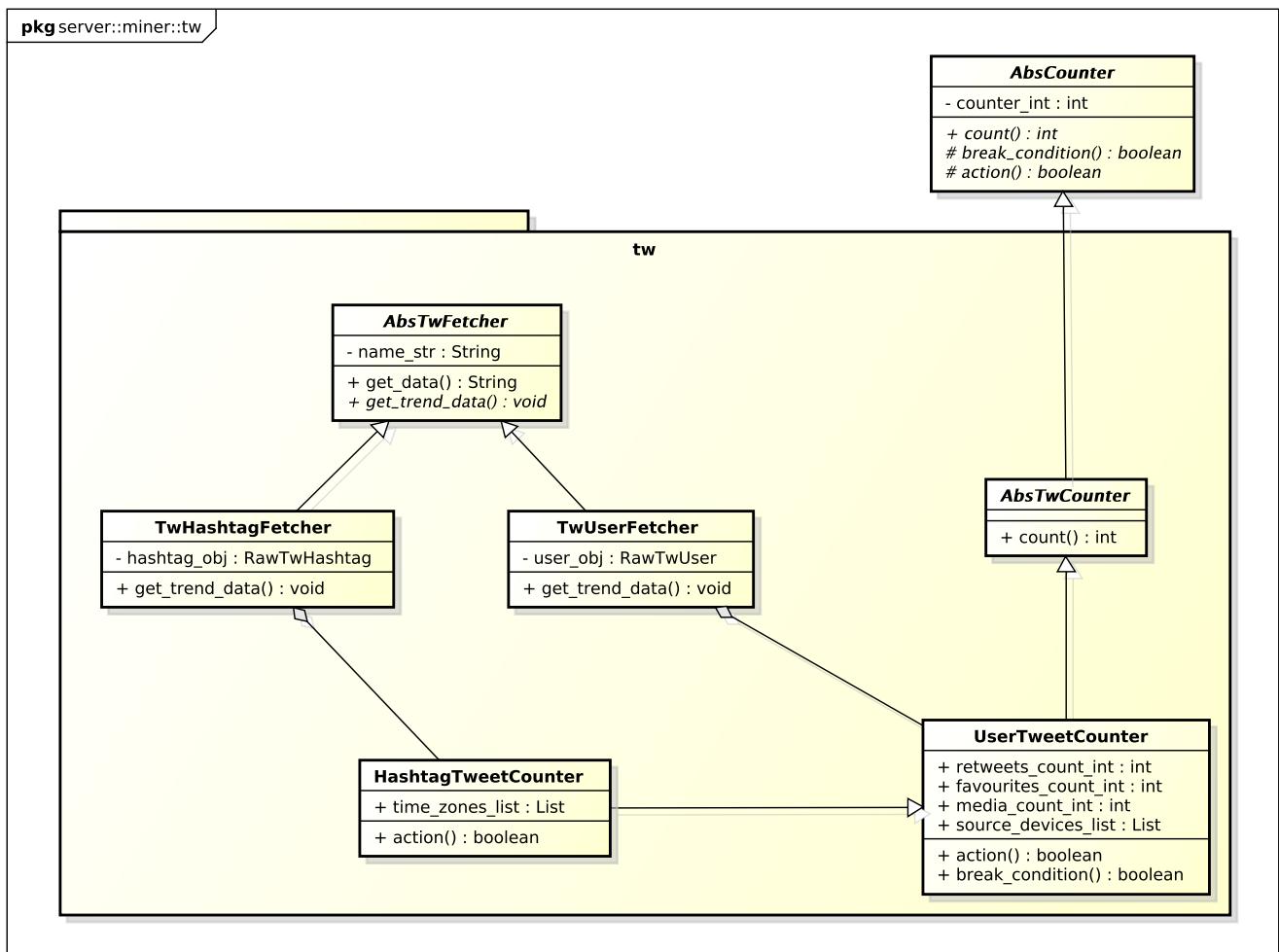


Figura 29: Package - server::miner::tw

4.2.18.1 Classi

4.2.18.1.1 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

4.2.18.1.2 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

4.2.18.1.3 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

4.2.18.1.4 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

4.2.18.1.5 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

4.2.18.1.6 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

4.2.19 server::miner::ig

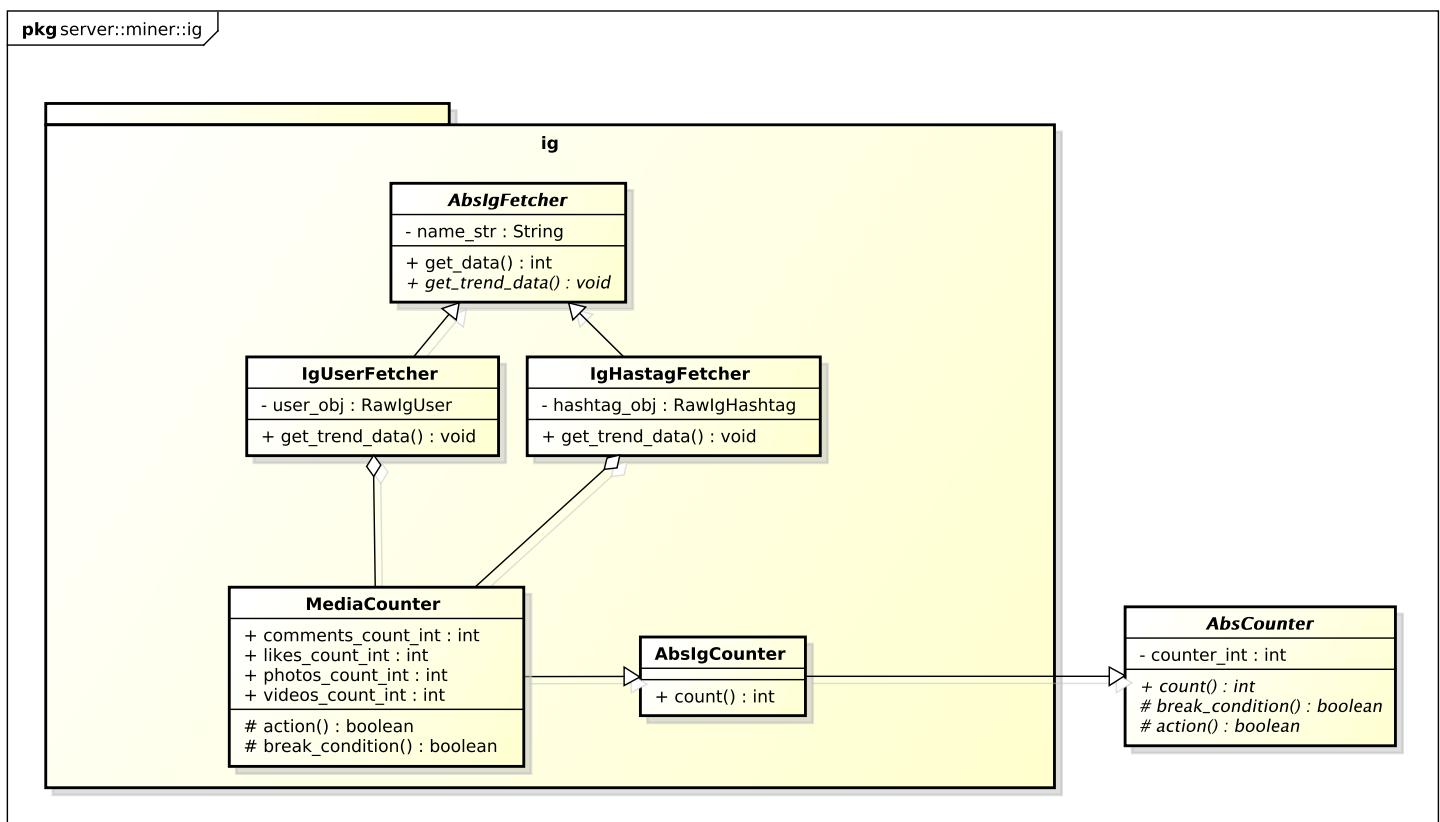


Figura 30: Package - server::miner::ig

- **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

4.2.19.1 Classi

4.2.19.1.1 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

4.2.19.1.2 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::miner_G::ig::AbsIgFetcher
- **Relazioni con altre classi:**
 - server::miner_G::ig::MediaCounter

4.2.19.1.3 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::miner_G::ig::AbsIgFetcher
- **Relazioni con altre classi:**
 - server::miner_G::ig::MediaCounter

4.2.19.1.4 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

4.2.19.1.5 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::ig::AbsIgCounter

4.2.20 server::endpoints

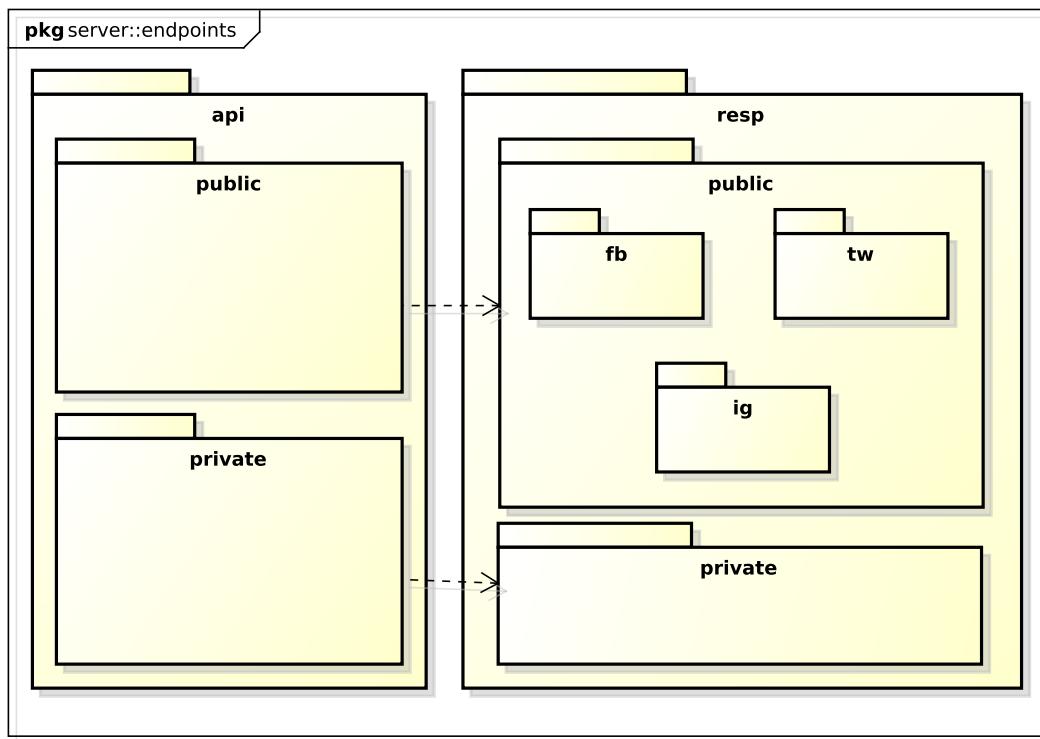


Figura 31: Package - server::endpoints

- **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

4.2.21 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 percorsi e query.

- **Descrizione:** è il package_G che definisce le web API_G offerte dall’applicazione e utilizzati dal client;
- **Padre:** server::endpoints
- **Package_G contenuti:**

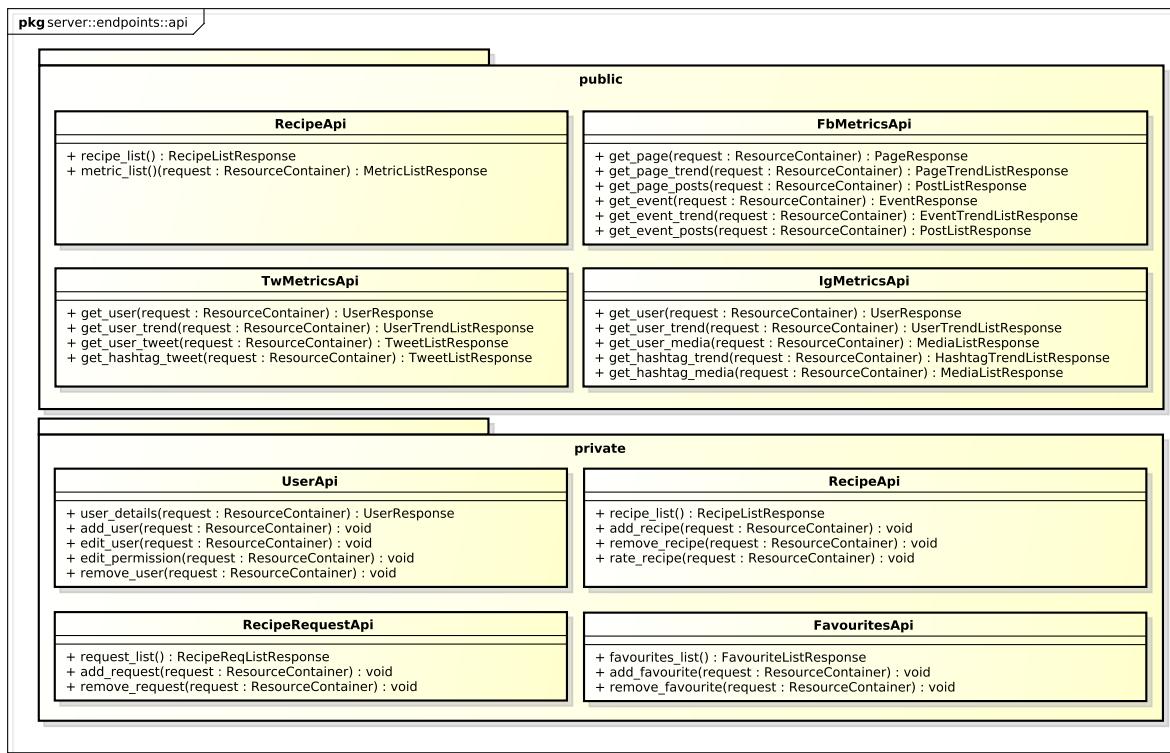


Figura 32: Package - server::endpoints::api

- 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

4.2.22 server::endpoints::api::public

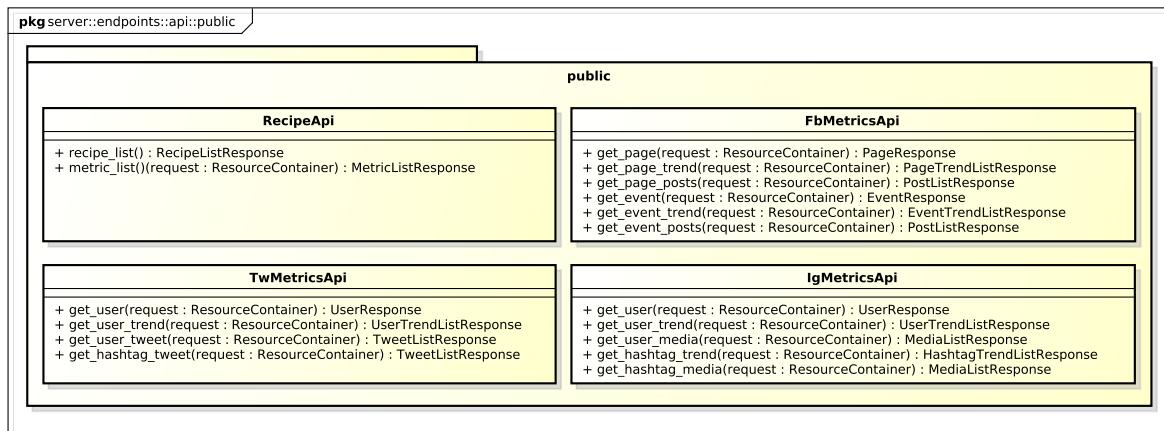


Figura 33: Package - server::endpoints::api::public

- **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

4.2.22.1 Classi

4.2.22.1.1 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::public::MetricListResponse;
 - server::endpoints::resp::private::RecipeListResponse;

4.2.22.1.2 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:**
 - 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

4.2.22.1.3 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

4.2.22.1.4 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

4.2.23 server::endpoints::api::private

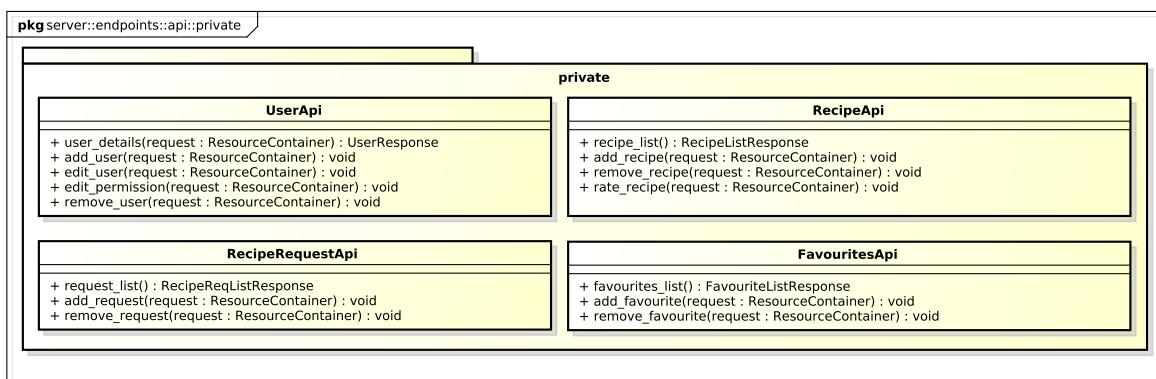


Figura 34: Package - server::endpoints::api::private

- **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

4.2.23.1 Classi

4.2.23.1.1 server::endpoints::api::private::RecipeApi

- **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

4.2.23.1.2 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 aggiungere o 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

4.2.23.1.3 server::endpoints::api::private::RecipeRequestApi

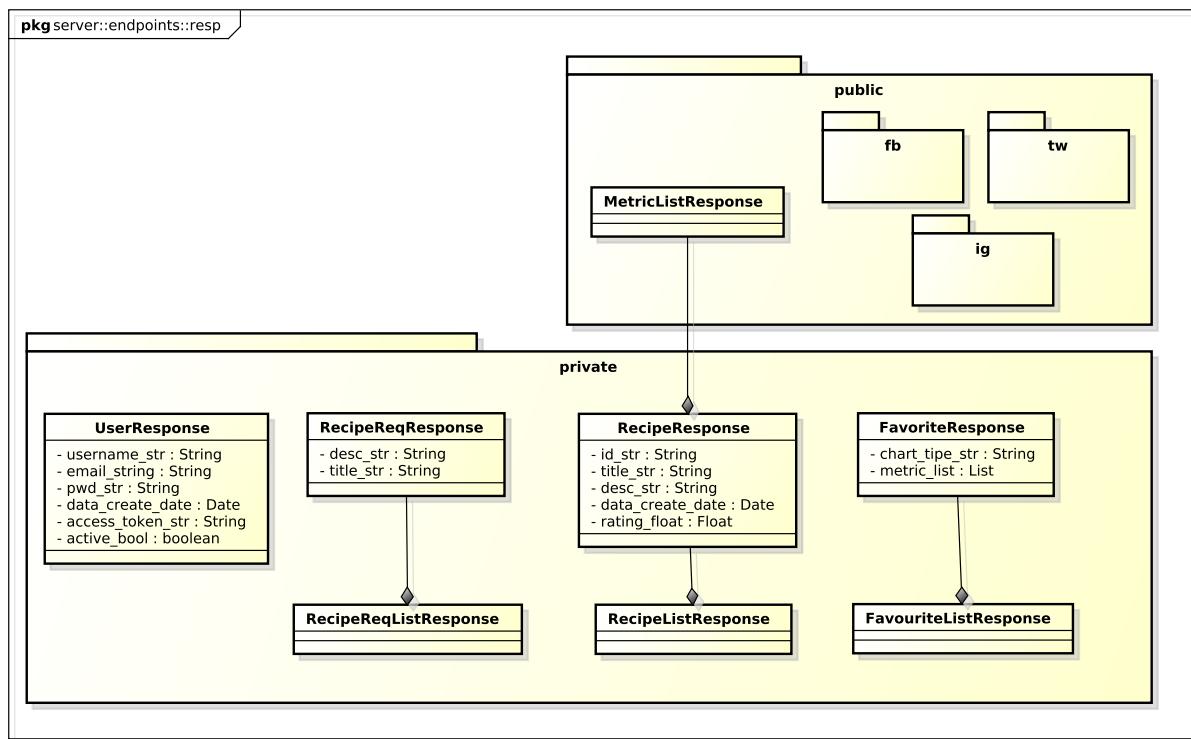
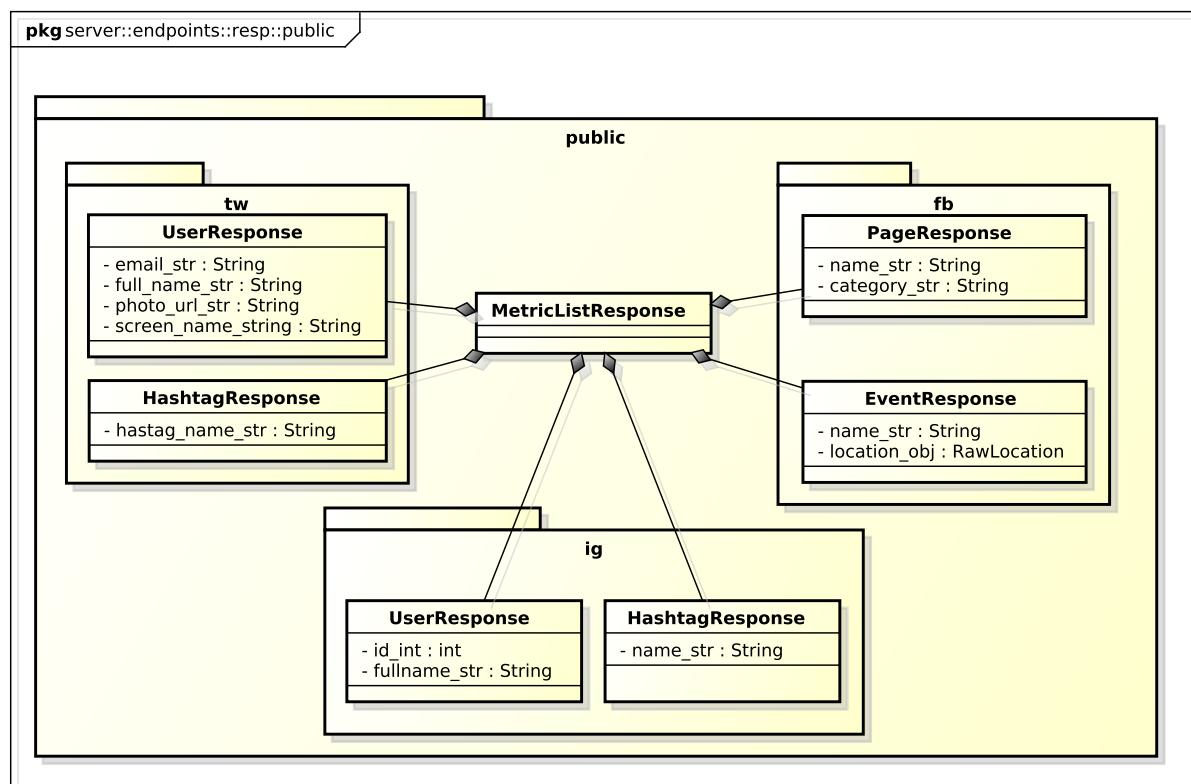
- **Descrizione:** classe che rappresenta l'implementazione delle web API_G relative alla gestione delle richieste di aggiunta Recipe_G;
- **Utilizzo:** i suoi metodi vengono invocati quando viene inviata una richiesta di nuova Recipe_G o la rimozione di una richiesta esistente da parte del client;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::RecipeReqListResponse

4.2.23.1.4 server::endpoints::api::private::FavoritesApi

- **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

4.2.24 server::endpoints::resp

- **Descrizione:** è il package_G che definisce il modelli 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


Figura 35: Package - server::endpoints::resp

Figura 36: Package - server::endpoints::resp::public

4.2.25 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:**
 - server::endpoints::resp::public::fb
 - server::endpoints::resp::public::tw
 - server::endpoints::resp::public::ig
- **Interazione con altri componenti:**
 - server::endpoints::resp::private

4.2.25.1 Classi

4.2.25.1.1 server::endpoints::resp::public::MetricListResponse

- **Descrizione:** rappresenta il modello dei dati della lista di metriche 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;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::fb::PageResponse
 - server::endpoints::resp::public::fb::EventResponse
 - server::endpoints::resp::public::tw::UserResponse
 - server::endpoints::resp::public::tw::HashtagResponse
 - server::endpoints::resp::public::ig::UserResponse
 - server::endpoints::resp::public::ig::HashtagResponse

4.2.26 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

4.2.26.1 Classi

4.2.26.1.1 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;

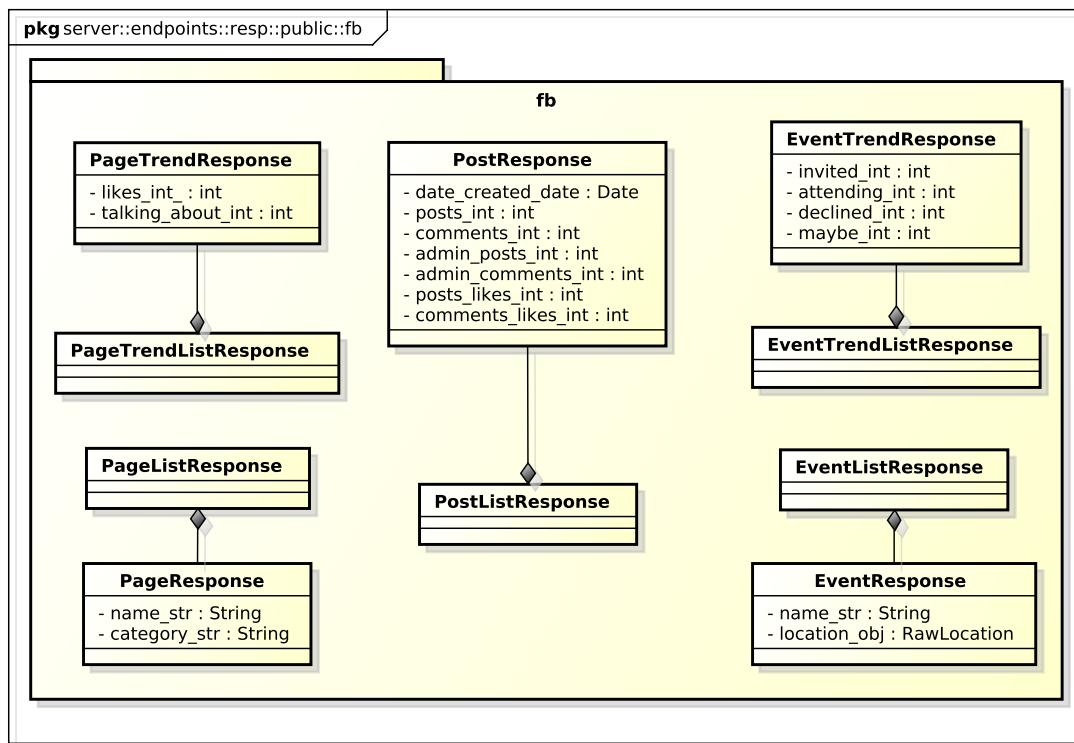


Figura 37: Package - `server::endpoints::resp::public::fb`

4.2.26.1.2 `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`

4.2.26.1.3 `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;

4.2.26.1.4 `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`

4.2.26.1.5 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;

4.2.26.1.6 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

4.2.26.1.7 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;

4.2.26.1.8 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

4.2.26.1.9 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;

4.2.26.1.10 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

4.2.27 server::endpoints::resp::public::tw

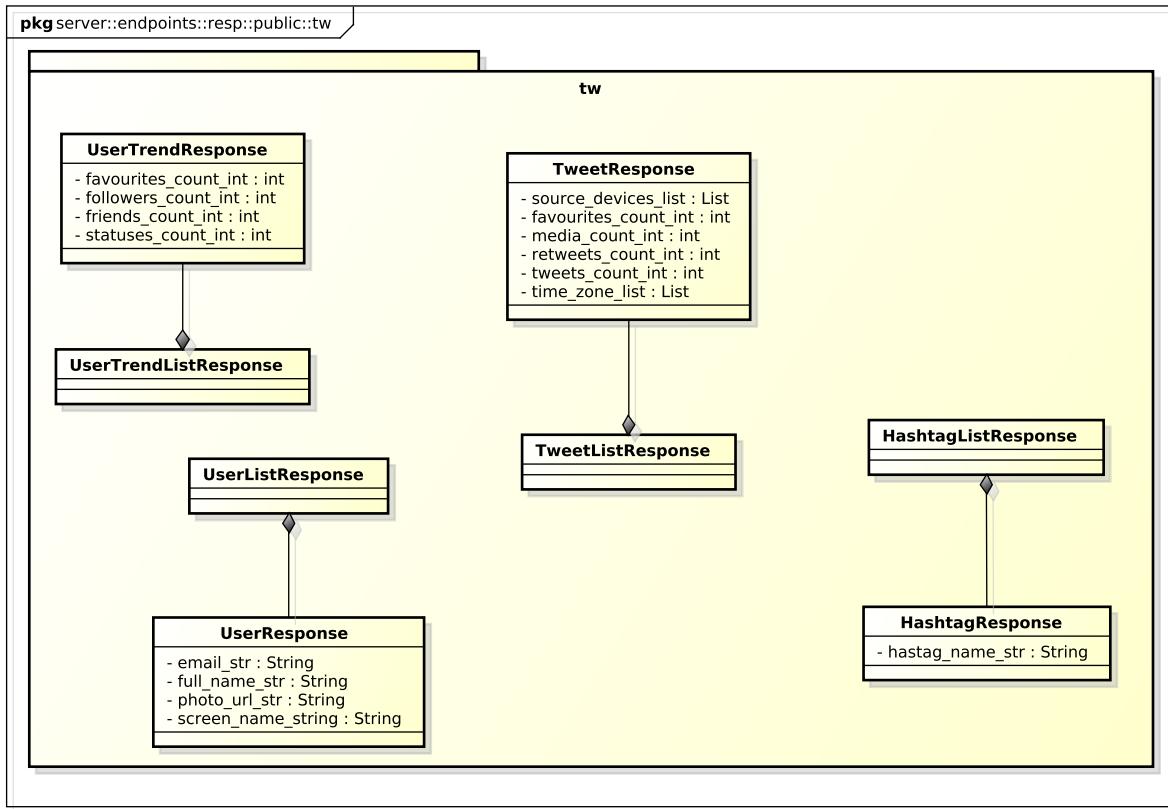


Figura 38: Package - server::endpoints::resp::public::tw

- **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

4.2.27.1 Classi

4.2.27.1.1 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;

4.2.27.1.2 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

4.2.27.1.3 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;

4.2.27.1.4 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

4.2.27.1.5 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;

4.2.27.1.6 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

4.2.27.1.7 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;

4.2.27.1.8 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

4.2.28 server::endpoints::resp::public::ig

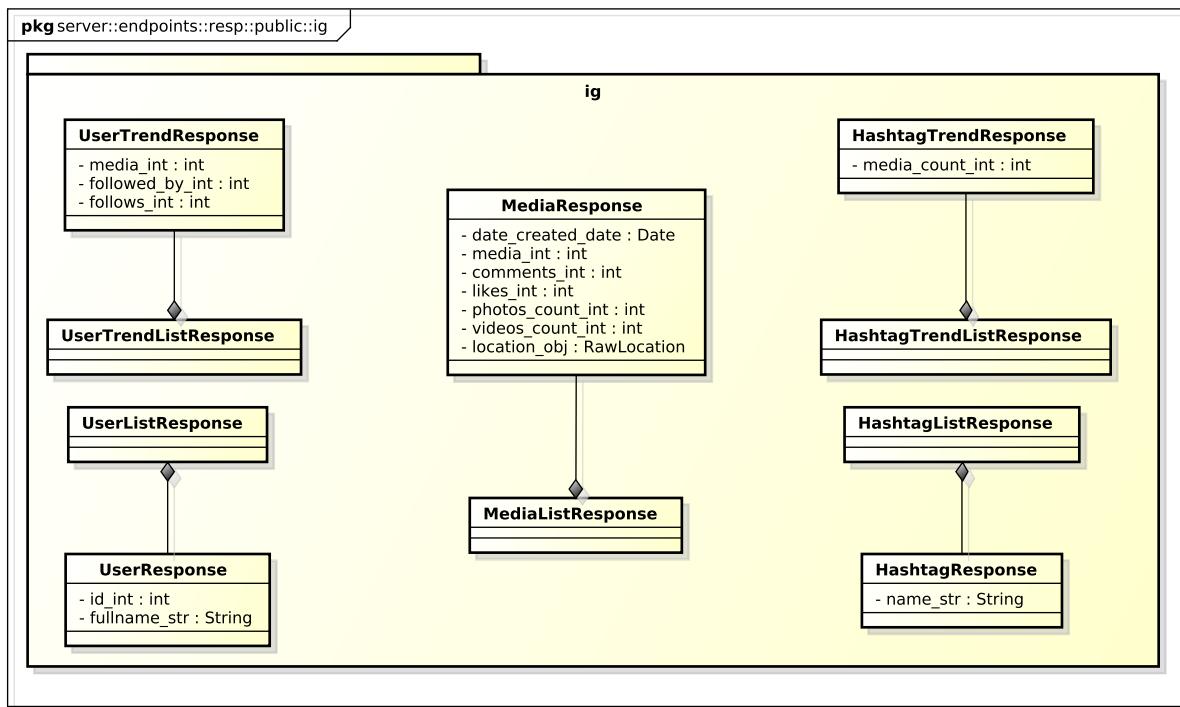


Figura 39: Package - server::endpoints::resp::public::ig

- **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

4.2.28.1 Classi

4.2.28.1.1 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;

4.2.28.1.2 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

4.2.28.1.3 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;

4.2.28.1.4 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

4.2.28.1.5 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;

4.2.28.1.6 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

4.2.28.1.7 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;

4.2.28.1.8 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

4.2.28.1.9 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;

4.2.28.1.10 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

4.2.29 server::endpoints::resp::private

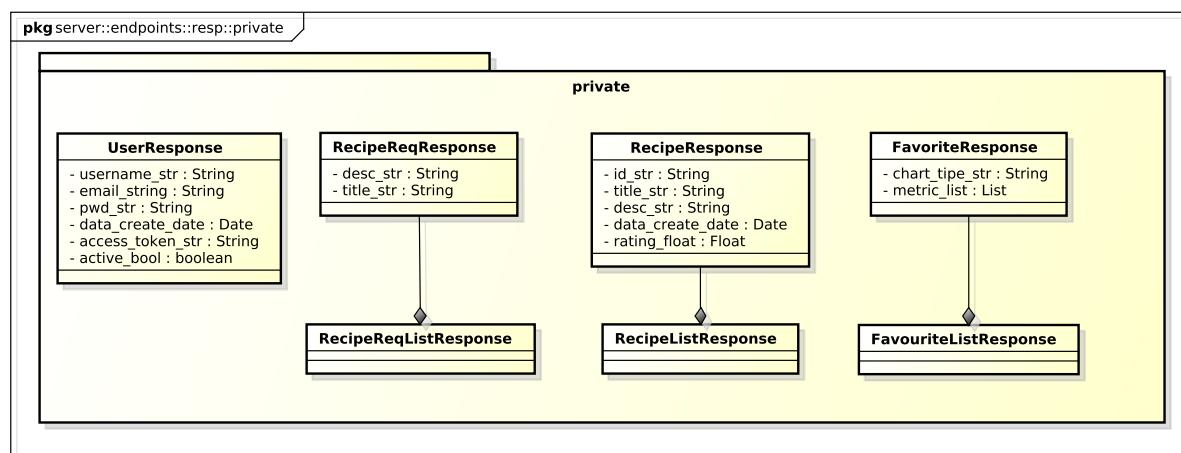


Figura 40: Package - server::endpoints::resp::private

- **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

4.2.29.1 Classi

4.2.29.1.1 server::endpoints::resp::private::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;
- **Relazioni con altre classi:**
 - server::endpoints::resp::public::MetricListResponse

4.2.29.1.2 server::endpoints::resp::private::RecipeListResponse

- **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;
- **Relazioni con altre classi:**
 - server::endpoints::resp::private::RecipeResponse

4.2.29.1.3 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;

4.2.29.1.4 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

4.2.29.1.5 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;

4.2.29.1.6 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;

4.2.29.1.7 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

4.3 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++_G'è un diagramma di come i dati siano in relazione tra loro.

Il modo quindi nel quale saranno salvati nel database, citato nella sezione 2.4, e in che formato, viene descritto dal Model dell'applicazione come illustrato alla sezione 4.2.2.

5 Interfaccia REST

Per servizi web offerti dal back-end è stato deciso di utilizzare delle API_G basate su un'architettura di tipo REST_G. Cloud Endpoints, integrato nella Google Cloud Platform, fornisce infatti un'astrazione che facilita l'integrazione di tali servizi nella piattaforma lato server e delle librerie lato client.

Le API_G RESTful, che aderiscono ai vincoli dell'architettura REST_G, utilizzano standard come HTTP per l'implementazione dei metodi, JSON_G per il formato di risposta e URI per l'identificazione della risorsa.

Di seguito sono riportati i metodi HTTP utilizzati utilizzati dei servizi RESTful della piattaforma:

Metodo	URI di una collezione, come <code>http://example.com/recipes/</code>	URI di uno specifico elemento, come <code>http://example.com/recipes/r18</code>
GET	Fornisce la lista degli elementi richiesti ed eventuali dettagli per ognuno.	Fornisce una rappresentazione dell'elemento richiesto.
PUT	Non utilizzato.	Sostituisce l'elemento richiesto o lo crea nel caso non esista ancora.
POST	Crea un nuovo elemento nella collezione.	Non utilizzato.
DELETE	Non utilizzato.	Elimina l'elemento specificato.

5.1 Lista servizi REST

Di seguito sono riportate tutte le web API_G offerte dal back-end specificando per ognuna l'URI relativo della chiamata, il metodo HTTP utilizzato ed una descrizione dell'utilizzo di tale API. Queste sono state divise a seconda dei permessi di utilizzo e, nello specifico, si divideranno in:

- pubbliche: contengono tutti i servizi utilizzabili dagli utenti registrati al sistema che hanno ottenuto un token di accesso. Sono le API_G che espongono i dati grezzi ricavati dai vari social network suddivisi per Recipe_G.
- private: contengono tutti i servizi utilizzati esclusivamente dal modulo client. Questi includono tutte le API_G mirate al funzionamento dell'applicazione.

5.1.1 API pubbliche

- */recipe_G*
 - **Tipo:** GET
 - **Utilizzo:** restituisce la lista di tutte le Recipe_G presenti nel sistema.
- */recipe_G/*{recipe-id}*/metrics*
 - **Tipo:** GET

- **Utilizzo:** restituisce la lista di tutte le metriche associate ad una Recipe_G.
- **/fb/page/{page-id}**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati di una pagina Facebook.
- **/fb/page/{page-id}/trend**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend di una pagina Facebook.
- **/fb/page/{page-id}/posts**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend dei post di una pagina Facebook.
- **/fb/event/{event-id}**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati di un evento Facebook.
- **/fb/event/{event-id}/trend**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend di un evento Facebook.
- **/fb/event/{event-id}/posts**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend dei post di un evento Facebook.
- **/tw/user/{user-id}**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati di un utente Twitter.
- **/tw/user/{user-id}/trend**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend di un utente Twitter.
- **/tw/user/{user-id}/tweets**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend dei tweet un utente Twitter.
- **/tw/user/{hashtag}/tweets**
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend dei tweet relativi ad un hashtag Twitter.

- */ig/user/{user-id}*
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati di un utente Instagram.
- */ig/user/{user-id}/trend*
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend di un utente Instagram.
- */ig/user/{user-id}/media*
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend dei media di un utente Instagram.
- */ig/hashtag/{hashtag-name}/trend*
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend di un hashtag Instagram.
- */ig/hashtag/{hashtag-name}/media*
 - **Tipo:** GET
 - **Utilizzo:** restituisce tutti i dati associati al trend dei media di un hashtag Instagram.

5.1.2 API private

- */user*
 - **Tipo:** POST
 - **Tipi:** aggiunge un utente al sistema.
- */user/{user-id}*
 - **Tipo:** GET
 - **Tipi:** ottiene le informazioni associate ad un utente.
- */user/{user-id}*
 - **Tipo:** PUT
 - **Tipi:** modifica le informazioni associate ad un utente.
- */user/{user-id}*
 - **Tipo:** DELETE
 - **Tipi:** elimina un utente dal sistema.
- */user/{user-id}/permission*
 - **Tipo:** PUT
 - **Tipi:** modifica i permessi di un utente.

- **/user/{user-id}/favourites**
 - **Tipo:** GET
 - **Tipo:** ottiene la lista delle View_G preferite di un utente.
- **/user/{user-id}/favourites**
 - **Tipo:** POST
 - **Tipo:** aggiunge una View_G tra le preferite di un utente.
- **/user/{user-id}/favourites**
 - **Tipo:** DELETE
 - **Tipo:** rimuove una View_G dalle preferite di un utente.
- **/recipe_G**
 - **Tipo:** POST
 - **Tipo:** aggiunge una nuova Recipe_G al sistema.
- **/recipe_G/{recipe-id}**
 - **Tipo:** DELETE
 - **Tipo:** rimuove una nuova Recipe_G dal sistema.
- **/recipe_G/{recipe-id}/rate**
 - **Tipo:** POST
 - **Tipo:** aggiunge un voto ad un Recipe_G.
- **/recipe_G/request**
 - **Tipo:** POST
 - **Tipo:** aggiunge una nuova richiesta di creazione Recipe_G al sistema.
- **/recipe_G/request**
 - **Tipo:** GET
 - **Tipo:** ottiene la lista delle richieste di creazione Recipe_G presenti nel sistema.
- **/recipe_G/request/{request-id}**
 - **Tipo:** GET
 - **Tipo:** ottiene i dati associati ad una richiesta di creazione Recipe_G.
- **/recipe_G/request/{request-id}**
 - **Tipo:** DELETE
 - **Tipo:** rimuove una richiesta di creazione Recipe_G dal sistema.

6 Design Pattern

In questa sezione verranno presentati i diversi design pattern_G utilizzati per la progettazione architettonale. I design pattern sono soluzioni a problemi ricorrenti. Adottarli porta diversi benefici:

- favorisce il riutilizzo del codice;
- semplifica l'attività di progettazione;
- rende l'architettura più manutenibile.

I design pattern_G possono essere suddivisi in:

- **Architetturali:** definiscono l'architettura dell'applicazione ad un livello elevato;
- **Creazionali:** permettono di nascondere i costruttori delle classi, consentendo la creazione di oggetti senza conoscerne la loro implementazione;
- **Strutturali:** consentono di riutilizzare classi preesistenti, fornendo un'interfaccia più adatta;
- **Comportamentali:** definiscono soluzioni per le interazioni tra oggetti.

Per una descrizione più approfondita dei design pattern utilizzati si faccia riferimento all'appendice B. I vari diagrammi che riprendono l'architettura non espongono tutte le sottoclassi e i metodi delle stesse. Lo scopo dei diagrammi è di mostrare le caratteristiche del design pattern adottato e come le varie classi interagiscono tra di loro. Nella realizzazione del progetto BDSMApp si è deciso di implementare i seguenti design pattern.

6.1 Design pattern architetturali

6.1.1 Three-Tier

- **Scope dell'utilizzo:** è stato scelto il pattern Three-tier per rendere massima la distribuzione delle componenti principali del sistema: client(front-end), server(back-end) e database_G (Datastore). La decisione di adottare un'architettura REST_G-like ha contribuito alla scelta dell'infrastruttura, che ci permette di separare al meglio le diverse parti;
- **Contesto dell'utilizzo:**
 - **Intera applicazione:** la decomposizione del sistema avviene secondo lo schema citato precedentemente. Ogni componente rappresenta un livello del pattern. Il client comunica con il server attraverso i servizi REST_G esposti da quest'ultimo, mentre il server comunica con il database_G attraverso le funzionalità offerte dalla Google App Engine_G.
Il diagramma che riporta lo schema del pattern è quello presente alla sezione 3.2.

6.1.2 MVW

- **Scope dell'utilizzo:** è un Design Pattern simile a MVC, che permette di avere una corrispondenza più diretta e automatica tra la *view_G* e il *model*. L'acronimo MVW sta infatti per Model-View_G-Whatever, dove *Whatever* indica “whatever works for you”;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato per gestire il lato client dell'applicazione. Questo ci viene fornito direttamente dal framework_G AngularJS_G. La parte W (Whatever) assume internamente due diversi aspetti.
Il diagramma che riporta lo schema del pattern è quello presente alla sezione 4.1.

6.1.2.1 MVC

- **Scope dell'utilizzo:** questo pattern è utilizzato per separare le responsabilità dell'applicazione a diversi componenti e permettere di fare una chiara divisione tra presentazione, struttura dei dati e operazioni su di essi;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato nel livello client per separare i componenti a seconda delle loro responsabilità e secondo una connotazione semantica. Il formalismo che viene quindi assunto per dividere i package è proprio quello fornito dal pattern in questione e cioè: **model, view e controller**. Questi però non interagiscono tra di loro nella maniera canonica che il pattern offre, ma seguono lo stile proposto da MVVM descritto alla sezione 6.1.2.2.

6.1.2.2 MVVM

- **Scope dell'utilizzo:** questo pattern è utilizzato per gestire il modo con il quale le diverse parti comunicano tra loro per scambiare i dati e per gestire le operazioni che l'utente richiede attraverso l'interazione con la View_G;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato nel livello client per gestire lo scambio dei dati e le interazioni che fanno cambiare le View_G e lo stato del Model, attraverso principalmente un sistema di two-way data binding e di eventi. Il Model però non interagisce direttamente con le View, ma passa attraverso un controller che fa da collante tra la View e il Model (ViewModel) attraverso l'oggetto \$scope.

6.1.3 Dependency Injection

- **Scope dell'utilizzo:** è un Design Pattern che viene utilizzato per favorire la separazione delle responsabilità tra i componenti dalla risoluzione delle dipendenze. Questo permette di avere una migliore modularità del codice, di avere un minor accoppiamento tra le diverse parti e garantisce una più facile fase di testing;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato direttamente da AngularJS_G per iniettare le diverse dipendenze nei moduli che le richiedono. In particolare vengono iniettati nei controller tutti i servizi e i modelli dei dati necessari all'interazione dell'utente con le viste.
Non ne viene fornita nessuna rappresentazione grafica in quanto non è una cosa che viene progettata dal team, ma usata direttamente attraverso il framework_G scelto;

6.2 Design pattern creazionali

6.2.1 Constructor Pattern

- **Scope dell'utilizzo:** questo pattern è utilizzato per emulare il costruttore tipico della programmazione ad oggetti attraverso delle funzioni che lavorano con gli oggetti;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato in tutte le classi del package_G `client::model`. Non è possibile fornirne una rappresentazione grafica in quanto questo pattern viene realizzato durante la codifica effettiva delle componenti.

6.2.2 Prototype Pattern

- **Scope dell'utilizzo:** questo pattern è utilizzato per generare il meccanismo di ereditarietà tra due classi. Viene anche scelto in modo che i metodi di una classe siano condivisi tra i diversi oggetti in quanto altrimenti, in JavaScript_G, siccome non è presente il concetto di classi si andrebbe a ripetere un metodo ogni volta che si istanzia un nuovo oggetto e questo non è ottimale;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato nelle classi del package_G `client::model` ad esempio tra `UserModel` e `UserAdminModel`. Non è possibile fornirne una rappresentazione grafica in quanto questo pattern viene realizzato durante la codifica effettiva delle componenti.

6.2.3 Module Pattern

- **Scope dell'utilizzo:** questo pattern serve per garantire, in particolare in JavaScript_G, l'incapsulamento e la privacy. È quindi utilizzato principalmente quando si vuole emulare il concetto di classe, definendo dei membri e dei metodi sia privati che pubblici;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato in tutte le classi del package_G `client::model::data` per incapsulare al meglio i membri e i metodi che i modelli dei dati usano. Non è possibile fornirne una rappresentazione grafica in quanto questo pattern viene realizzato durante la codifica effettiva delle componenti.

6.2.4 Singleton

- **Scope dell'utilizzo:** questo pattern è utilizzato per limitare l'instiazione di un certo tipo classe ad un solo oggetto in modo tale che esso rimanga unico nel sistema in cui risiede;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato direttamente da AngularJS_G quando si utilizzano i “service” o i “factory”, utilizzati per creare la logica di business del front-end nel package_G `client::model`. Il framework_G gestisce internamente questo pattern attraverso una hash map che risiede nella cache. Essa

rappresenta un singleton manager che detiene le dipendenze che vengono istanziate e le restituisce quando richieste senza crearne una nuova se questa esiste già.

Non ne viene fornita nessuna rappresentazione grafica in quanto non è una cosa che viene progettata dal team, ma usata direttamente attraverso il framework_G scelto;

- **Server:** viene utilizzato dalla classe `server::endpoints::RequestHandler` in quanto quest'ultima, essendo implementata secondo il pattern Front Controller, rappresenta il punto di accesso comune a tutto il sistema in cui confluiscono le richieste provenienti dalle classi appartenenti al package `server::endpoints` (4.2.20).

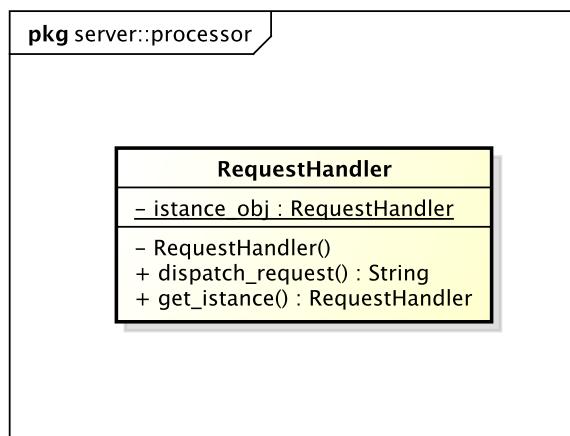


Figura 41: Contestualizzazione Singleton - Server

6.3 Design pattern strutturali

6.3.1 Adapter (object)

- **Scope dell'utilizzo:** questo pattern è utilizzato quando si vogliono utilizzare librerie esterne che però hanno interfacce diverse rispetto quelle utilizzate dall'applicazione. Per non modificare quindi parte del codice che utilizza queste componenti, le interfacce vengono adattate passando per una interna già esistente che farà da tramite con quella esterna;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato nel package `G client::model::services` per utilizzare un modulo `ng-auth` esterno che fornisce i servizi necessari per le operazioni di autenticazione, basato sui token, senza che vengano implementati dal gruppo.

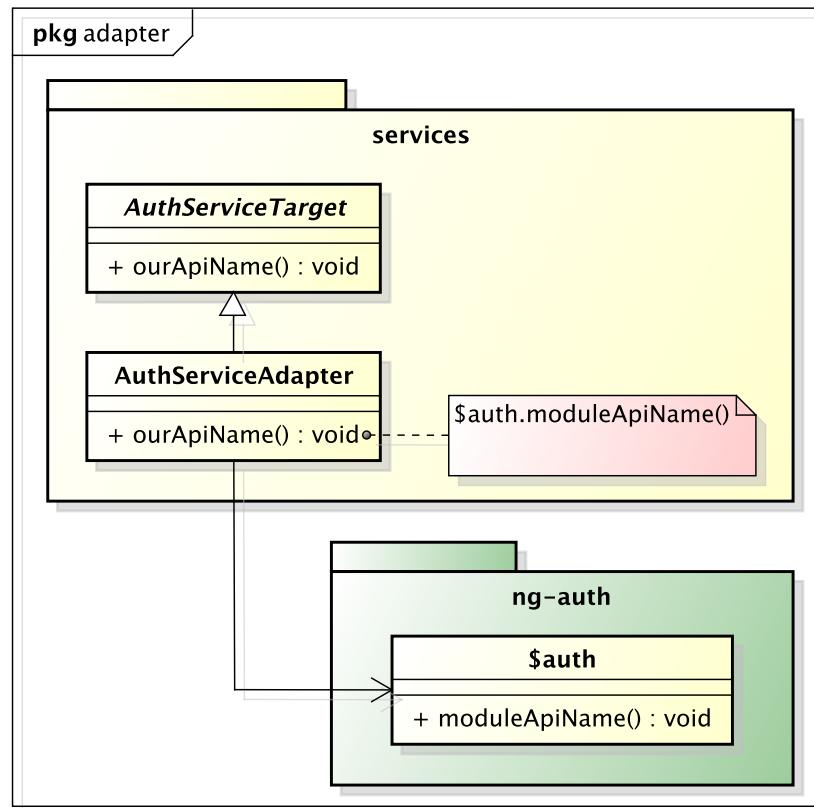


Figura 42: Contestualizzazione Adapter - Client

6.3.2 Façade

- **Scope dell'utilizzo:** questo pattern è utilizzato per fornire un'interfaccia di alto livello unificata di tante interfacce di un sotto sistema più complesso. Questo rende più semplice al client interagire con quel sistema senza preoccuparsi di come le cose vengono implementate da esso;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato direttamente da AngularJS_G in alcuni servizi come quello `$http` o `$resource` che permettono all'utilizzatore di non sapere

come viene effettivamente implementata la chiamata alle API_G, effettuandola quindi in maniera più semplice di come è realmente.

Non ne viene fornita nessuna rappresentazione grafica in quanto non è una cosa che viene progettata dal team, ma usata direttamente attraverso il framework_G scelto.

6.3.3 Front controller

- **Scope dell'utilizzo:** questo pattern è utilizzato per fornire una unica entità con lo scopo di gestire tutte le richieste in entrata da un determinato modulo dell'applicativo. Il Front Controller può inoltre effettuare semplici check o operazioni a seconda dei dati in entrata o lasciare l'elaborazione della richiesta ad una seconda entità, il Dispatcher, il quale si occupa di far confluire tale richiesta al relativo comando contenente la logica per soddisfarla. In questo modo, oltre a creare una gestione centralizzata delle richieste, si crea una divisione tra la logica di implementazione delle richieste e le modalità con cui quest'ultime vengono ricevute ed assegnate;
- **Contesto dell'utilizzo:**
 - **Server:** tale pattern è utilizzato dalle classi appartenenti al package_G `server::processor`, in cui il **Front Controller** è rappresentato dalla classe `RequestHandler` che si occupa di ricevere tutte le richieste del client, provenienti dalle classi del package `server::endpoints`. La classe `CommandDispatcher`, inoltre, si occupa di far confluire tali richieste al relativo comando.

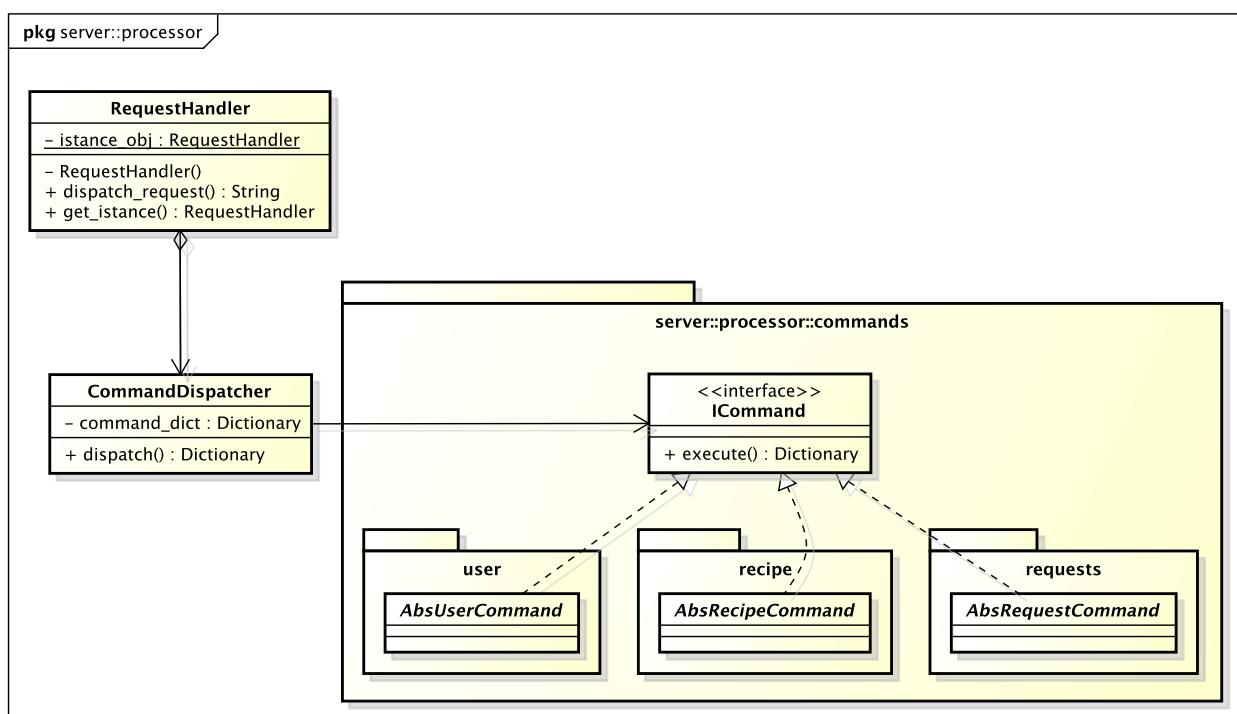


Figura 43: Contestualizzazione Front Controller - Server

6.4 Design pattern comportamentali

6.4.1 Page Controller

- **Scope dell'utilizzo:** questo pattern serve per definire un oggetto che detiene le richieste per una specifica pagina o per un'azione su un sito web. In AngularJS_G però questo componente è maggiormente limitato in quanto al controller sono affidate meno responsabilità quali l'interazione con l'utente e l'aggiornamento del model, mentre la parte delle richieste è lasciato a servizi come \$route o \$state;

- **Contesto dell'utilizzo:**

- **Client:** viene utilizzato tra tutti i template_G HTML e i controller associati ad essi. L'associazione viene definita principalmente nei file che si occupano del routing.

Non ne viene fornita alcuna illustrazione in quanto lo schema è molto simile a quello presentato per il pattern Template View, reperibile alla sezione 6.4.2, solo che il focus nel pattern non è sul template HTML, bensì sul controller che lo gestisce.

6.4.2 Template View

- **Scope dell'utilizzo:** questo pattern serve per interpretare alcune informazioni incorporate nei template_G HTML. Nei sistemi di template generalmente vengono utilizzati dei segnaposto (markers) di qualche formato che verranno interpretati e sostituiti con il codice HTML adeguato. In AngularJS_G invece non c++_G è un formato intermediario perché vengono usate direttamente delle direttive HTML che quando saranno trovate dal compilatore di Angular, verrà invocata la logica ad esse associata;

- **Contesto dell'utilizzo:**

- **Client:** viene utilizzato in tutti i template HTML presentati nel package client::view presente alla sezione 4.1.5.

Ne viene qui di seguito illustrata una implementazione relativa al template_G HTML **Settings**.

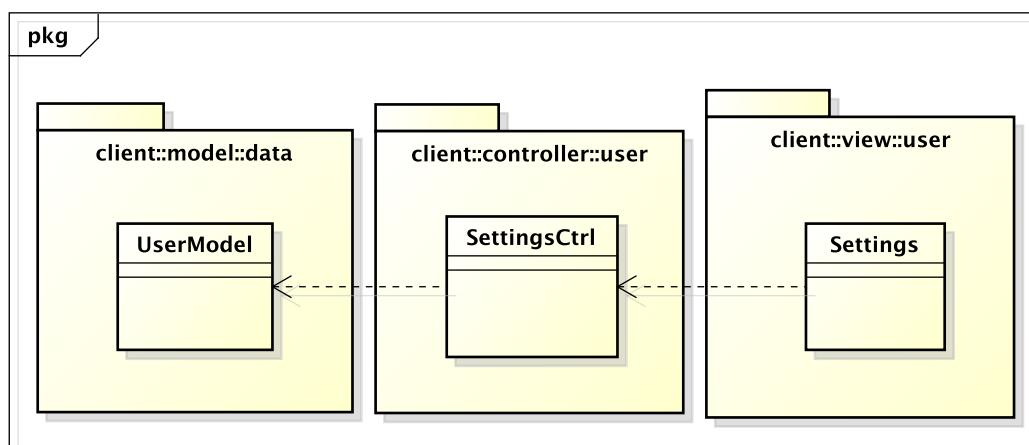


Figura 44: Contestualizzazione Template View - Client

6.4.3 Command

- **Scope dell'utilizzo:** questo pattern è utilizzato per encapsulare la logica di una richiesta in un comando totalmente indipendente dall'oggetto che lo invoca. In questo modo si crea una separazione tra logica di esecuzione della richiesta e quella di invocazione, oltre a facilitare l'estensibilità dei comandi associati;
- **Contesto dell'utilizzo:**
 - **Server:** viene utilizzato nel package `server::processor` in associazione al pattern Front Controller, il cui dispatcher si occupa di indirizzare la richiesta ricevuta al relativo comando.

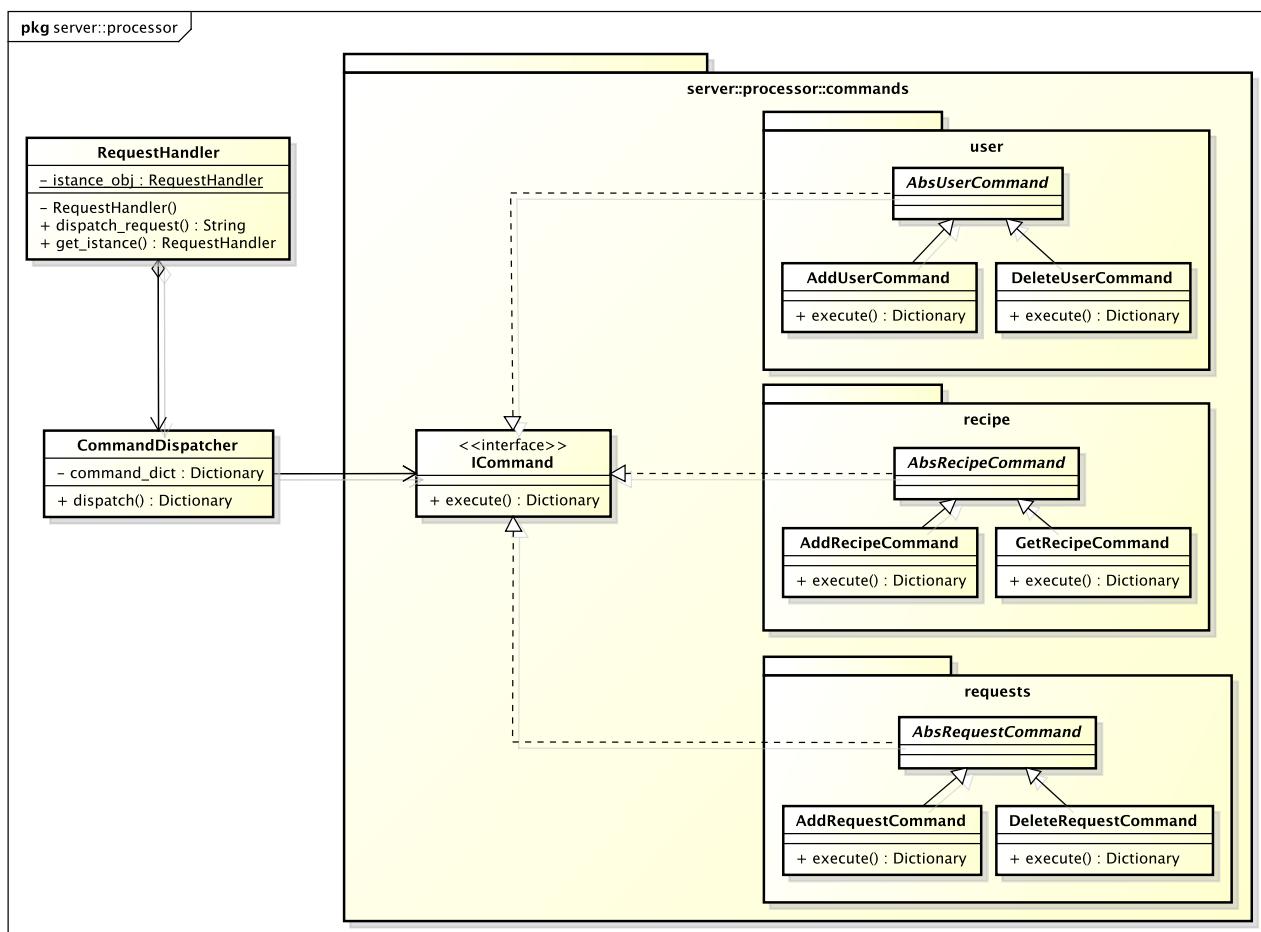


Figura 45: Contestualizzazione Command - Server

6.4.4 Template Method

- **Scope dell'utilizzo:** questo pattern serve per definire lo scheletro di un algoritmo, lasciando l'implementazione di alcuni passi alle sottoclassi;
- **Contesto dell'utilizzo:**
 - **Client:** viene utilizzato nel package 4.1.4, per permettere di generare tipi di grafici diversi che però hanno in comune la prima parte dell'algoritmo

che li genera. In particolare la parte comune si occupa del recupero dei dati a prescindere da quali grafico li userà.

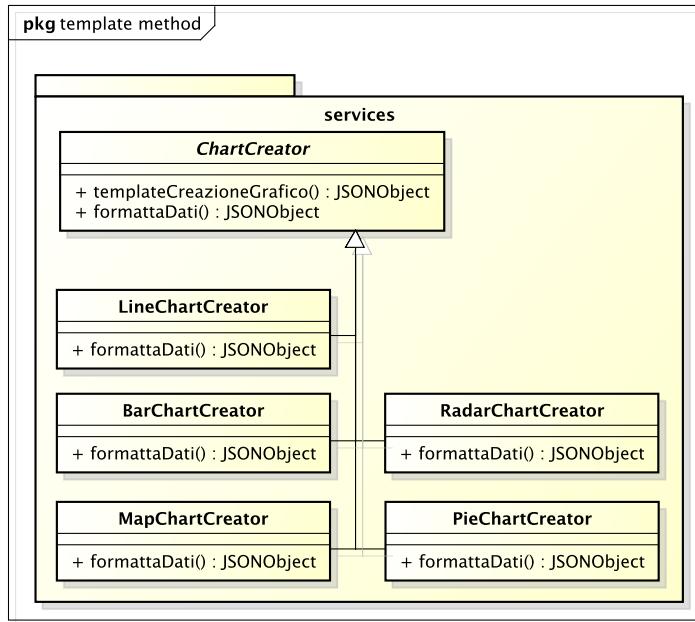


Figura 46: Contestualizzazione Template Method - Client

- **Server:** viene utilizzato nel package_G `server::minerG`, implementato dalla gerarchia di classi con `AbsCounter` come classe padre. L'algoritmo serve ad effettuare il counting di differenti entità ottenute tramite le API_G dei vari social network al fine di ottenere i dati grezzi richiesti. L'algoritmo contiene uno scheletro comune esposto dalla classe padre e viene implementato a seconda del social e del dato di cui si vuole effettuare il conteggio.

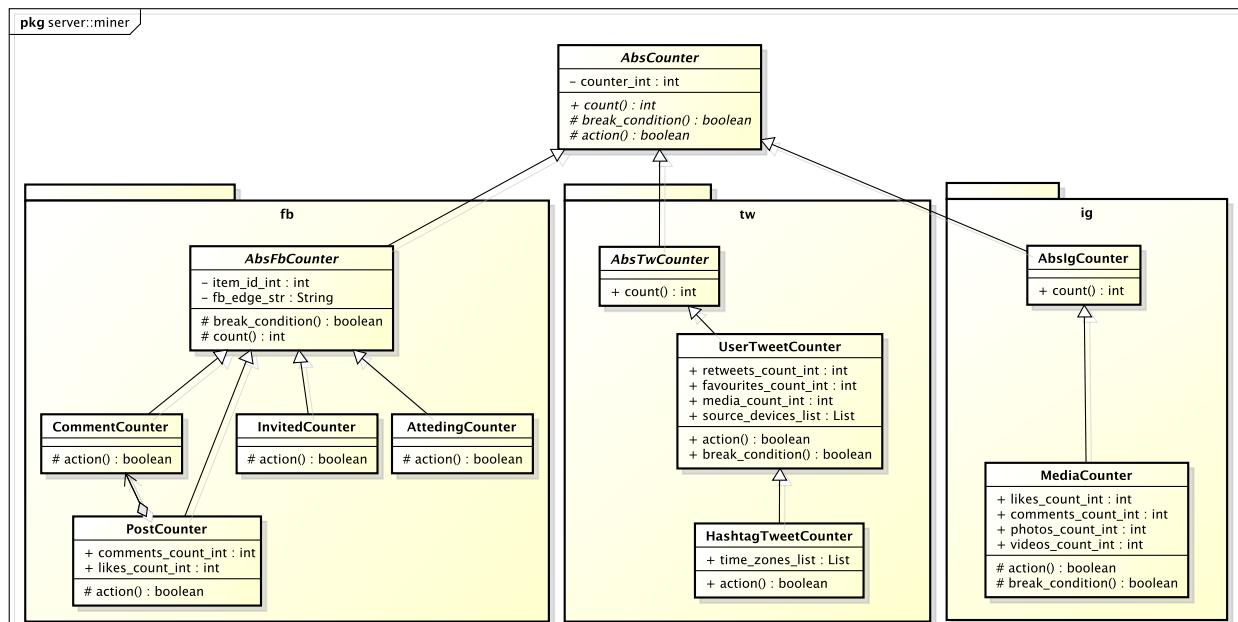


Figura 47: Contestualizzazione Template Method - Server

7 Stime di fattibilità e bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente a fornire una stima sulla fattibilità e di bisogno di risorse. L'analisi dell'architettura ha messo in evidenza che le tecnologie scelte risultano sufficientemente adeguate per la realizzazione del prodotto, anche se, per quanto riguarda la parte relativa al server, si sono trovate alcune difficoltà a progettare delle soluzioni estendibili in quanto l'utilizzo della Google App Engine con Python rende le componenti fortemente accoppiate.

La maggior parte delle tecnologie e degli strumenti sono nuove e poco conosciute per tutti i componenti del gruppo. Il team ha comunque appreso un buon livello su di esse dove avere fatto un po' di prototipazione e studiando il materiale fornito dall'*Amministratore di Progetto*. Questo permetterà al gruppo di non eccedere i tempi previsti per la realizzazione del prodotto.

8 Tracciamento

8.1 Componenti-Requisiti

Componente	Requisiti
client::model::data	ROF5.6 RFF7 RFF7.1 RFF7.2
client::model::services	ROF5.1 ROF5.2 ROF5.3 ROF5.3.1 ROF5.4 ROF5.4.1 ROF5.5 ROF5.5.1 RDF6 RDF6.1 RDF6.2 RDF6.3 RDF6.4
client::view _G	RDV4 ROV7 ROV8
client::view _G ::public	ROF1 ROF1.1 ROF1.2 ROF1.3 ROF1.4 ROF1.5 ROF1.6 ROF1.7 ROF1.8 ROF2 ROF2.1 ROF2.2 ROF2.3 ROF2.4 ROF2.4.1 ROF2.4.2 ROF2.5 ROF3.1.2 ROF3.3.6 RFF5.1.3 ROF12 ROF12.1 ROF12.2

client::view _G ::user	ROF3 ROF3.1 ROF3.2 ROF3.3 ROF4 ROF4.1 ROF5 ROF5.1 ROF5.2 ROF5.3 ROF5.4 ROF5.5 ROF5.6 ROF5.6.1 ROF5.6.2 ROF6 ROF6.1 ROF6.2 ROF6.3 ROF6.4 RFF7 RFF7.1
client::view _G ::admin	ROF8 ROF8.1 ROF8.2 ROF8.3 RFF8.4 ROF9 ROF9.1 ROF9.2 ROF9.3 RFF10 RFF10.1 RFF10.2 RFF10.3 RFF10.4 RFF10.5
client::controller	

client::controller::public	ROF1 ROF1.1 ROF1.2 ROF1.3 ROF1.4 ROF1.5 ROF1.6 ROF1.7 ROF1.8 ROF2 ROF2.1 ROF2.2 ROF2.3 ROF2.4 ROF2.5 ROF12 ROF12.1 ROF12.2
client::controller::user	ROF3 ROF3.1 ROF3.2 ROF3.3 ROF4 ROF4.1 ROF5 ROF5.1 ROF5.2 ROF5.3 ROF5.4 ROF5.5 ROF5.6 ROF6 ROF6.1 ROF6.2 ROF6.3 ROF6.4 RFF7 RFF7.1

client::controller::admin	ROF8 ROF8.1 ROF8.2 ROF8.3 RFF8.4 ROF9 ROF9.1 ROF9.2 ROF9.3 RFF10 RFF10.1 RFF10.2 RFF10.3 RFF10.4 RFF10.5
server::db	RDV1.2
server::db::raw_data	ROF5.2.1.1
server::db::raw_data::fb	ROF5.3.1.1 ROF5.3.1.2 ROF5.3.1.3 ROF5.3.1.4 ROF5.3.1.5 ROF5.3.1.6
server::db::raw_data::tw	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
server::db::raw_data::ig	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
server::db::app_data	

server::db::app_data::user	ROF1.1 ROF1.1.2 ROF1.2 ROF1.3 ROF1.4 ROF2.1 ROF2.2 ROF3.1 ROF3.3 ROF3.3.7 ROF6 ROF8.1 ROF9 ROF11.2
server::db::app_data::recipe _G	RFF3.2.3 ROF5 RFF7 ROF8.1 ROF8.2 ROF8.3 RFF8.4 RFF10 ROF11.3.1.1 ROF11.3.1.2
server::endpoints	RDV2 RDV1.4
server::endpoints::api _G	ROF11 ROF11.1 ROF11.2 ROF11.3
server::endpoints::api _G ::public	ROF11.3.1.1 ROF11.3.1.2 ROF11.3.1.3 ROF5.2.1 ROF5.2.2 ROF5.6.2 ROF5.6.3 ROF5.6.4 RFF7.1.1.3 RFF7.1.1.5 RFF7.1.1.6 RFF7.1.1.7 ROF8.1.1.3 ROF8.1.1.4 RFF3.2.3 ROF5

server::endpoints::api _G ::private	ROF1.1 ROF1.1.2 ROF1.2 ROF1.3 ROF1.4 ROF2.1 ROF2.2 ROF3.1 ROF3.3 ROF3.3.7 RDF6 ROF8.1 ROF9 ROF11.1 ROF11.2 RFF7 RFF10
server::endpoints::resp	RDV1.4
server::endpoints::resp::public	RDV1.4
server::endpoints::resp::public::fb	RDV1.4
server::endpoints::resp::public::tw	RDV1.4
server::endpoints::resp::public::ig	RDV1.4
server::endpoints::resp::private	RDV1.4
server::miner _G	RDV2 ROV6
server::miner _G ::fb	ROV6.1
server::miner _G ::tw	ROV6.2
server::miner _G ::ig	ROV6.3
server::processor _G	RDV2
server::processor _G ::commands	

server::processor _G ::commands::user	ROF1.1.1 ROF1.1.3 ROF1.3.1 ROF1.3.2 ROF1.3.3 ROF1.4.1 ROF1.1 ROF1.1.2 ROF1.2 ROF1.3 ROF1.4 RDF1.8 ROF2.1 ROF2.2 ROF3.1 ROF3.1.1 ROF3.1.2 ROF3.1.3 RFF3.2.2 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 ROF3.3.7 RDF6 RDF6.1 RDF6.2 RDF6.3
---	--

server::processor _G ::commands::recipe _G	RFF3.2.3 ROF5 RFF7 ROF8.1 ROF8.2 ROF8.3 RFF8.4 RFF10 ROF11.3.1.1 ROF11.3.1.2
server::processor _G ::commands::requests	RFF7 RFF10
server::processor _G ::commands::social	ROF11.3.1.2 ROF5.2.1 ROF5.2.2 ROF5.6.3 ROF5.3 ROF5.4 ROF5.5

8.2 Requisiti-Componenti

Requisito	Componenti
ROF1	client::view _G ::public client::controller::public
ROF1.1	client::view _G ::public client::controller::public server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF1.1.1	server::processor _G ::commands::user
ROF1.1.2	server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF1.1.3	server::processor _G ::commands::user
ROF1.2	client::view _G ::public client::controller::public server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user

ROF1.3	client::view _G ::public client::controller::public server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF1.3.1	server::processor _G ::commands::user
ROF1.3.2	server::processor _G ::commands::user
ROF1.3.3	server::processor _G ::commands::user
ROF1.4	client::view _G ::public client::controller::public server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF1.4.1	server::processor _G ::commands::user
ROF1.6	client::view _G ::public client::controller::public
ROF1.7	client::view _G ::public client::controller::public
RDF1.8	server::processor _G ::commands::user
ROF2	client::view _G ::public client::controller::public
ROF2.1	client::view _G ::public client::controller::public server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF2.2	client::view _G ::public client::controller::public server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF2.3	client::view _G ::public client::controller::public
ROF2.4	client::view _G ::public client::controller::public
ROF2.4.1	client::view _G ::public
ROF2.4.2	client::view _G ::public
ROF2.5	client::view _G ::public client::controller::public

ROF3	client::view _G ::user client::controller::user
ROF3.1	client::view _G ::user client::controller::user server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF3.1.1	server::processor _G ::commands::user
ROF3.1.2	client::view _G ::public server::processor _G ::commands::user
ROF3.1.3	server::processor _G ::commands::user
RFF3.2.2	server::processor _G ::commands::user
RFF3.2.3	server::db::app_data::recipe _G server::endpoints::api _G ::public server::processor _G ::commands::recipe
ROF3.3	client::view _G ::user client::controller::user server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF3.3.1	server::processor _G ::commands::user
ROF3.3.2	server::processor _G ::commands::user
ROF3.3.2.1	server::processor _G ::commands::user
ROF3.3.2.2	server::processor _G ::commands::user
ROF3.3.3	server::processor _G ::commands::user
ROF3.3.3.1	server::processor _G ::commands::user
ROF3.3.3.2	server::processor _G ::commands::user
ROF3.3.3.3	server::processor _G ::commands::user
ROF3.3.4	server::processor _G ::commands::user
ROF3.3.4.1	server::processor _G ::commands::user
ROF3.3.4.2	server::processor _G ::commands::user
ROF3.3.4.2.1	server::processor _G ::commands::user
ROF3.3.4.3	server::processor _G ::commands::user
ROF3.3.4.3.1	server::processor _G ::commands::user
ROF3.3.6	client::view _G ::public

ROF3.3.7	server::db::app_data::user server::endpoints::api _G ::private server::processor _G ::commands::user
ROF4	client::view _G ::user client::controller::user
ROF4.1	client::view _G ::user client::controller::user
ROF5	client::view _G ::user client::controller::user server::db::app_data::recipe _G server::endpoints::api _G ::public server::processor _G ::commands::recipe
ROF5.1	client::model::services client::view _G ::user client::controller::user
RFF5.1.3	client::view _G ::public
ROF5.2	client::model::services client::view _G ::user client::controller::user
ROF5.2.1	server::endpoints::api _G ::public server::processor _G ::commands::social
ROF5.2.1.1	server::db::raw_data
ROF5.2.2	server::endpoints::api _G ::public server::processor _G ::commands::social
ROF5.3	client::model::services client::view _G ::user client::controller::user server::processor _G ::commands::social
ROF5.3.1	client::model::services
ROF5.3.1.1	server::db::raw_data::fb
ROF5.3.1.2	server::db::raw_data::fb
ROF5.3.1.3	server::db::raw_data::fb
ROF5.3.1.4	server::db::raw_data::fb
ROF5.3.1.5	server::db::raw_data::fb
ROF5.3.1.6	server::db::raw_data::fb
ROF5.4	client::view _G ::user client::controller::user server::processor _G ::commands::social
ROF5.4.1.1	server::db::raw_data::tw

ROF5.4.1.2	server::db::raw_data::tw
ROF5.4.1.3	server::db::raw_data::tw
ROF5.4.1.4	server::db::raw_data::tw
ROF5.4.1.5	server::db::raw_data::tw
ROF5.4.1.6	server::db::raw_data::tw
ROF5.4.1.7	server::db::raw_data::tw
ROF5.5	client::model::services client::view _G ::user client::controller::user server::processor _G ::commands::social
ROF5.5.1	client::model::services
ROF5.5.1.1	server::db::raw_data::ig
ROF5.5.1.2	server::db::raw_data::ig
ROF5.5.1.3	server::db::raw_data::ig
ROF5.5.1.4	server::db::raw_data::ig
ROF5.5.1.5	server::db::raw_data::ig
ROF5.5.1.6	server::db::raw_data::ig
ROF5.5.1.7	server::db::raw_data::ig
ROF5.5.1.8	server::db::raw_data::ig
ROF5.5.1.9	server::db::raw_data::ig
ROF5.5.1.10	server::db::raw_data::ig
ROF5.6	client::model::data client::view _G ::user client::controller::user
ROF5.6.1	client::view _G ::user
ROF5.6.2	client::view _G ::user server::endpoints::api _G ::public
ROF5.6.3	server::endpoints::api _G ::public server::processor _G ::commands::social
ROF5.6.4	server::endpoints::api _G ::public
RDF6	client::model::services server::endpoints::api _G ::private server::processor _G ::commands::user
RDF6.1	client::model::services server::processor _G ::commands::user

RDF6.2	client::model::services server::processor _G ::commands::user
RDF6.3	client::model::services server::processor _G ::commands::user
RDF6.4	client::model::services
RFF7	client::model::data client::view _G ::user client::controller::user server::db::app_data::recipe _G server::endpoints::api _G ::private server::processor _G ::commands::recipe server::processor::commands::requests
RFF7.1	client::model::data client::view _G ::user client::controller::user
RFF7.1.1.3	server::endpoints::api _G ::public
RFF7.1.1.5	server::endpoints::api _G ::public
RFF7.1.1.6	server::endpoints::api _G ::public
RFF7.1.1.7	server::endpoints::api _G ::public
RFF7.2	client::model::data
ROF8	client::view _G ::admin client::controller::admin
ROF8.1	client::view _G ::admin client::controller::admin server::db::app_data::user server::db::app_data::recipe _G server::endpoints::api _G ::private server::processor _G ::commands::recipe
ROF8.1.1.3	server::endpoints::api _G ::public
ROF8.1.1.4	server::endpoints::api _G ::public
ROF8.2	client::view _G ::admin client::controller::admin server::db::app_data::recipe _G server::processor _G ::commands::recipe
ROF8.3	client::view _G ::admin client::controller::admin server::db::app_data::recipe _G server::processor _G ::commands::recipe

RFF8.4	client::view _G ::admin client::controller::admin server::db::app_data::recipe _G server::processor _G ::commands::recipe
ROF9	client::view _G ::admin client::controller::admin server::db::app_data::user server::endpoints::api _G ::private
ROF9.1	client::view _G ::admin client::controller::admin
ROF9.2	client::view _G ::admin client::controller::admin
ROF9.3	client::view _G ::admin client::controller::admin
RFF10	client::view _G ::admin client::controller::admin server::db::app_data::recipe _G server::endpoints::api _G ::private server::processor _G ::commands::recipe server::processor::commands::requests
RFF10.1	client::view _G ::admin client::controller::admin
RFF10.2	client::view _G ::admin client::controller::admin
RFF10.3	client::view _G ::admin client::controller::admin
RFF10.4	client::view _G ::admin client::controller::admin
RFF10.5	client::view _G ::admin client::controller::admin
ROF11	server::endpoints::api _G
ROF11.1	server::endpoints::api _G server::endpoints::api::private
ROF11.2	server::db::app_data::user server::endpoints::api _G server::endpoints::api::private
ROF11.3	server::endpoints::api _G
ROF11.3.1.1	server::db::app_data::recipe _G server::endpoints::api _G ::public server::processor _G ::commands::recipe

ROF11.3.1.2	server::db::app_data::recipe _G server::endpoints::api _G ::public server::processor _G ::commands::recipe server::processor::commands::social
ROF11.3.1.3	server::endpoints::api _G ::public
ROF12	client::view _G ::public client::controller::public client::controller::public
ROF12.1	client::view _G ::public
ROF12.2	client::view _G ::public
RDV1.2	server::db
RDV1.4	server::endpoints server::endpoints::resp server::endpoints::resp::public server::endpoints::resp::public::fb server::endpoints::resp::public::tw server::endpoints::resp::public::ig server::endpoints::resp::private
RDV2	server::endpoints server::miner _G server::processor _G
RDV4	client::view _G
ROV6	server::miner _G
ROV6.1	server::miner _G ::fb
ROV6.2	server::miner _G ::tw
ROV6.3	server::miner _G ::ig
ROV7	client::view _G
ROV8	client::view _G

A Mockup

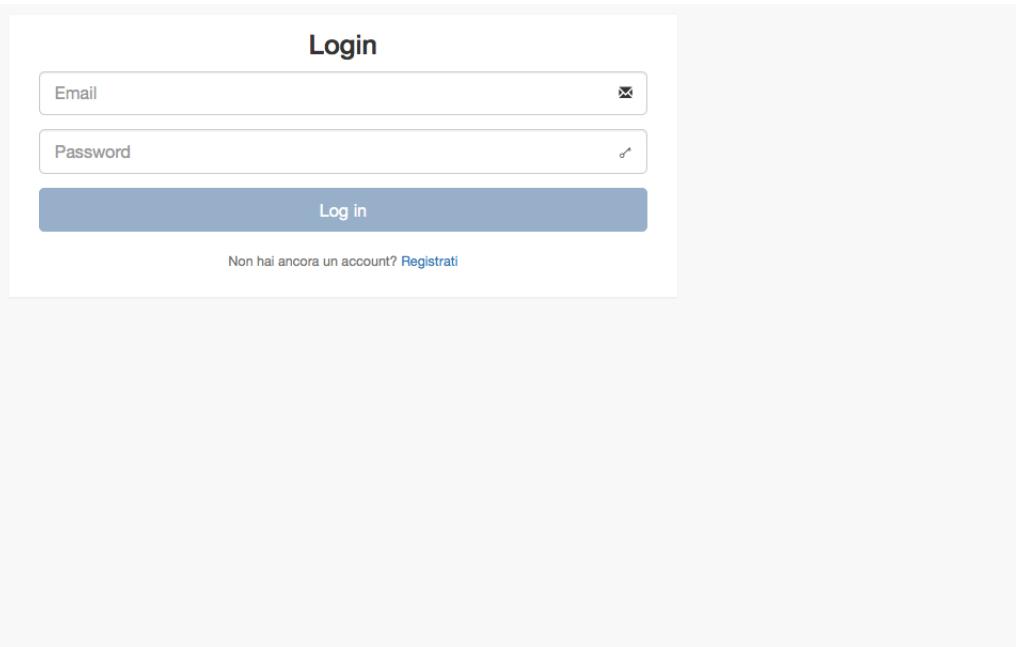
Nota: le immagini qui proposte sono solo un'idea iniziale di come saranno alcune pagine presenti nell'applicazione finale. Possono quindi mancare di alcuni dettagli grafici che verranno aggiunti in seguito.

A.1 Login

A.1.1 Descrizione generale

La pagina di login permette all'utente, che si è già registrato al sistema, di accederci inserendo la propria email e la propria password. Se l'utente non si è ancora registrato al sistema è presente un link per effettuare la registrazione.

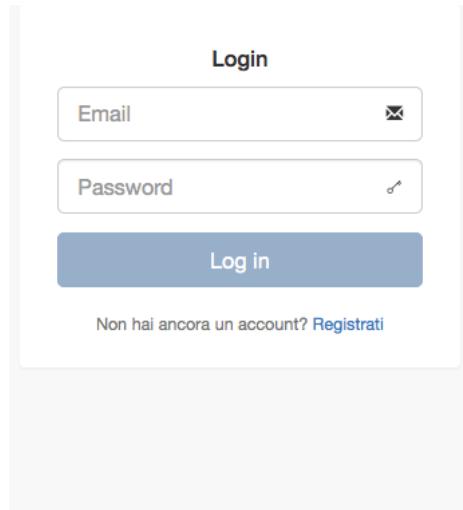
A.1.2 Vista desktop



The image shows a simplified wireframe or mockup of a desktop browser window. The title bar says "Login". Inside, there are two input fields: one labeled "Email" and another labeled "Password", both with clear "X" icons at their ends. Below these is a large blue rectangular button with the white text "Log in". At the bottom of the window, there is a small line of text that reads "Non hai ancora un account? [Registrati](#)". The entire window is set against a light gray background.

Figura 48: Page - Login (vista desktop)

A.1.3 Vista mobile



The image shows a mobile login interface. At the top center is the word "Login". Below it are two input fields: the first for "Email" and the second for "Password". Both fields have small clear icons in the top right corner. Below these fields is a large blue rectangular button with the white text "Log in". At the bottom of the screen, there is a small line of text that reads "Non hai ancora un account? [Registrati](#)".

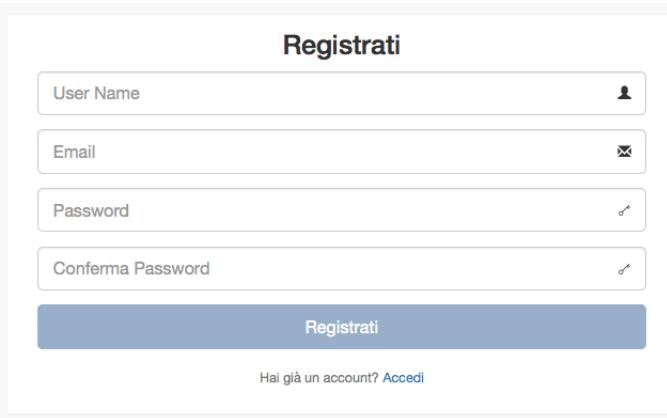
Figura 49: Page - Login (vista mobile)

A.2 Register

A.2.1 Descrizione generale

La pagina register permette all'utente che non si è ancora registrato, di farlo, andando ad inserire uno username, una mail e una password. I campi devono essere tutti compilati e validi, altrimenti la registrazione non potrà terminare con successo.

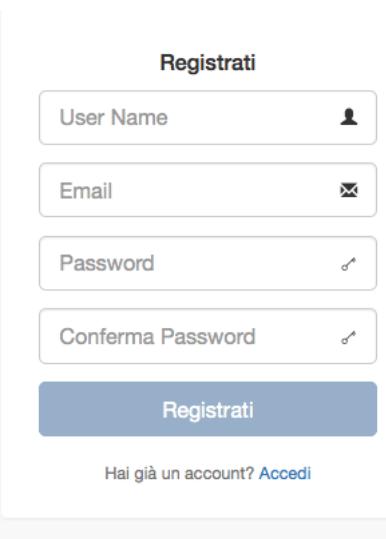
A.2.2 Vista desktop



The screenshot shows a registration form titled "Registrati". It contains four input fields: "User Name" (with a person icon), "Email" (with an envelope icon), "Password" (with a lock icon), and "Conferma Password" (with a lock icon). Below the fields is a large blue "Registrati" button. At the bottom of the form, there is a link "Hai già un account? Accedi".

Figura 50: Page - Register (vista desktop)

A.2.3 Vista mobile



The screenshot shows a registration form titled "Registrati" designed for mobile devices. It contains four input fields: "User Name" (with a person icon), "Email" (with an envelope icon), "Password" (with a lock icon), and "Conferma Password" (with a lock icon). Below the fields is a large blue "Registrati" button. At the bottom of the form, there is a link "Hai già un account? Accedi".

Figura 51: Page - Register (vista mobile)

A.3 Recipe

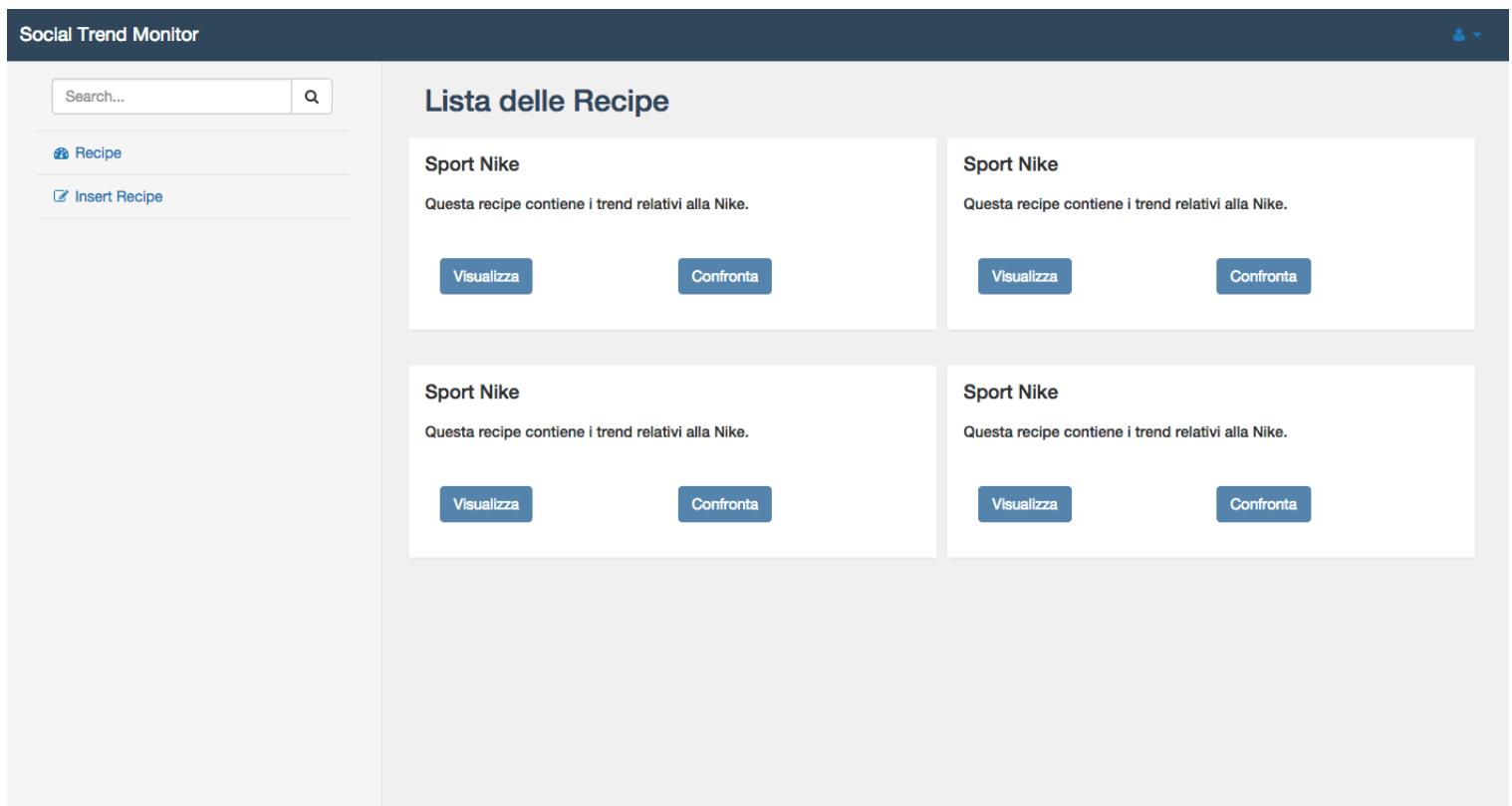
A.3.1 Descrizione generale

La pagina in questione fa visualizzare tutte le recipe_G presenti nel sistema. Per ognuna di esse fornisce il nome e due pulsanti, uno per andare a visualizzare tutte le metriche presenti in essa e uno per andare ad effettuare un confronto tra le metriche.

Nota: Come informazioni generale riguardo a questa pagina e a tutte le successive si ha che da visuale desktop l'header della pagina contiene il menu principale che permette di accedere sempre alla Home page, alla pagina Settings e di effettuare il logout. Questo menu rimane invariato anche nella visuale mobile.

È sempre presente inoltre una barra laterale contenente i collegamenti alla pagine che incapsulano le funzionalità previste per gli utenti. Questa barra, in modalità mobile, scomparirà e verrà rimpiazzata con un menu a tendina che consentirà all'utente di avere una migliore visualizzazione del contenuto principale della pagina.

A.3.2 Vista desktop



The screenshot shows the 'Social Trend Monitor' application interface. At the top, there is a dark header bar with the title 'Social Trend Monitor'. On the left side, there is a sidebar with a search bar, a 'Recipe' button, and an 'Insert Recipe' button. The main content area is titled 'Lista delle Recipe' (List of Recipes). It displays four items, each representing a 'Sport Nike' recipe. Each item has a title ('Sport Nike'), a description ('Questa recipe contiene i trend relativi alla Nike.'), and two buttons at the bottom: 'Visualizza' (View) and 'Confronta' (Compare).

Figura 52: Page - Recipe (vista desktop)

A.3.3 Vista mobile



The image shows a mobile application interface titled "Social Trend Monitor". At the top, there is a dark header bar with the title and a user profile icon. Below the header is a navigation menu icon. The main content area is titled "Lista delle Recipe". A card displays a recipe for "Sport Nike", which is described as containing trends related to Nike. Two buttons at the bottom of the card are labeled "Visualizza" and "Confronta".

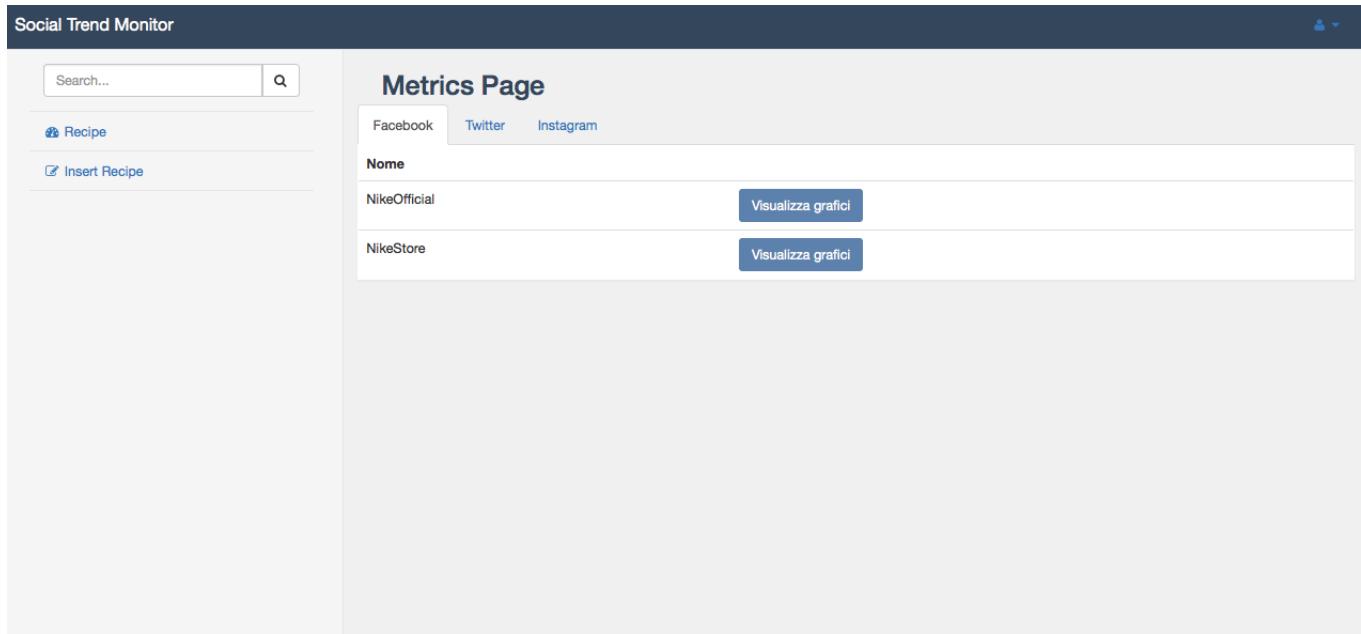
Figura 53: Page - Recipe (vista mobile)

A.4 Metrics

A.4.1 Descrizione generale

La pagina metrics permette all'utente di visualizzare tutte le metriche appartenenti ad una determinata Recipe_G. Le metriche vengono mostrate divisi in diversi pannelli a seconda della categoria_G di cui fan parte.

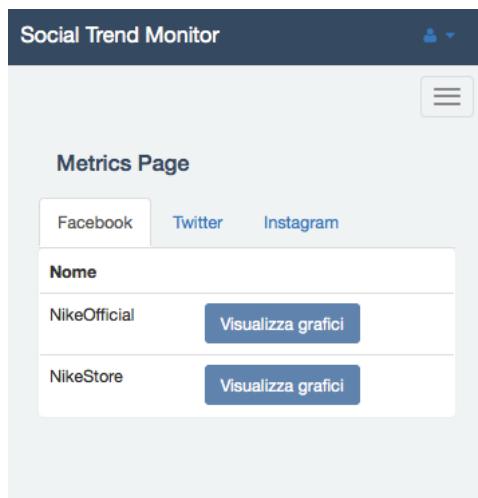
A.4.2 Vista desktop



The screenshot shows the 'Metrics Page' of the Social Trend Monitor application. At the top, there is a search bar with placeholder text 'Search...' and a magnifying glass icon. Below the search bar, there are three tabs: 'Facebook' (selected), 'Twitter', and 'Instagram'. A sidebar on the left contains a 'Recipe' section with a 'Nike' logo and a link to 'Insert Recipe'. The main content area displays two entries under the heading 'Nome': 'NikeOfficial' and 'NikeStore', each with a 'Visualizza grafici' button.

Figura 54: Page - Metrics (vista desktop)

A.4.3 Vista mobile



The screenshot shows the 'Metrics Page' of the Social Trend Monitor application in mobile view. The interface is similar to the desktop version, featuring a search bar, a sidebar with a 'Recipe' section, and a main content area with 'Nome' entries ('NikeOfficial' and 'NikeStore') and 'Visualizza grafici' buttons. A menu icon (three horizontal lines) is visible in the top right corner of the main content area.

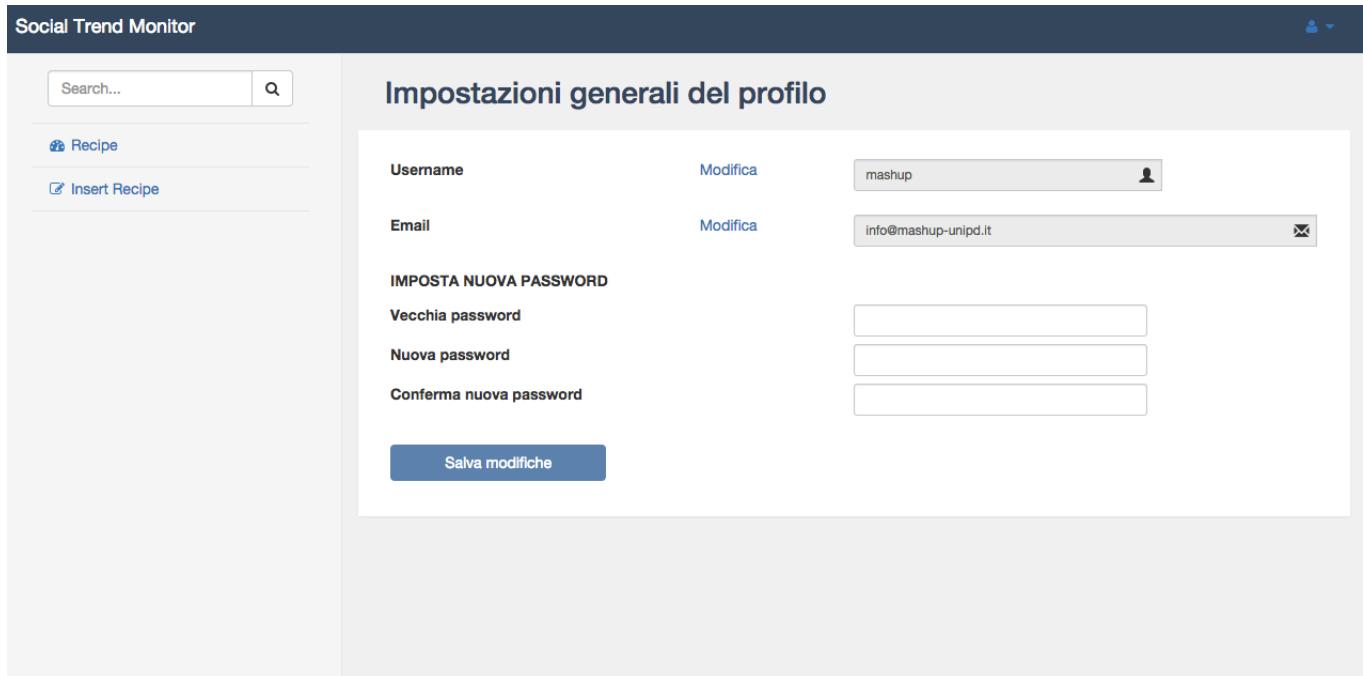
Figura 55: Page - Metrics (vista mobile)

A.5 Settings

A.5.1 Descrizione generale

La pagina settings permette all'utente di visualizzare i dati relativi al proprio account salvati nel sistema e di effettuare delle modifiche su di essi. In particolare per il cambio password, sarà richiesto di inserire la quella vecchia e successivamente per due volte quella nuova.

A.5.2 Vista desktop



Social Trend Monitor

Search...

[Recipe](#)
[Insert Recipe](#)

Impostazioni generali del profilo

Username	Modifica	mashup
Email	Modifica	info@mashup-unipd.it

IMPOSTA NUOVA PASSWORD

Vecchia password
Nuova password
Conferma nuova password

Figura 56: Page - Settings (vista desktop)

A.5.3 Vista mobile

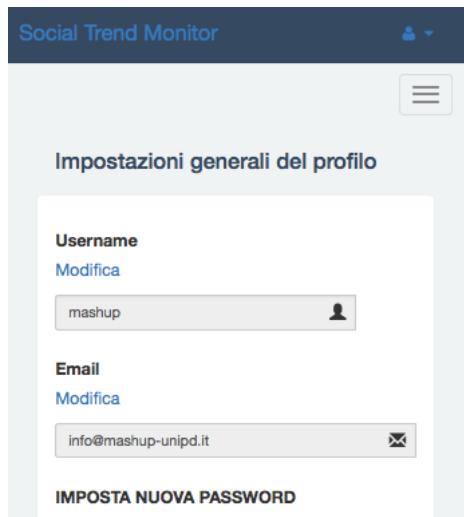


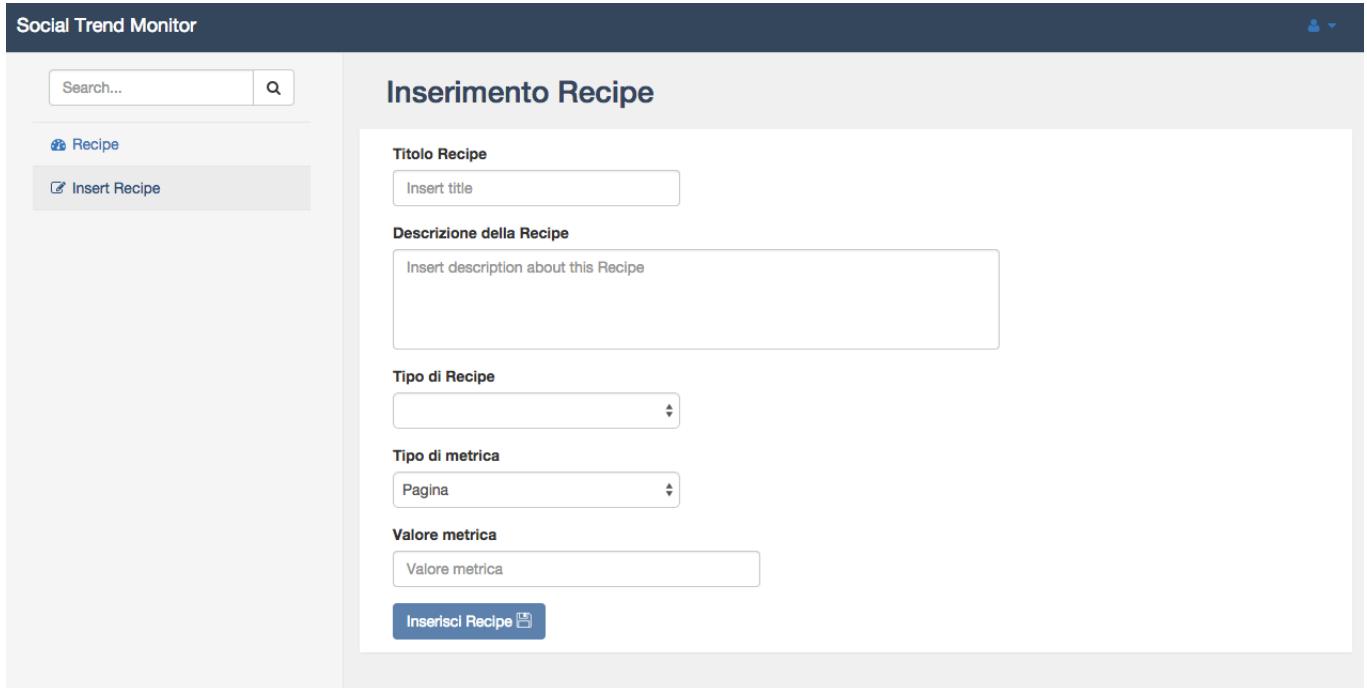
Figura 57: Page - Settings (vista mobile)

A.6 RecipeConfig

A.6.1 Descrizione generale

La pagina permette all'amministratore di andare ad inserire una nuova Recipe_G nel sistema. Essa visualizza un form_G che dovrà essere compilato in maniera adeguata per poter permettere l'inserimento della Recipe. Il form è molto guidato da scelte vincolate dal sistema tramite dei combo box che limitano i possibili errori umani.

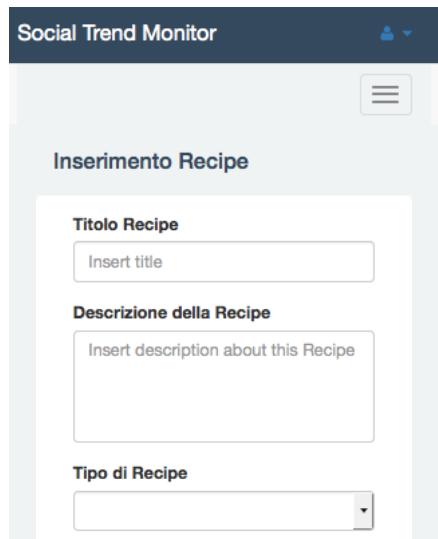
A.6.2 Vista desktop



The screenshot shows the 'Inserimento Recipe' (Insert Recipe) page of the Social Trend Monitor application. The page has a dark header bar with the title 'Social Trend Monitor' and a search bar. On the left, there's a sidebar with 'Recipe' and 'Insert Recipe' buttons. The main area is titled 'Inserimento Recipe'. It contains several input fields: 'Titolo Recipe' (Title Recipe) with placeholder 'Insert title'; 'Descrizione della Recipe' (Description of Recipe) with placeholder 'Insert description about this Recipe'; 'Tipo di Recipe' (Type of Recipe) with a dropdown menu; 'Tipo di metrica' (Type of metric) with a dropdown menu set to 'Pagina' (Page); and 'Valore metrica' (Metric value) with a placeholder 'Valore metrica'. At the bottom is a blue 'Inserisci Recipe' (Insert Recipe) button.

Figura 58: Page - Recipe Config (vista desktop)

A.6.3 Vista mobile



The image shows a mobile application interface titled "Social Trend Monitor". At the top, there is a user icon and a menu icon. Below the title, the section "Inserimento Recipe" is displayed. It contains three input fields: "Titolo Recipe" with placeholder "Insert title", "Descrizione della Recipe" with placeholder "Insert description about this Recipe", and "Tipo di Recipe" with a dropdown menu open. The entire interface is designed for mobile devices.

Figura 59: Page - Recipe Config (vista mobile)

B Descrizione Design Pattern

B.1 Design pattern architetturali

B.1.1 Three-Tier

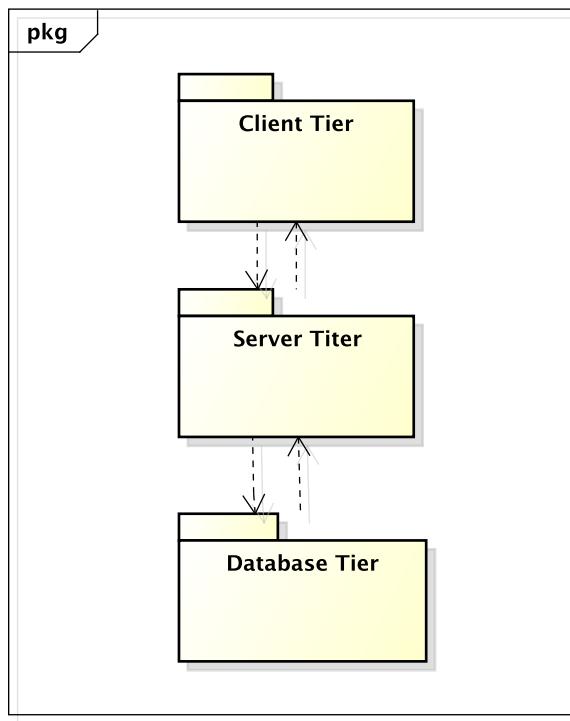


Figura 60: Design pattern architetturale - Three-Tier

- Scopo:** L’architettura Three-Tier, detta anche “a tre strati”, identifica una particolare architettura software per l’esecuzione di un’applicazione web che prevede la suddivisione dell’applicazione in tre diversi moduli o strati dedicati rispettivamente all’interfaccia utente, alla logica funzionale e alla gestione dei dati persistenti. Tale architettura va tipicamente a mappare a livello fisico-infrastrutturale quella del sistema informatico ospitante l’applicazione da eseguire. Tali moduli sono intesi interagire fra loro secondo le linee generali del paradigma client-server (l’interfaccia è cliente della business logic, e questa è cliente del modulo di gestione dei dati persistenti) e utilizzando interfacce ben definite. In questo modo, ciascuno dei tre moduli può essere modificato o sostituito indipendentemente dagli altri conferendo scalabilità e manutenibilità all’applicazione. Nella maggior parte dei casi, i diversi moduli possono essere distribuiti su diversi nodi di una rete anche eterogenea;
- Motivazione:** È stato scelto questo pattern perché il suo utilizzo è particolarmente diffuso nelle web application per poter disporre di moduli distribuiti, posti su piattaforme separate. Questa separazione aumenta le prestazioni e la scalabilità, l’estensibilità dei moduli e lo sviluppo parallelo degli stessi;
- Applicabilità:** Oltre ai vantaggi abituali nell’utilizzare software modulare con interfacce ben definite, l’architettura Three-Tier consente anche a qualsiasi dei tre livelli di essere aggiornato o sostituito indipendentemente dal cambiamento di requisiti o tecnologia nell’applicazione;

B.1.2 MVC

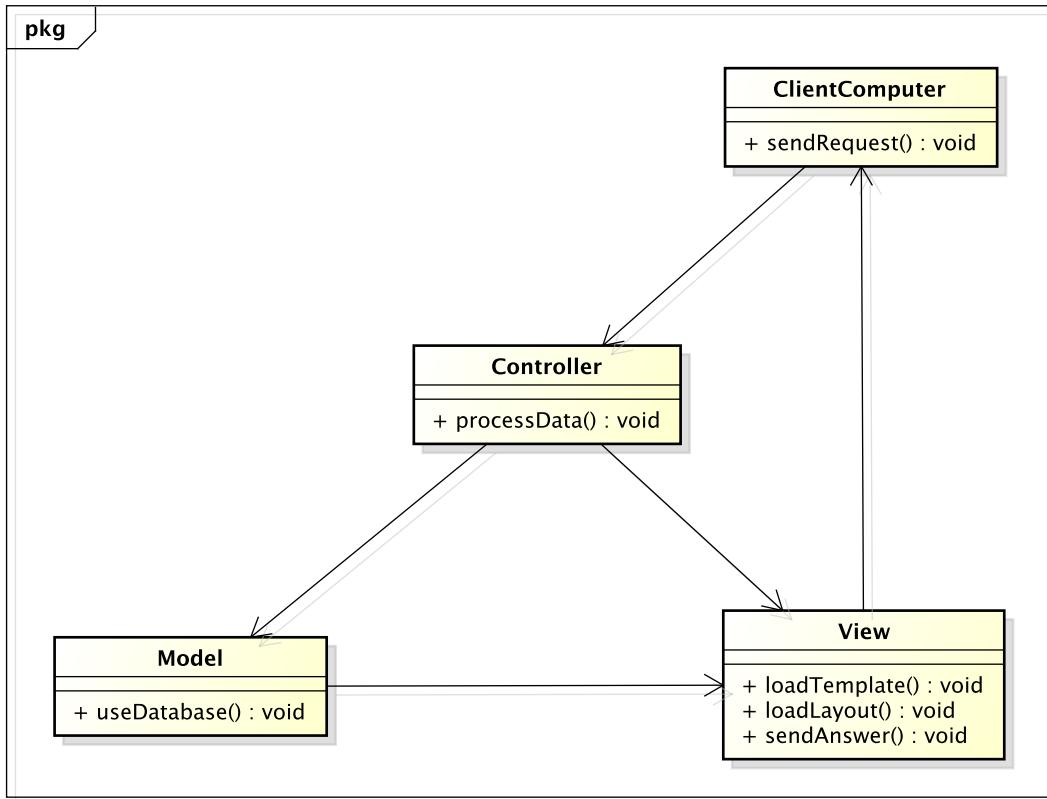


Figura 61: Design pattern architetturale - MVC

- **Scopo:** Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:
 - **Model** che fornisce i metodi per accedere ai dati utili all'applicazione;
 - **View_G** che visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
 - **Controller** che riceve i comandi dell'utente (in genere attraverso il view_G) e li attua modificando lo stato degli altri due componenti.

Questo schema implica anche la tradizionale separazione fra la logica applicativa (in questo contesto spesso chiamata “logica di business”), a carico del Controller e del model, e l’interfaccia utente a carico del view_G. I dettagli delle interazioni fra questi tre oggetti software dipendono molto dalle tecnologie usate (linguaggio di programmazione, eventuali librerie, middleware e via dicendo) e dal tipo di applicazione (per esempio se si tratta di un'applicazione web, o di un'applicazione desktop). Quasi sempre la relazione fra view e model è descrivibile anche come istanza del pattern Observer. A volte, quando è necessario cambiare il comportamento standard dell'applicazione a seconda delle circostanze, il Controller implementa anche il pattern Strategy.;

- **Motivazione:** È stato scelto questo pattern perché è nata la necessità di rendere modulari le funzionalità dell'interfaccia utente. Utilizzando MVC viene quindi separata la modellazione dal dominio, dalla presentazione e dalle azioni basate sugli input degli utenti in tre classi separate;

- **Applicabilità:** Il pattern MVC può essere utilizzato in diversi casi:

- Quando si vuole disaccoppiare View_G e Model instaurando un protocollo di sottoscrizione e notifica tra loro;
- Quando si vuole trattare un gruppo di oggetti come un oggetto singolo;
- Quando si vogliono agganciare più View_G a un Model per fornire più rappresentazioni del Model stesso;

B.1.3 MVVM

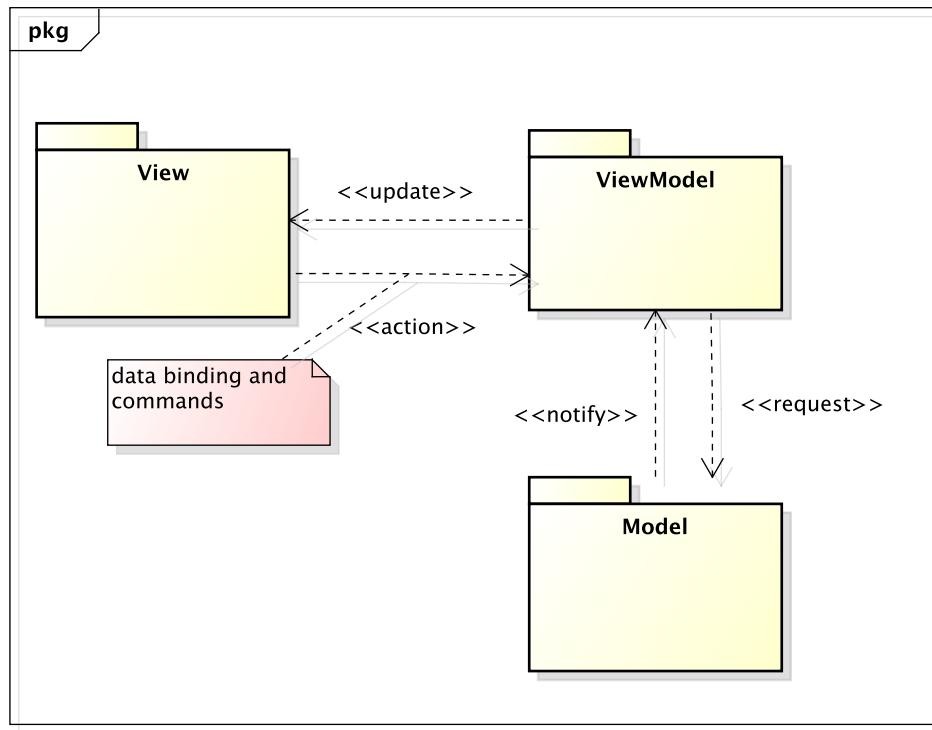


Figura 62: Design pattern architetturale - MVVM

- **Scopo:** È una variante del pattern MVC. Questo pattern propone un ruolo più attivo della View_G rispetto a MVC: la View è in grado di gestire eventi, eseguire operazioni ed effettuare il data-binding. In questo contesto, quindi, alcune delle funzionalità del Controller vengono inglobate nella View, la quale si appoggia su un'estensione del Model: il ViewModel. Il ViewModel è quindi un Model esteso con funzionalità per la manipolazione dei dati e per l'interazione con la View_G ;
- **Motivazione:** È stato scelto questo pattern perché ha un impatto positivo nella progettazione di interfacce utente e viene implementato in modo semplice da Angular JS;
- **Applicabilità:** Il pattern MVVM è consigliato qualora si voglia separare interamente la progettazione dell'interfaccia grafica dalla business logic dell'applicazione;

B.1.4 Dependency Injection

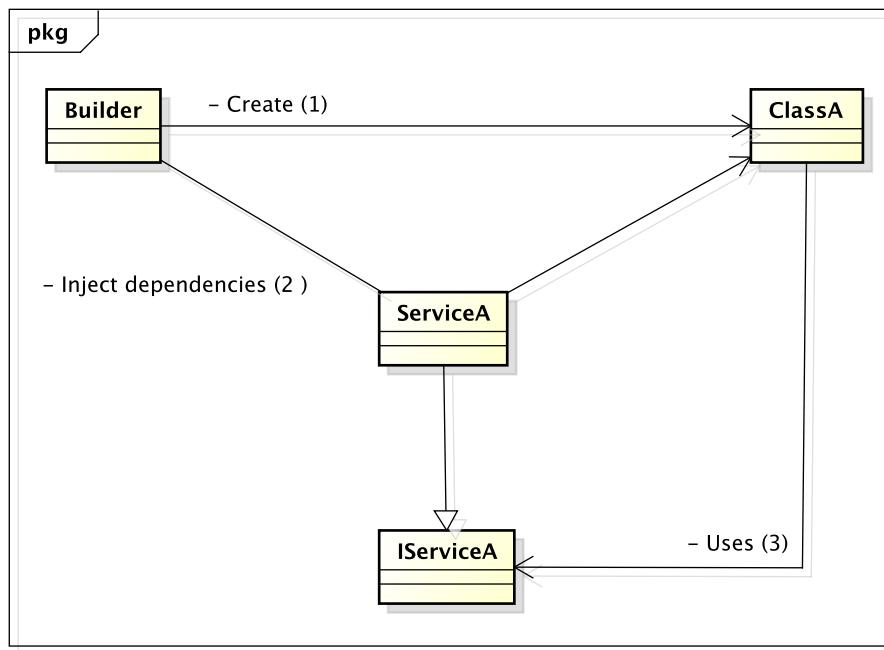


Figura 63: Design pattern architetturale - Dependency Injection

- **Scopo:** È quello di semplificare lo sviluppo e migliorare la testabilità di software di grandi dimensioni. Il pattern Dependency Injection coinvolge almeno tre elementi:
 - una componente dipendente;
 - la dichiarazione delle dipendenze del componente, definite come interface contracts;
 - un injector (chiamato anche provider o container) che crea, a richiesta, le istanze delle classi che implementano delle dependency interfaces;
- **Motivazione:** È stato scelto questo pattern perché semplifica lo sviluppo e migliorare la testabilità di software di grandi dimensioni;
- **Applicabilità:** Il pattern Dependency Injection viene utilizzato principalmente nei seguenti casi di:
 - Constructor Injection, dove la dipendenza viene iniettata tramite l'argomento del costruttore;
 - Setter Injection, dove la dipendenza viene iniettata attraverso un metodo "set";
 - Interface Injection, che si basa sul mapping tra interfaccia e relativa implementazione;

La Dependency Injection può essere realizzata in molteplici modi, tra cui il più semplice consiste nell'utilizzo di un factory method;

B.2 Design pattern creazionali

B.2.1 Prototype Pattern

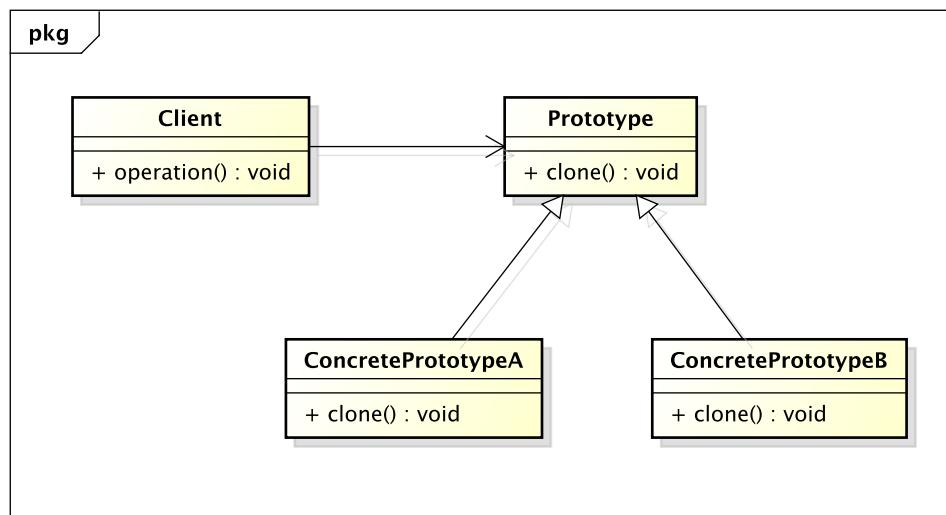


Figura 64: Design pattern creazionale - Prototype Pattern

- **Scopo:** Prototype permette di creare nuovi oggetti clonando un oggetto iniziale, detto appunto prototipo. A differenza di altri pattern come Abstract Factory o Factory Method permette di specificare nuovi oggetti a tempo d'esecuzione (run-time), utilizzando un gestore di prototipi (detto prototype manager) per salvare e reperire dinamicamente le istanze degli oggetti desiderati;
- **Motivazione:** È stato scelto questo pattern perché permette di encapsulare al suo interno la modalità di istanziazione degli oggetti, liberando il client dalla necessità di conoscere i nomi delle classi da istanziare. Inoltre permette di ridurre la complessità della gerarchia delle sottoclassi rispetto al Factory Method ed evitare la duplicazione di codice quando si usano diverse classi simili tra loro;
- **Applicabilità:** Come altri pattern creazionali, Prototype mira a rendere indipendente un sistema dal modo in cui i suoi oggetti vengono creati;

B.2.2 Module Pattern

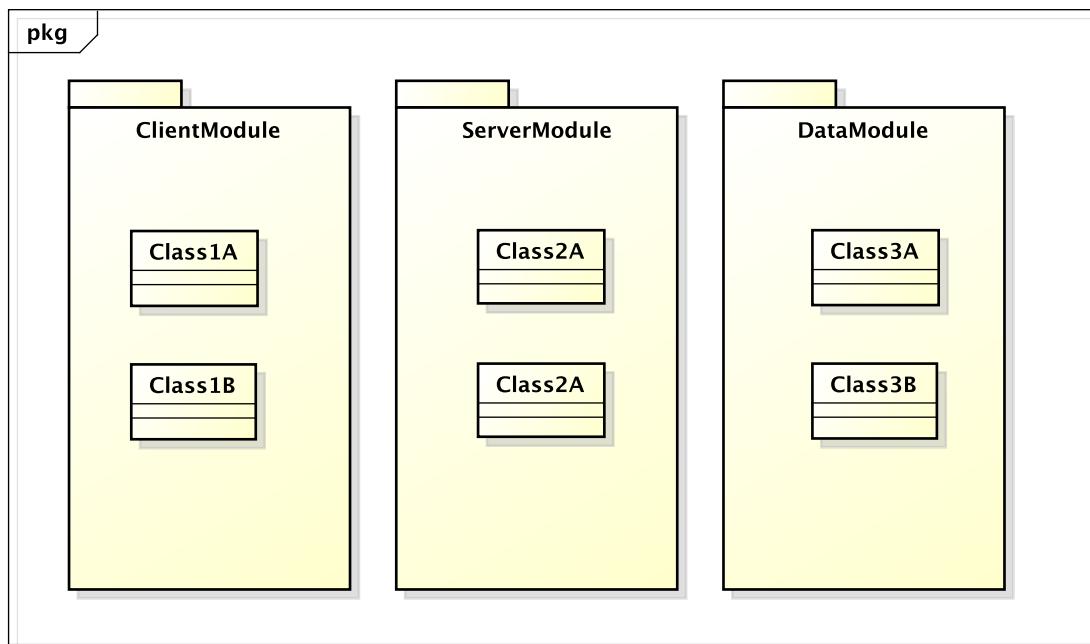


Figura 65: Design pattern creazionale - Module Pattern

- **Scopo:** Il module pattern prevede che l'applicazione che lo implementa sia divisa in moduli funzionali distinti. Essi consentono di organizzare le parti di un'applicazione in unità separate ma integrabili grazie ai meccanismi di esportazione e importazione, cioè rispettivamente della possibilità di rendere pubblicamente accessibile del codice e di accedere a codice esportato da altri moduli;
- **Motivazione:** È stato scelto questo pattern perché permette all'applicazione di essere robusta e facilmente manutenibile definendo un codice più chiaro e modulare. Permette inoltre una rapida risoluzione dei nomi, evitando ambiguità con i metodi e le variabili di altre funzioni globali;
- **Applicabilità:** Il module pattern è un design pattern usato per implementare il concetto dei moduli software in linguaggi di programmazione che non lo supportano nativamente, come per esempio JavaScript;

B.2.3 Singleton

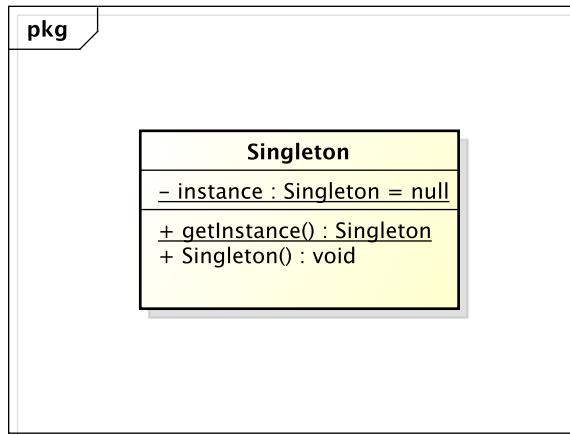


Figura 66: Design pattern creazionale - Singleton

- **Scopo:** Il singleton ha lo scopo di garantire che di una determinata classe venga creata una e una sola istanza, e di fornire un punto di accesso globale a tale istanza;
- **Motivazione:** È stato scelto questo pattern perché permette di creare istanze uniche di classi all'interno dell'applicazione;
- **Applicabilità:** Il pattern Singleton può essere utilizzato quando deve esistere una sola istanza di una classe in tutta l'applicazione e quando l'unica istanza deve poter essere estesa attraverso la definizione di sottoclassi garantendo che i client possano utilizzare le istanze estese senza dover modificare il proprio codice;

B.3 Design pattern strutturali

B.3.1 Façade

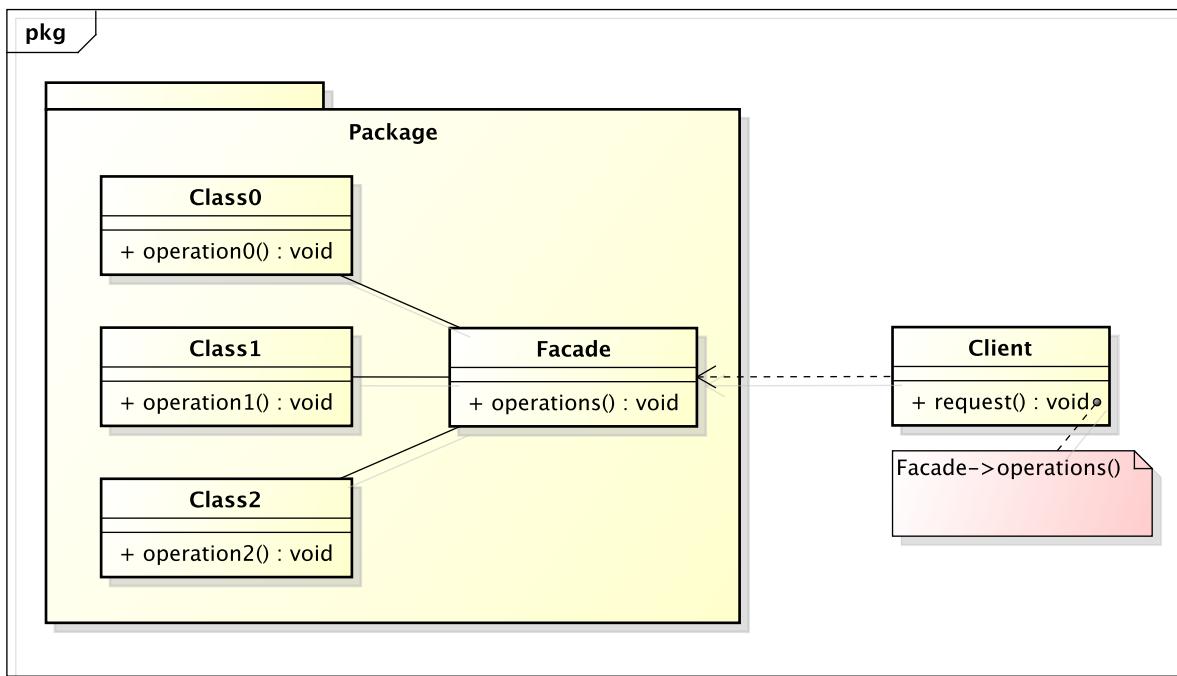


Figura 67: Design pattern strutturale - Façade

- **Scopo:** Il Façade pattern suggerisce la creazione di un oggetto che presenti un’interfaccia semplificata al cliente, ma in grado di gestire tutta la complessità delle interazioni tra gli oggetti delle diverse classi per compiere l’obiettivo desiderato;
- **Motivazione:** È stato scelto questo pattern perché fornisce una interfaccia unificata per un insieme di interfacce di un sottosistema, rendendo più facile l’uso di quest’ultimo;
- **Applicabilità:** Questo pattern può essere utilizzato per fornire una vista semplice e di default di un sottosistema complesso e nel caso di sottosistema stratificato, come entry point per ciascun livello;

B.3.2 Adapter

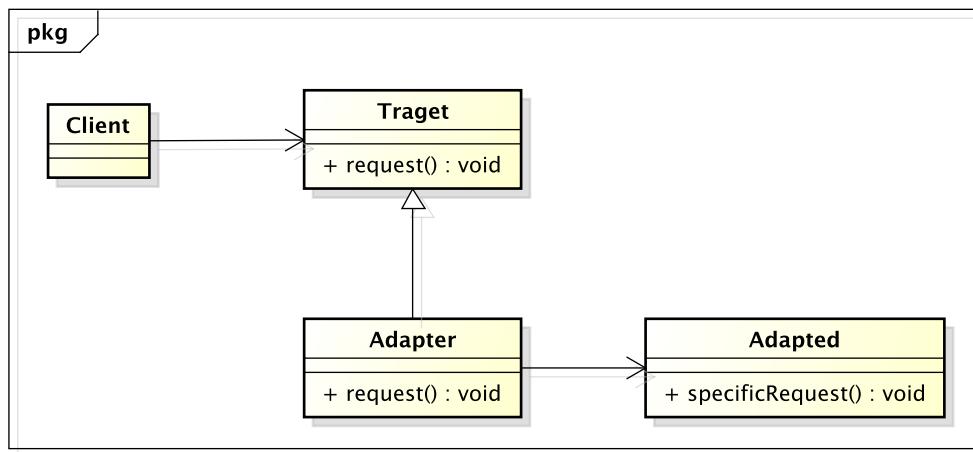


Figura 68: Design pattern strutturale - Adapter

- **Scopo:** Lo scopo è quello di fornire una soluzione astratta al problema dell'interoperabilità tra interfacce differenti. Questo problema si presenta ogni volta nel progetto di un software si debbano utilizzare oggetti la cui interfaccia non è perfettamente compatibile con quanto richiesto da applicazioni già esistenti. Invece di riscrivere parte del sistema, compito oneroso e non sempre possibile se non si ha a disposizione il codice sorgente, può essere comodo scrivere un adapter, appunto, che faccia da tramite;
- **Motivazione:** È stato scelto questo pattern perché permette l'interoperabilità tra diversi moduli, anche di terze parti, senza doverli modificare o riscrivere;
- **Applicabilità:** Questo pattern può essere utilizzato quando interfacce di classi differenti devono comunque poter comunicare tra loro. Alcuni casi sono:
 - l'utilizzo di una classe esistente che presenta un'interfaccia diversa da quella desiderata;
 - la scrittura di una determinata classe senza poter conoscere a priori le altre classi con cui dovrà operare;

B.4 Design pattern comportamentali

B.4.1 Page Controller

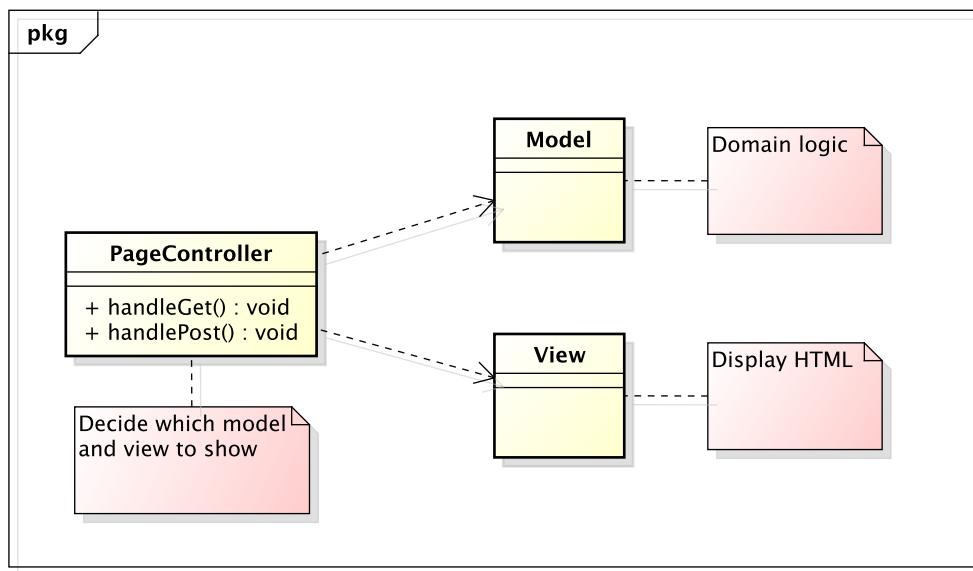


Figura 69: Design pattern comportamentale - Page Controller

- **Scopo:** Il Page Controller pattern suggerisce la creazione di un oggetto dedicato che si occupa di gestire una richiesta per una specifica pagina o azione in un sito Web. Implica quindi che esiste un singolo file che gestisce la richiesta di una specifica pagina. Questo semplice meccanismo rispecchia il funzionamento delle pagine web statiche;
- **Motivazione:** È stato scelto questo pattern perché semplifica la gestione delle pagine web statiche utilizzate dall'applicazione e permette di eseguire test e riutilizzare codice in modo agevole;
- **Applicabilità:** Questo pattern può essere utilizzato in quei casi in cui si voglia utilizzare un singolo oggetto per gestire tutte le richieste di una singola pagina logica, a differenza del Front Controller che invece gestisce le richieste di molteplici pagine logiche con un singolo oggetto;

B.4.2 Template View

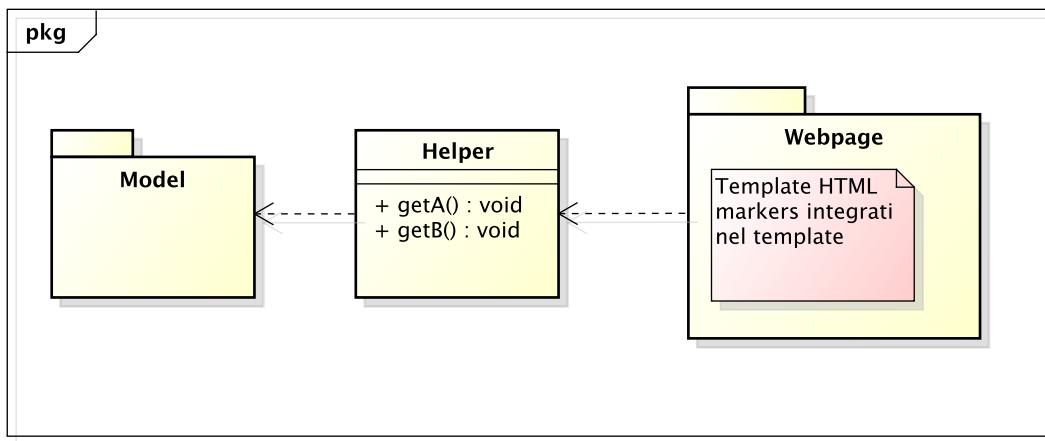


Figura 70: Design pattern comportamentale - Template View

- **Scopo:** Lo scopo di questo pattern è quello di implementare la componente View_G del design pattern $\text{Model View Controller}$ mediante l'uso di template HTML . In questo modo i template svolgono il compito di isolare la presentation logic dal codice del linguaggio di programmazione utilizzato nell'applicazione web.
- **Motivazione:** È stato scelto questo pattern perché riduce in modo marcato la complessità della realizzazione di una pagina dinamica, tramite l'inserimento di marcatori in una struttura prefissa che serviranno come punti di ingresso per il contenuto dinamico generato;
- **Applicabilità:** Questo pattern viene utilizzato quando si vuole generare una pagina web dinamica allo stesso modo di una pagina web statica, definendo una struttura fissa e completandola con il contenuto dinamico richiesto;

B.4.3 Template Method

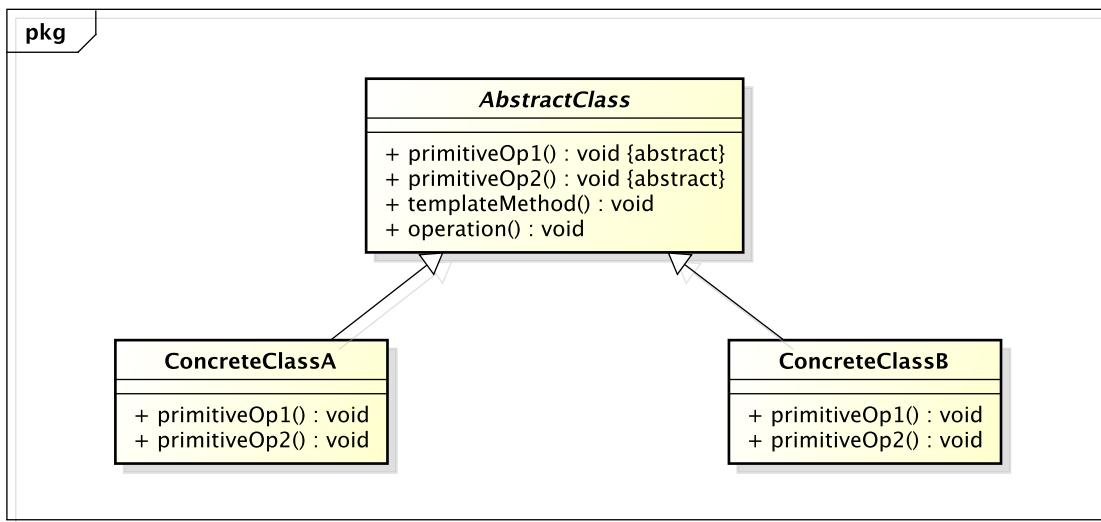


Figura 71: Design pattern comportamentale - Template Method

- **Scopo:** Il Template_G Method è un pattern comportamentale basato sulle classi e viene utilizzato prevalentemente nell’ambito della programmazione ad oggetti. Questo pattern permette di definire la struttura di un algoritmo delegando alle sottoclassi il compito di implementarne alcuni passi. In questo modo si può ridefinire e personalizzare parte del comportamento nelle varie sottoclassi senza dover riscrivere più volte il codice in comune nella classe principale;
- **Motivazione:** È stato scelto questo pattern perché permette una maggior libertà nella definizione dei metodi, dando la possibilità di lavorare nelle sottoclassi senza dover apportare modifiche alla classe astratta principale;
- **Applicabilità:** Questo pattern può essere utilizzato nei seguenti casi:
 - quando si vuole implementare la parte invariante di un algoritmo una volta sola e lasciare che le sottoclassi implementino il comportamento variabile dello stesso;
 - quando il comportamento comune di più classi può essere inserito all’interno di una classe a parte per evitare la duplicazione di codice;
 - per avere modo di controllare quali metodi vengono ereditati dalla superclasse, facendo in modo che i metodi template_G siano gli unici metodi sovra scrivibili;

B.4.4 Command

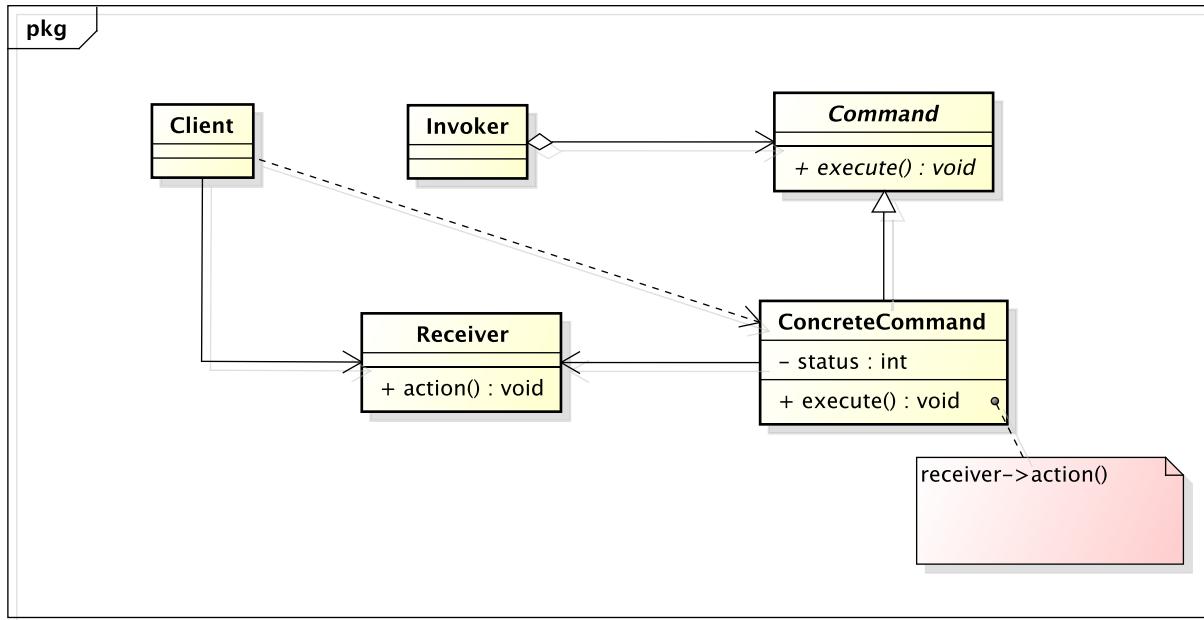


Figura 72: Design pattern comportamentale - Command

- **Scopo:** Il Command pattern permette di isolare la porzione di codice che effettua un’azione anche molto complessa dal codice che ne richiede l’esecuzione; l’azione viene tipicamente incapsulata in un oggetto specifico detto “command”. L’obiettivo è rendere variabile l’azione del client senza però conoscere i dettagli dell’operazione stessa. Un altro aspetto importante è che il destinatario della richiesta può non essere deciso staticamente all’atto dell’istanziazione dell’oggetto “command” ma ricavato a tempo di esecuzione;
- **Motivazione:** È stato scelto questo pattern perché permette agli oggetti “command” di essere decisi dinamicamente in base all’operazione che bisogna eseguire;
- **Applicabilità:** Questo pattern può essere utilizzato nei casi in cui:
 - si richiede che un’azione sia atomica: si può implementare un oggetto di comando in modo che le transazioni al suo interno vengano svolte in toto o per nulla;
 - si vuole rendere asincrona la scelta dei comandi rispetto alla loro esecuzione. Un certo numero di “command” possono essere elaborati da un altro oggetto che li riceve in un tempo diverso dalla loro selezione;