

NORRIS Framework



FLAMETECH Inc.

Definizione di Prodotto

Informazioni sul documento

Versione	2.0.0
Redazione	Faggin Andrea Meneguzzo Francesco Persegona Mattia
Verifica	Sartor Michele
Responsabile	Zanetti Davide
Uso	Esterno
Lista di distribuzione	FlameTech Inc. Prof. Vardanega Tullio Prof. Cardin Riccardo CoffeeStrap

Descrizione

Questo documento descrive la progettazione di dettaglio definita dal gruppo **FlameTech Inc.** relativa al progetto Norris.



Versione	Modifica	Autore	Ruolo	Data	Stato
2.0.0	Approvazione documento	Zanetti Davide	Responsabile	2015/06/11	Approvato
1.1.0	Verifica documento	Sartor Michele	Verificatore	2015/06/11	Verificato
1.0.13	Aggiunta metodi e campi dati ai controller delle componenti di NorrisApp, con relativa spiegazione di dettaglio	Faggin Andrea	Progettista	2015/06/10	In Lavorazione
1.0.12	Spiegazione mancanza di attributi e metodi delle componenti che rappresentano le view nella sezione 3.2.2.2	Faggin Andrea	Progettista	2015/06/10	In Lavorazione
1.0.11	Inserimento termini “mount” e “subrouting” in glossario	Faggin Andrea	Progettista	2015/06/09	In Lavorazione
1.0.10	Corretta privatizzazione dei campi della classe ActiveResources	Persegona Mattia	Progettista	2015/06/09	In Lavorazione
1.0.9	Correzione staticità costruttori della classi del DataLayer	Persegona Mattia	Progettista	2015/06/09	In Lavorazione
1.0.8	Dettagliata maggiormente la descrizione dei metodi della classe SocketService e rimandata descrizione di dettaglio della classe SocketController alle funzioni di dettaglio della prima classe	Meneguzzo Francesco	Progettista	2015/06/09	In Lavorazione
1.0.7	Corretta descrizione metodi della classe DataConsistency aggiungendo i valori corretti per ogni metodo	Faggin Andrea	Progettista	2015/06/09	In Lavorazione
1.0.6	Descrizione struttura generale dei template della descrizione della classe DataConsistency	Meneguzzo Francesco	Progettista	2015/06/09	In Lavorazione



Versione	Modifica	Autore	Ruolo	Data	Stato
1.0.5	Ampliata e chiarita descrizione della classe ProgressiveID per evitare accessi concorrenti e medesimo ID tra risorse diverse	Meneguzzo Francesco	Progettista	2015/06/08	In Lavorazione
1.0.4	Corretta mancanza dei tipi di ritorno nei diagrammi delle classi	Persegona Mattia	Progettista	2015/06/08	In Lavorazione
1.0.3	Corrette occorrenze del termine “settata” a “imposta”	Persegona Mattia	Progettista	2015/06/08	In Lavorazione
1.0.2	Inserita descrizione per la motivazione dei metodi principalmente statici nella sezione 2.4	Faggin Andrea	Progettista	2015/06/08	In Lavorazione
1.0.1	Inserita sezione 3.2.2 per introdurre nuovamente l'architettura generale del prodotto	Meneguzzo Francesco	Progettista	2015/06/08	In Lavorazione
1.0.0	Approvazione documento	Meneguzzo Francesco	Responsabile	2015/05/01	Approvato
0.1.0	Verifica documento	Zanetti Davide	Verificatore	2015/05/01	Verificato
0.0.8	Stesura tracciamento	Cardin Andrea	Progettista	2015/04/30	In Lavorazione
0.0.7	Termine stesura sezione 3	Merlo Gianluca	Progettista	2015/04/30	In Lavorazione
0.0.6	Termine stesura sezione 4	Persegona Mattia	Progettista	2015/04/30	In Lavorazione
0.0.5	Inizio stesura sezione 4	Persegona Mattia	Progettista	2015/04/29	In Lavorazione
0.0.4	Inizio stesura sezione 3	Merlo Gianluca	Progettista	2015/04/29	In Lavorazione
0.0.3	Stesura sezione Standard di Progetto	Persegona Mattia	Progettista	2015/04/28	In Lavorazione
0.0.2	Stesura sezione Introduzione	Cardin Andrea	Progettista	2015/04/28	In Lavorazione
0.0.1	Impostazione scheletro del documento	Persegona Mattia	Progettista	2015/04/28	In Lavorazione

Indice

1	Introduzione	1
1.1	Scopo del documento	1
1.2	Scopo del prodotto	1
1.3	Glossario	1
1.4	Riferimenti	1
1.4.1	Normativi	1
1.4.2	Informativi	1
2	Standard di Progetto	2
2.1	Standard di progettazione architettuale	2
2.2	Standard di documentazione del codice	2
2.3	Standard di denominazione di entità e relazioni	2
2.4	Standard di programmazione	2
2.5	Strumenti di lavoro	2
3	Descrizione Architettura	3
3.1	Metodo e formalismo di specifica	3
3.2	Architettura generale	3
3.2.1	Interfaccia REST-like	3
3.2.2	Norris	5
3.2.2.1	Lib	6
3.2.2.2	NorrisApp	9
4	Specifica classi del <i>back-end_G</i>	11
4.1	Componente Norris::Lib::BusinessLayer	11
4.1.1	Classe ActiveRecordController	11
4.1.2	Classe BarChartController	12
4.1.3	Classe DataConsistency	14
4.1.4	Classe LineChartController	19
4.1.5	Classe MapChartController	21
4.1.6	Classe PageController	22
4.1.7	Classe SocketController	23
4.1.8	Classe TableController	25
4.2	Componente Norris::Lib::DataLayer	29
4.2.1	Classe ActiveRecord	29
4.2.2	Classe BarChartModel	30
4.2.3	Classe LineChartModel	32
4.2.4	Classe MapChartModel	34
4.2.5	Classe PageModel	36
4.2.6	Classe TableModel	37
4.3	Componente Norris::Lib::PresentationLayer	38
4.3.1	Classe BarChart	38
4.3.2	Classe LineChart	40
4.3.3	Classe MapChart	41
4.3.4	Classe Norris	43
4.3.5	Classe Page	43
4.3.6	Classe PageRouter	44
4.3.7	Classe Table	45



4.4	Componente Norris::Lib::Utils	46
4.4.1	Classe ColorManager	46
4.4.2	Classe NorrisError	47
4.4.3	Classe ProgressiveID	48
4.4.4	Classe SocketService	48
5	Specifica classi del <i>front-end</i>_G	50
5.1	Componente Norris::NorrisApp::Controllers	50
5.1.1	Classe BarLineChartCtrl	50
5.1.2	Classe FrontCtrl	51
5.1.3	Classe MapChartCtrl	52
5.1.4	Classe TableCtrl	53
5.2	Componente Norris::NorrisApp::Model	54
5.2.1	Classe BarChartMdl	54
5.2.2	Classe FrontMdl	55
5.2.3	Classe LineChartMdl	56
5.2.4	Classe MapChartMdl	57
5.2.5	Classe TableMdl	58
5.3	Componente Norris::NorrisApp::Services	59
5.3.1	Classe BarLineSvc	59
5.3.2	Classe ColorsSvc	60
5.3.3	Classe FirstConnectSvc	61
5.3.4	Classe FrontSvc	62
5.3.5	Classe MapSvc	63
5.3.6	Classe SocketsSvc	65
5.3.7	Classe TableSvc	67
5.4	Componente Norris::NorrisApp::Views	68
5.4.1	Classe BarChartView	68
5.4.2	Classe Index	69
5.4.3	Classe LineChartView	69
5.4.4	Classe MapChartView	70
5.4.5	Classe TableView	70
6	Tracciamento	72
6.1	Tracciamento requisiti-classi	72
6.2	Tracciamento classi-requisiti	76
6.3	Tracciamento test di unità - metodi	81
6.4	Tracciamento metodi-test di unità	85



Elenco delle tabelle

2	Chiamate URI	4
3	Classe ActiveResourcesController	11
4	Classe BarChartController	12
5	Classe DataConsistency	14
6	Classe LineChartController	19
7	Classe MapChartController	21
8	Classe PageController	22
9	Classe SocketController	23
10	Classe TableController	25
11	Classe ActiveResources	29
12	Classe BarChartModel	30
13	Classe LineChartModel	32
14	Classe MapChartModel	34
15	Classe PageModel	36
16	Classe TableModel	37
17	Classe BarChart	38
18	Classe LineChart	40
19	Classe MapChart	41
20	Classe Norris	43
21	Classe Page	43
22	Classe PageRouter	44
23	Classe Table	45
24	Classe ColorManager	46
25	Classe NorrisError	47
26	Classe ProgressiveID	48
27	Classe SocketService	48
28	Classe BarLineChartCtrl	50
29	Classe FrontCtrl	51
30	Classe MapChartCtrl	52
31	Classe TableCtrl	53
32	Classe BarChartMdl	54
33	Classe FrontMdl	55
34	Classe LineChartMdl	56
35	Classe MapChartMdl	57
36	Classe TableMdl	58
37	Classe BarLineSvc	59
38	Classe ColorsSvc	60
39	Classe FirstConnectSvc	61
40	Classe FrontSvc	62
41	Classe MapSvc	63
42	Classe SocketsSvc	65
43	Classe TableSvc	67
44	Classe BarChartView	68
45	Classe Index	69
46	Classe LineChartView	69
47	Classe MapChartView	70
48	Classe TableView	70



49	Tracciamento Requisiti - Classi	75
50	Tracciamento Classi - Requisiti	80
51	Tracciamento test di unità-metodi	84
52	Tracciamento metodi-test di unità	90

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione di dettaglio del progetto Norris.

Il documento si basa sulla *Specifica Tecnica v2.0.0*. I programmatori si serviranno di tale documento per procedere con le attività di codifica.

1.2 Scopo del prodotto

Lo scopo del prodotto è la realizzazione di un *framework_G* per *Node.js_G*, compatibile con l'utilizzo standard dei *middleware_G* di *Express_G* in versione 4.x, per la realizzazione rapida di grafici aggiornabili in tempo reale.

1.3 Glossario

Per evitare ogni possibile ambiguità che potrebbe sorgere verrà allegato il *Glossario_ver5.0.0* dove verranno inseriti termini tecnici, acronimi, termini di dominio ed eventuali parole che potrebbero comportare delle incomprensioni o delle ambiguità nella lettura dei documenti. Per rendere la lettura più facile i termini verranno riportati in corsivo ed in pedice verrà posta una "G" maiuscola. (Esempio: *Android_G*).

1.4 Riferimenti

1.4.1 Normativi

- **Analisi dei Requisiti:** *AnalisiRequisiti_ver5.0.0*;
- **Norme di Progetto:** *NormeDiProgetto_ver3.0.0*;
- **Capitolato d'appalto C3 Norris:** Node Real-time Intelligence
<http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C3.pdf>;

1.4.2 Informativi

- Presentazione capitolato d'appalto: <http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C3ps.pdf>;
- Ingegneria del software - Ian Sommerville - 9a edizione (2011), Parte terza: Advance Software Engineering, Capitolo 18.3: Architectural patterns for distributed systems;
- Design Patterns - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - 1a edizione italiana (2008);

2 Standard di Progetto

2.1 Standard di progettazione architettuale

Gli standard di progettazione sono definiti nella *Specifica Tecnica v2.0.0*.

Per chiarezza, si evidenzia che in aggiunta al formalismo *UML_G 2.0* è stata utilizzata una notazione ad hoc per rappresentare il tipo di dato di una funzione **function(nomeParametro:tipo):tipodiritorno** rappresenta quindi il tipo di dato di una funzione che richiede i parametri **nomeParametro:tipo** e che ritorna un oggetto di **tipodiritorno**.

Inoltre, per definire la visibilità e la tipologia dei metodi sono state adottate le convenzioni *UML_G 2.0* che prevedono la sottolineatura nel caso il metodo sia statico e nessuna sottolineatura altrimenti. La visibilità invece, come per i campi dati, è identificata dai simboli + per la visibilità pubblica e - per quella privata, posizionati prima del nome del metodo.

2.2 Standard di documentazione del codice

Gli standard per la scrittura della documentazione del codice sono definiti nelle *Norme di Progetto v3.0.0*.

2.3 Standard di denominazione di entità e relazioni

Tutti gli elementi definiti come *package_G*, classi, metodi o attributi, devono avere denominazioni chiare ed esplicative. Il nome deve avere una lunghezza tale da non pregiudicarne la leggibilità e chiarezza. È preferibile utilizzare dei sostantivi per le entità e dei verbi per le relazioni. Le abbreviazioni sono ammesse se:

- immediatamente comprensibili;
- non ambigue;
- sufficientemente contestualizzate.

Le regole tipografiche relative ai nomi delle entità sono definite nelle *Norme di Progetto v3.0.0*.

2.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nelle *Norme di Progetto v3.0.0*. Si è scelto di fornire all'utente sviluppatore, che andrà ad utilizzare la libreria Norris, dei metodi con cui interagire incapsulati attraverso le classi del *package_G* Presentation-Layer. Per quanto riguarda il resto della libreria, non accessibile all'utente sviluppatore, si è invece scelto di utilizzare quasi esclusivamente metodi statici raggruppati logicamente tra loro in maniera equivalente a delle classi statiche.

2.5 Strumenti di lavoro

Per gli strumenti di lavoro da utilizzare durante la codifica e le procedure per il loro corretto funzionamento e coordinamento si rimanda al documento *Norme di Progetto v3.0.0*.

3 Descrizione Architettura

3.1 Metodo e formalismo di specifica

Di seguito viene riportata una parte relativa alla descrizione dell'architettura presa dal documento *SpecificaTecnica_ver3.0.0*.

Nell'esposizione dell'architettura di Norris si procederà con un approccio *top-down_G*, descrivendo l'architettura iniziando dal generale ed andando al particolare. Si procederà quindi alla descrizione dei *package_G* e dei componenti, per poi descrivere nel dettaglio le singole classi, specificando per ognuna una descrizione, il suo utilizzo e le relazioni in ingresso ed in uscita. Nel trattare i componenti, si chiarisce che sono da intendersi come *package_G* e i due termini verranno quindi usati come sinonimi.

Da notare, inoltre, che progettare il sistema con un'architettura ad oggetti classica non permette di rappresentare in modo naturale la gestione dinamica dei tipi e le caratteristiche tipiche degli stili di programmazione funzionali.

In certi casi, pertanto, è stato necessario introdurre interfacce e classi "fittizie", che non verranno codificate. Dato che questo introduce numerosi schematismi che appesantiscono i diagrammi e che non sono richiesti dal linguaggio di programmazione, si è cercato di limitarli soltanto ai casi in cui sono particolarmente utili.

3.2 Architettura generale

Il progetto è composto da tre parti: una componente *client_G* web, costituita dal browser degli utenti che visualizzeranno le pagine *front-end_G*, una componente *AndroidApp* che costituisce l'applicazione per *smartphone_G* *Android_G* e una componente web *server_G* costituita dalla sezione *back-end_G* di Norris.

3.2.1 Interfaccia REST-like

Per l'interfaccia della componente *back-end_G* di Norris si è scelto di utilizzare uno stile REST-like, ovvero basato sullo stile *REST_G*. I motivi che hanno spinto alla scelta di *REST_G* sono:

- Semplicità di utilizzo;
- Facile integrazione con i *framework_G* esistenti (*AngularJS_G* e *Express_G*);
- Indipendenza dal linguaggio di programmazione utilizzato.

REST_G utilizza il concetto di risorsa, ovvero un aggregato di dati con un nome (*URI_G*) e una rappresentazione, su cui è possibile invocare le operazioni *CRUD_G* tramite la seguente corrispondenza:

Risorsa	URI_G di un $template_G HTML_G$ di Norris es. <code>http://example.com/page/</code>	URI_G del contenuto di una pagina es. <code>http://example.com/page/raw</code>
GET	Ritorna un $template_G HTML_G$ nel quale vengono visualizzati i grafici attivi in una determinata pagina.	Ritorna i dati grezzi di una determinata pagina in formato JSON; questi dati contengono a loro volta tutti i dati dei grafici contenuti nella pagina stessa.
POST	Non usato	Non usato
PUT	Non usato	Non usato
DELETE	Non usato	Non usato

Tabella 2: Chiamate URI

L' URI_G di una particolare risorsa è arbitrario e a scelta dello sviluppatore che utilizzerà il $framework_G$ Norris. Una volta assegnato quello per il $template_G HTML_G$, il $framework_G$ si occuperà autonomamente di generare l' URI_G dei dati grezzi a partire da quello fornito dallo sviluppatore concatenando ad esso la stringa `/raw`.

Per il formato di rappresentazione dei dati è stato scelto $JSON_G$, in quanto si integra molto facilmente con i $framework_G$ utilizzati e con il linguaggio $JavaScript_G$, a differenza di XML o CSV che richiederebbero l'utilizzo di librerie specifiche.

3.2.2 Norris

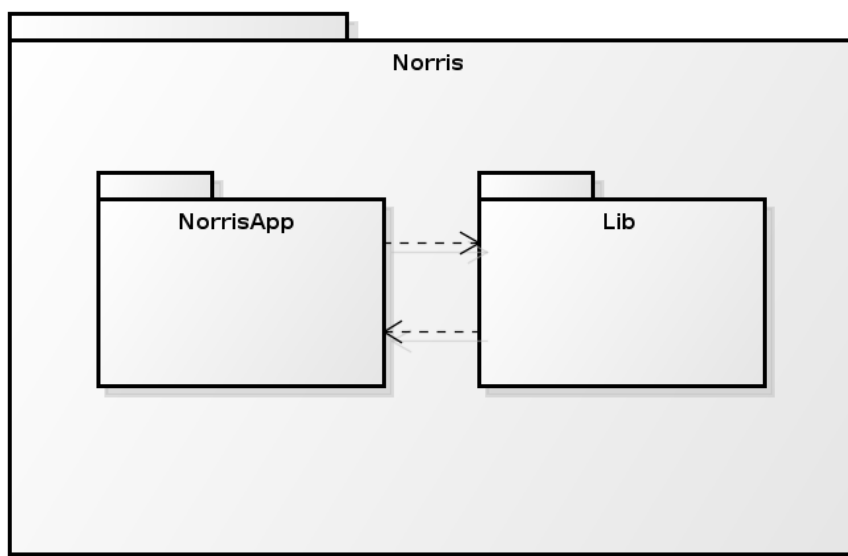


Figura 1: Diagramma dei *package_G* Norris

L'architettura del sistema è divisa in due *package_G* principali, a loro volta divisi in sotto *package_G*:

- **Lib**: si occuperà della gestione logica del *framework_G*;
- **NorrisApp**: si occuperà di fornire un'interfaccia visualizzabile ai *client_G* web.

NorrisApp eseguirà richieste get sulla Lib per ottenere i primi dati della pagina e dei grafici ad essa correlata, successivamente la Lib invierà tramite notifiche push gli aggiornamenti ai singoli grafici.

3.2.2.1 Lib

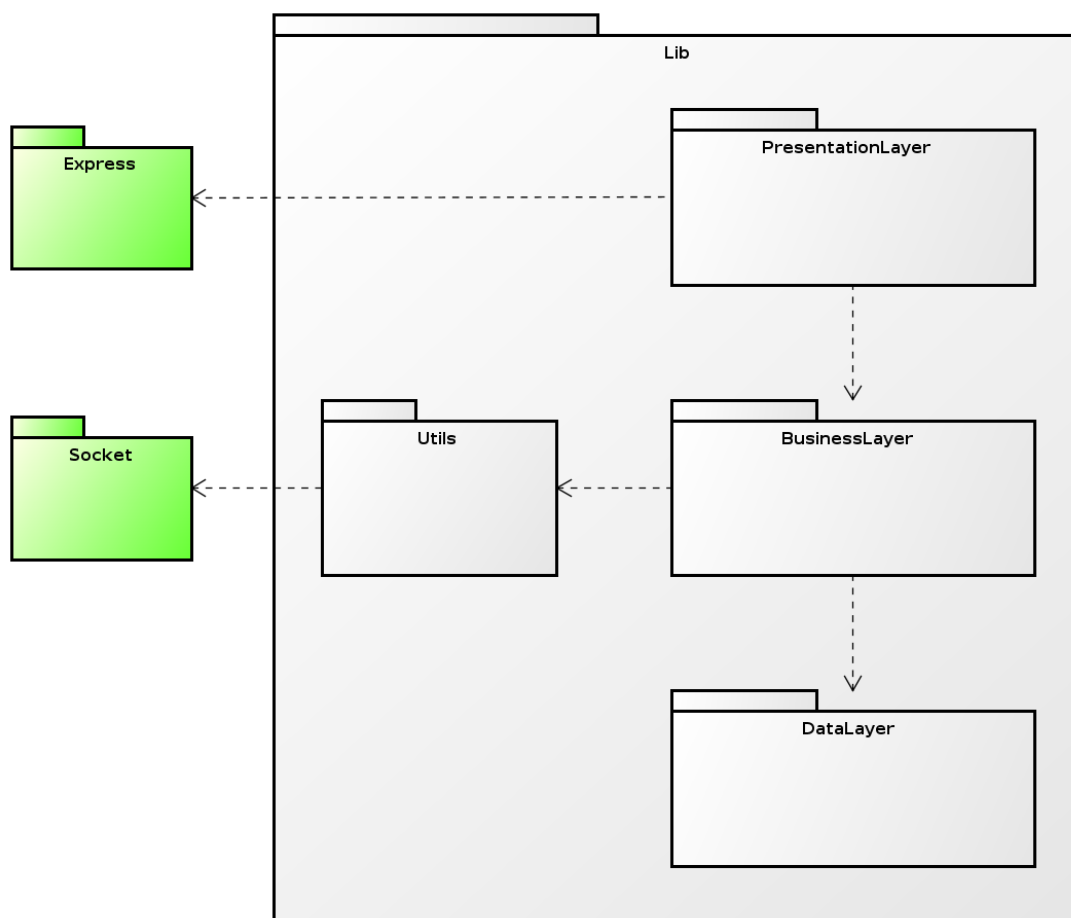


Figura 2: Diagramma del `package_G Lib`

Il `package_G Lib` contiene le componenti necessarie al `framework_G` per la gestione delle funzionalità sviluppatore, tra cui la creazione e la modifica di pagine e grafici, e le funzionalità di comunicazione con i `client_G`, quali browser web e applicativo *Android_G*. È strutturato secondo lo stile architetturale *Three Tier Architecture_G* che rappresenta la suddivisione logica delle componenti presenti al suo interno, ovvero: interfaccia verso l'esterno nel presentation layer, l'elaborazione nel business layer e i dati nel data layer ed un `package_G Utils` contenente delle classi di supporto.

- **Presentation Layer:** in questo layer sono presenti le classi che forniscono un punto d'accesso visibile dall'esterno alla libreria Norris. La classe **Norris** fornisce un indice degli oggetti che sarà possibile andare a creare e, richiamando le altre classi dello stesso *package_G*, permetterà l'accesso ai loro metodi specifici. Si occuperà inoltre di istanziare autonomamente un *subrouting_G* nel punto di *mount_G* designato dallo sviluppatore che gestirà le richieste dai *client_G* e l'invio degli aggiornamenti.

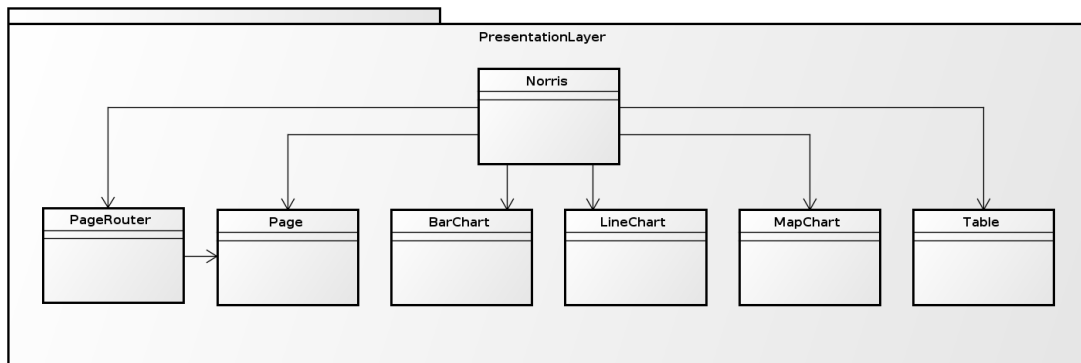


Figura 3: Il *package_G* Presentation Layer della libreria Norris.

- **Business Layer:** all'interno di questo *package_G* saranno presenti tutte le classi contenenti i metodi necessari allo svolgimento delle attività della libreria, qui verranno anche effettuati i controlli sugli errori e la logica necessaria all'invio dei nuovi dati ai *client_G*. Si è optato per creare una classe controller per ogni tipologia di oggetto in modo tale da separare le varie componenti, sarà inoltre presente una classe, **ActiveResourceController**, che consentirà di andare a ricercare risorse all'interno del **Data Layer**

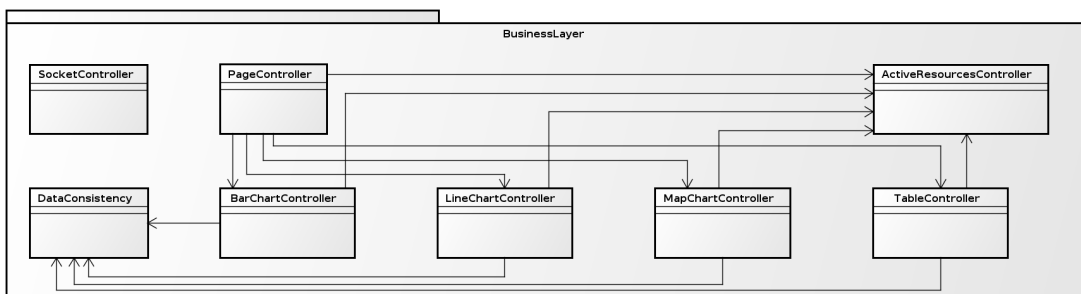


Figura 4: Il *package_G* Business Layer della libreria Norris.

- **Data Layer:** in questo layer sono presenti i modelli degli oggetti messi a disposizione dalla libreria Norris. Oltre ad essi sarà presente anche un oggetto **ActiveResource** che manterrà traccia di tutti gli oggetti creati all'interno dell'istanza di Norris.

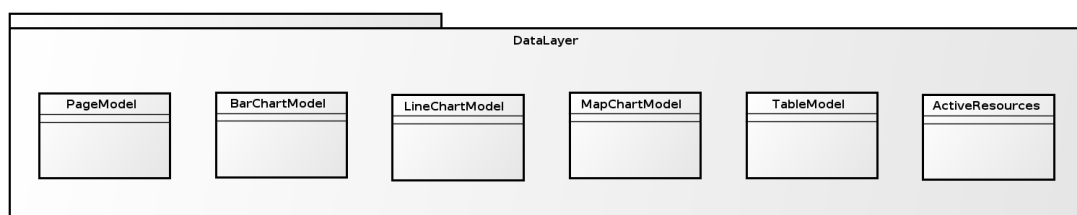


Figura 5: Il *package_G* Data Layer della libreria Norris.

- Utils:** questo *package_G* non appartiene allo stile architetturale *Three Tier Architecture_G* e stato inserito all'interno del *package_G* **Lib** allo scopo di contenere al suo interno alcune classi di supporto alle attività svolte dal **Business Layer**. La classe **NorrisError** si occuperà di restituire la tipologia di errore riscontrato, la classe **SocketService** conterrà la struttura necessaria per l'apertura di una connessione tramite *WebSocket_G* mentre le classi **ColorGenerator** e **ProgressiveID** saranno utilizzate in fase di creazione di una risorsa: la prima in caso di necessità di inserire un colore random all'interno dei grafici, l'altra produrrà un id univoco per ogni singolo oggetto istanziato.

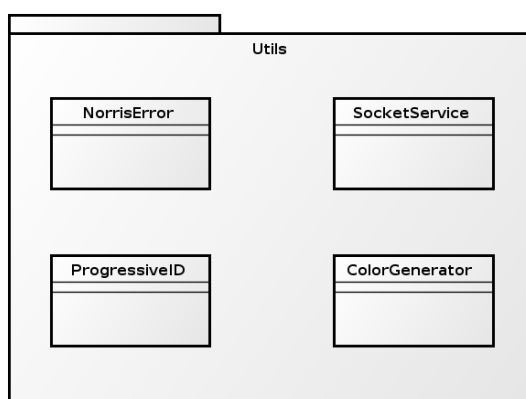


Figura 6: Il *package_G* Utils della libreria Norris.

3.2.2.2 NorrisApp

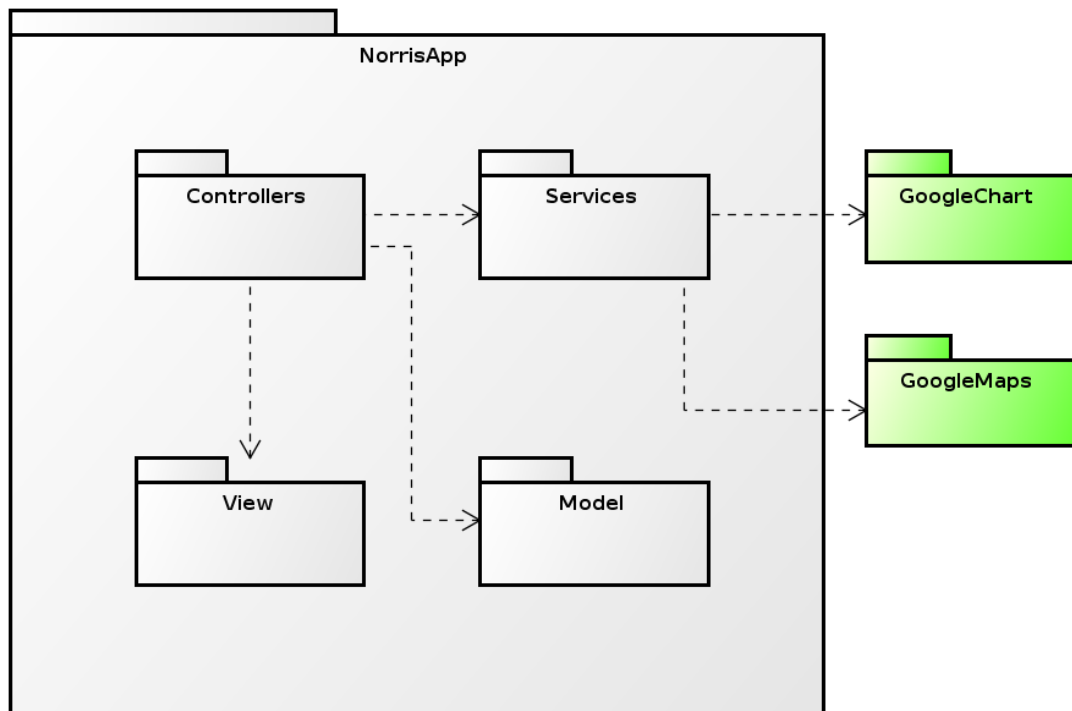


Figura 7: Diagramma dei $package_G$ di NorrisApp

Questa parte dell'architettura si occupa di contenere la sezione $front-end_G$. Comprende il sottosistema che viene eseguito nei browser degli utenti e che fornisce l'interfaccia grafica all'utente finale che visualizzerà le pagine create dallo sviluppatore, ed è suddivisa in quattro $package_G$ principali:

- **View:** $package_G$ comprendente le classi che costituiscono la view relativa al $design\ pattern_G\ MVC_G$ del componente $Front-end_G$. Ogni view rappresenta un tipo di grafico, che verrà popolato con i dati richiesti. Le componenti di questo $package_G$ non possiedono né campi dati né metodi associati, poichè la logica di business è gestita interamente dalle classi del $package_G$ Controllers, in particolare nella componente scope. Le classi di questo $package_G$ sono implementate tramite file $HTML_G$ e costituiscono i $template_G$ di pagina $AngularJS_G$.
- **Controller:** $package_G$ comprendente le classi che costituiscono i controller relativi al $design\ pattern_G\ MVC_G$ del componente $Front-end_G$. Ogni controller gestisce la logica applicativa riguardante un determinato tipo di grafico mediante le funzioni esposte dalle classi del $package_G$ Services corrispondente. Esso specifica anche le modalità di aggiornamento della view che presenta i contenuti del grafico all'utente, mediante l'uso della componente scope, che si mantiene aggiornata rispetto al modello dei dati.
- **Model:** $package_G$ che comprende le classi dei modelli relativi al $design\ pattern_G\ MVC_G$, dei dati utilizzati dal $front-end_G$. Servono a fornire al Controller e al Service i dati ricevuti dal $back-end_G$.

- **Services:** *package_G* comprendente le classi che descrivono i meccanismi con cui il *front-end_G* può interfacciarsi con il *back-end_G*, insieme alle operazioni sui vari tipi di grafico e sui dati ad essi associati. Permette di ricevere i dati da inserire nel Model.

4 Specifica classi del *back-end_G*

4.1 Componente Norris::Lib::BusinessLayer

4.1.1 Classe ActiveResourcesController

ActiveResourcesController
<div>+ <u>storeGraph(id:Integer, graph:Graph):void</u></div> <div>+ <u>retrieveGraph(id:Integer):Graph</u></div> <div>+ <u>retrievePage(id:Integer):Page</u></div> <div>+ <u>storePage(id:Integer, page:Page):void</u></div>

Tabella 3: Classe ActiveResourcesController

Descrizione

Questa classe comprende i metodi per gestire le pagine e i grafici attivi.

Utilizzo

Sarà utilizzata dalle classi del BusinessLayer per indicizzare e accedere alle risorse attive.

Attributi

Assenti.

Metodi

+ storeGraph(id:Integer, graph:Graph):void

Questo metodo è usato per salvare un riferimento ad un grafico quando questo viene creato.

- o **id:Integer**
Intero che contiene l'id di un grafico.
- o **graph:Graph**
Un oggetto di tipo BarChart, LineChart, MapChart o Table.

+ retrieveGraph(id:Integer):Graph

Ritorna l'oggetto Graph il cui id corrisponde all'id passato come parametro.

- o **id:Integer**
Intero che contiene l'id del grafico che verrà ritornato.

+ retrievePage(id:Integer):Page

Ritorna l'oggetto Page il cui id corrisponde all'id passato come parametro.

- **id:Integer**

Intero che contiene l'id dell'oggetto di tipo Page che verrà ritornato.

+ storePage(id:Integer,page:Page):void

Questo metodo viene usato per salvare il riferimento ad un oggetto di tipo Page quando questo viene creato.

- **id:Integer**

Intero che contiene l'id di un oggetto di tipo Page.

- **page:Page**

Oggetto di tipo Page il cui riferimento verrà salvato.

4.1.2 Classe BarChartController

BarChartController
<pre> + createBarChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object) :Integer + getChartInfo(graphID:Integer):Object + updateInPlace(graphID:Integer, label:String, set:Integer, newValue:Integer):void </pre>

Tabella 4: Classe BarChartController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Bar Chart_G*.

Utilizzo

Sarà utilizzata da *PresentationLayer::BarChart* per l'utilizzo delle *API_G* per la creazione e l'aggiornamento dei grafici di tipo *Bar Chart_G*.

Attributi

Assenti.

Metodi

+ createBarChart(title:String,xAxisName:String,yAxisName:String,labels:Array, data:Array,options:Object):Integer

Questo metodo crea un oggetto di tipo BarChart e ritorna un intero contenente l'id del grafico appena creato.

- **title:String**

Stringa che contiene il titolo del grafico.

- **xAxisName:String**
Stringa che contiene l'etichetta dell'asse delle X.
- **yAxisName:String**
Stringa che contiene l'etichetta dell'asse delle Y.
- **labels:Array**
Array che contiene le etichette dei set di dati.
- **data:Array**
Array che contiene i dati iniziali del grafico.
- **options:Object**
Parametro opzionale, oggetto *JSON_G* che contiene le opzioni aggiuntive da assegnare al grafico. Se non presente verranno impostate le opzioni di default.

+ **getChartInfo(graphID:Integer):Object**

Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

- **graphID:Integer**
Intero che contiene l'id del grafico di cui si desidera ottenere i dati.

+ **updateInPlace(graphID:Integer,label:String,set:Integer,newValue:Integer):void**

Aggiorna un dato del grafico con modalità in place.

- **graphID:Integer**
Intero che contiene l'id del grafico che si vuole aggiornare.
- **label:String**
Indica il nome della label del dato da cambiare.
- **set:Integer**
Intero che indica l'indice della serie a cui appartiene il dato da cambiare.
- **newValue:Integer**
Intero che contiene il nuovo valore da assegnare al dato.

4.1.3 Classe DataConsistency

DataConsistency
<ul style="list-style-type: none"> - <u>checkTemplate(opt:String, template:Object):void</u> - <u>checkOrientation(obj:String):void</u> - <u>checkLegendPosition(obj:String):void</u> - <u>checkMapLegendPosition(obj:String):void</u> - <u>checkMapZoom(obj:String):void</u> - <u>pathMode(obj:String):void</u> - <u>checkOrderBy(obj:Object, template:Object):void</u> - <u>checkInsertPosition(obj:String):void</u> - <u>checkDisplayedLines(obj:String):void</u> - <u>checkColors(obj:Array):void</u> - <u>checkHex(obj:String):void</u> - <u>checkColorArray(obj:Object, opt:String):void</u> - <u>checkColorMatrix(obj:Object, opt:String):void</u> - <u>checkSeries(obj:Array):void</u> - <u>checkValueType(obj:String):void</u> - <u>checkDecimals(obj:Integer):void</u> - <u>checkTableFormat(obj:Array):void</u> - <u>checkBounds(obj:Array, opt:String):void</u> - <u>checkAllColors(obj:Array, opt:String):void</u> + <u>jsonConsistencyCheck(obj:Object, template:Object):boolean</u> + <u>inPlaceTableOptionsConsistency(obj:Object, template:Object):boolean</u> + <u>streamTableOptionsConsistency(obj:Object, template:Object):boolean</u> + <u>labelConsistency(labels:Array):boolean</u> + <u>seriesConsistency(labels:Array, series:Array):boolean</u> + <u>checkUpdateHex(opt:String):Array</u> + <u>checkPropertyShowing(obj:boolean):void</u>

Tabella 5: Classe DataConsistency

Descrizione

Questa classe si occupa di effettuare i controlli sui dati .

Utilizzo

Sarà utilizzata allo scopo di verificare la forma e la consistenza dei dati forniti dallo sviluppatore.

I *template_G* che le varie funzioni ricevono in input hanno una struttura variabile e contengono tutte le proprietà necessarie affinché si possa verificare che i dati dello sviluppatore siano interamente o parzialmente coerenti con le richieste del sistema.

Ad esempio, un *template*_G delle opzioni di un tipo di grafico conterrà tutte le proprietà possibili per quel tipo di grafico, mentre un *template*_G di una proprietà interna conterrà solo le proprietà che interessano quest'ultima.

Attributi

Assenti.

Metodi

- `checkTemplate(opt:String,template:Object):void`

Controlla che la proprietà `opt` sia presente nel *template* delle opzioni.

- **`opt:String`**
Nome della proprietà da controllare.
- **`template:Object`**
*Template*_G delle opzioni in cui controllare la presenza di una proprietà.

- `checkOrientation(obj:String):void`

Controlla che l'opzione 'orientation' abbia il valore vertical o horizontal.

- **`obj:String`**
Valore della proprietà da controllare.

- `checkLegendPosition(obj:String):void`

Controlla che la proprietà 'legendPosition' dei grafici di tipo *Bar Chart*_G e *Line Chart*_G abbia il valore top, bottom, left o right.

- **`obj:String`**
Valore della proprietà da controllare.

- `checkMapLegendPosition(obj:String):void`

Controlla che la proprietà 'mapLegendPosition' dei grafici di tipo *Map Chart*_G abbia il valore top-right, top-left, bottom-right o bottom-left.

- **`obj:String`**
Valore della proprietà da controllare.

- `checkMapZoom(obj:String):void`

Controlla che la proprietà 'zoom' di un grafico di tipo *Map Chart*_G sia un intero compreso tra 0 e 19.

- **`obj:String`**
Valore della proprietà da controllare.

- `pathMode(obj:String):void`

Controlla che la proprietà 'pathMode' abbia il valore walking, driving, bicycling o transit.

- **`obj:String`**
Valore della proprietà da controllare.

- `checkOrderBy(obj:Object,template:Object):void`

Controlla che la proprietà 'orderBy' dei grafici di tipo *Table*_G sia un oggetto in formato JSON, contenente le proprietà 'column' e 'order'.

- **obj:Object**
Contiene le proprietà e i valori da controllare.
- **template:Object**
Template_G delle opzioni in cui controllare la presenza di una proprietà.

- **checkInsertPosition(obj:String):void**

Controlla che la proprietà 'insertPosition' dei grafici di tipo *Table_G* abbia il valore top o bottom.

- **obj:String**
Valore della proprietà da controllare.

- **checkDisplayedLines(obj:String):void**

Controlla che la proprietà 'displayedLines' dei grafici di tipo *Table_G* sia un intero maggiore di 0.

- **obj:String**
Valore della proprietà da controllare.

- **checkColors(obj:Array):void**

Controlla che la proprietà 'colors' sia un array contenente valori esadecimali validi.

- **obj:Array**
Array contenente i colori da controllare.

- **checkHex(obj:String):void**

Controlla che il colore passato sia un valore esadecimale valido.

- **obj:String**
Valore della proprietà da controllare.

- **checkColorArray(obj:Object,opt:String):void**

Controlla che le proprietà 'colorColumn', 'colorRow', 'colorColumnFont', 'color-RowFont' dei grafici di tipo *Table_G* siano array di array, nei quali ogni array interno abbia due elementi: l'indice di riga o colonna, e il valore esadecimale corrispondente al colore.

- **obj:Object**
Contiene le proprietà da controllare.
- **opt:String**
Identifica la proprietà da controllare.

- **checkColorMatrix(obj:Object,opt:String):void**

Controlla che le proprietà 'colorFont' e 'colorCell' dei grafici di tipo *Table_G* siano array di array, nei quali ogni array interno abbia tre elementi: l'indice di riga, l'indice di colonna e il valore esadecimale corrispondente al colore.

- **obj:Object**
Contiene le proprietà da controllare.
- **opt:String**
Identifica la proprietà da controllare.

- **checkSeries(obj:Array):void**

Controlla che la proprietà 'series' sia un array di stringhe.

- **obj:Array**
Proprietà da controllare.

- **checkValueType(obj:String):void**

Controlla che la proprietà 'valueType' dei grafici di tipo *Bar Chart_G* e *Line Chart_G* abbia il valore euro, dollars o pounds.

- **obj:String**
Valore della proprietà da controllare.

- **checkDecimals(obj:Integer):void**

Controlla che la proprietà 'decimals' dei grafici di tipo *Bar Chart_G* e *Line Chart_G* sia un intero compreso tra 0 e 6.

- **obj:Integer**
Valore della proprietà da controllare.

- **checkTableFormat(obj:Array):void**

Controlla che la proprietà 'format' dei grafici di tipo *Table_G* sia un oggetto in formato JSON, contenente le proprietà 'column', 'valueType' e 'decimals'.

- **obj:Array**
Contiene le proprietà da controllare.

- **checkBounds(obj:Array,opt:String):void**

Controlla che le proprietà 'labelsLimit' e 'rowsLimit' rispettivamente dei grafici di tipo *Line Chart_G* e *Table_G* siano interi maggiori di 0.

- **obj:Array**
Contiene le opzioni da controllare.
- **opt:String**
Contiene il nome dell'opzione da controllare.

- **checkAllColors(obj:Array,opt:String):void**

Richiama tutti gli altri metodi che controllano le opzioni dei colori.

- **obj:Array**
Contiene le proprietà da controllare.
- **opt:String**
Contiene il nome della proprietà da controllare.

+ **jsonConsistencyCheck(obj:Object,template:Object):boolean**

Controlla che il parametro 'options' dei grafici abbia una forma corretta, richiama gli altri metodi di DataConsistency in base al tipo di grafico in esame.

- **obj:Object**
Oggetto che contiene tutte le opzioni aggiuntive del grafico.
- **template:Object**
Oggetto che contiene un *template_G* di opzioni con cui fare il confronto.

+ **inPlaceTableOptionsConsistency(obj:Object,template:Object):boolean**

Controlla che il parametro 'options' dell'update con modalità in place dei grafici di tipo *Table_G* sia un oggetto in formato JSON, contenente le proprietà 'colorFont' e 'colorCell'.

- **obj:Object**
Oggetto che contiene tutte le opzioni aggiuntive dell'update con modalità in place dei grafici di tipo *Table_G*.
- **template:Object**
Oggetto che contiene un *template_G* di opzioni con cui fare il confronto.

+ streamTableOptionsConsistency(obj:Object,template:Object):boolean

Controlla che il parametro 'options' dell'update con modalità stream dei grafici di tipo *Table_G* sia un oggetto in formato JSON, contenente le proprietà 'colorFont', 'colorCell', 'colorRow' e 'colorRowFont'.

- **obj:Object**
Oggetto che contiene tutte le opzioni aggiuntive dell'update con modalità stream dei grafici di tipo *Table_G*.
- **template:Object**
Oggetto che contiene un *template_G* di opzioni con cui fare il confronto.

+ labelConsistency(labels:Array):boolean

Il metodo controlla la correttezza del label.

- **labels:Array**
Array contenente i label.

+ seriesConsistency(labels:Array,series:Array):boolean

Controlla se le dimensioni di labels coincide con quella di series.

- **labels:Array**
Array contenente i label.
- **series:Array**
Array contenente i nomi delle serie dei dati.

+ checkUpdateHex(opt:String):Array

Controlla che il colore passato sia un valore esadecimale valido.

- **opt:String**
Nome della proprietà da controllare.

+ checkPropertyShowing(obj:boolean):void

Questo metodo controlla che il parametro passato sia un booleano. In caso negativo solleva un errore.

- **obj:boolean**
Parametro da controllare.

4.1.4 Classe LineChartController

LineChartController
<pre> + createLineChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object) :Integer + updateStream(graphID:Integer, newLabel:String, newValues:Array):void + getChartInfo(graphID:Integer):Object + updateInPlace(graphID:Integer, label:String, set:String, newValue:Array):void </pre>

Tabella 6: Classe LineChartController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Line Chart_G*.

Utilizzo

Sarà utilizzata da *PresentationLayer::LineChart* per l'utilizzo delle *API_G* per la creazione e l'aggiornamento dei grafici di tipo *Line Chart_G*.

Attributi

Assenti.

Metodi

+ createLineChart(title:String,xAxisName:String,yAxisName:String,labels:Array, data:Array,options:Object):Integer

Questo metodo crea un oggetto di tipo *LineChart* e ritorna un intero contenente l'id del grafico appena creato.

- **title:String**
Stringa che contiene il titolo del grafico.
- **xAxisName:String**
Stringa che contiene l'etichetta dell'asse delle X.
- **yAxisName:String**
Stringa che contiene l'etichetta dell'asse delle Y.
- **labels:Array**
Array che contiene le etichette dei set di dati.
- **data:Array**
Array che contiene i dati iniziali del grafico.

- **options:Object**

Parametro opzionale, oggetto *JSON_G* che contiene le opzioni aggiuntive da assegnare al grafico. Se non presente verranno impostate le opzioni di default.

+ **updateStream(graphID:Integer,newLabel:String,newValues:Array):void**

Metodo che aggiunge una nuova label e un valore per ogni set di dati in modalità stream.

- **graphID:Integer**

Intero che contiene l'id del grafico che si desidera aggiornare.

- **newLabel:String**

Stringa che contiene il nome della label che verrà aggiunta.

- **newValues:Array**

Array che contiene i nuovi valori per ogni set di dati.

+ **getChartInfo(graphID:Integer):Object**

Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

- **graphID:Integer**

Intero che contiene l'id del grafico di cui si desidera ottenere i dati.

+ **updateInPlace(graphID:Integer,label:String,set:String,newValue:Array):void**

Metodo che aggiorna un dato del grafico con modalità in place.

- **graphID:Integer**

Intero che contiene l'id del grafico che si vuole aggiornare.

- **label:String**

Indica il nome della label del dato da cambiare.

- **set:String**

Indica l'indice della serie a cui appartiene il dato da cambiare.

- **newValue:Array**

Array che contiene i nuovi valori del dato.

4.1.5 Classe MapChartController

MapChartController
<pre> + createMapChart(title:String, paths:Array, points:Array, centerLatitude:Float, centerLongitude:Float, options:Object):Integer + updateMovie(graphID:Integer, newPoints:Array):void + getChartInfo(graphID:Integer):Object + updateInPlace(graphID:Integer, label:Array, latitude:Float, longitude:Float):void - pointsConsistency(points:Array):boolean </pre>

Tabella 7: Classe MapChartController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Map Chart_G*.

Utilizzo

Sarà utilizzata da `PresentationLayer::MapChart` per l'utilizzo delle *API_G* per la creazione e l'aggiornamento dei grafici di tipo *Map Chart_G*.

Attributi

Assenti.

Metodi

`+ createMapChart(title:String, paths:Array, points:Array, centerLatitude:Float, centerLongitude:Float, options:Object):Integer`

Costruttore della classe MapChartController.

- **title:String**
Stringa che contiene il titolo del grafico.
- **paths:Array**
Array che contiene i percorsi da tracciare sulla mappa.
- **points:Array**
Array che contiene le coordinate dei punti da visualizzare sulla mappa.
- **centerLatitude:Float**
Latitudine del punto centrale della mappa.
- **centerLongitude:Float**
Longitudine del punto centrale della mappa.
- **options:Object**
Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.

+ updateMovie(graphID:Integer,newPoints:Array):void

Aggiorna i punti della mappa con modalità movie.

- **graphID:Integer**
Intero che contiene l'id del grafico che si vuole aggiornare.
- **newPoints:Array**
Array contenente i nuovi punti.

+ getChartInfo(graphID:Integer):Object

Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

- **graphID:Integer**
Intero che contiene l'id del grafico di cui si desidera ottenere i dati.

+ updateInPlace(graphID:Integer,label:Array,latitude:Float,longitude:Float):void

Aggiorna un punto della mappa con modalità in place.

- **graphID:Integer**
Intero che contiene l'id del grafico che si vuole aggiornare.
- **label:Array**
Array contenente i label.
- **latitude:Float**
Nuova latitudine del punto da modificare.
- **longitude:Float**
Nuova longitudine del punto da modificare.

- pointsConsistency(points:Array):boolean

Controlla che le etichette non siano contengano duplicati.

- **points:Array**
Array di punti da controllare.

4.1.6 Classe PageController

PageController
<u>+ createPage(title:String, options:Object):Integer</u> <u>+ addGraphToPage(pageID:Integer, graph:Graph):void</u> <u>+ getPageInfo(pageID:Integer):Object</u> <u>- createPageOptions(options:Object):Object</u>

Tabella 8: Classe PageController

Descrizione

Questa classe comprende i metodi per la creazione delle pagine.

Utilizzo

Sarà utilizzata da `PresentationLayer::Page` per l'utilizzo delle *API_G* per la creazione delle pagine.

Attributi

Assenti.

Metodi

+ createPage(title:String,options:Object):Integer

Crea un oggetto di tipo `Page` e ritorna l'id assegnato all'oggetto appena creato.

- **title:String**
Stringa che contiene il titolo da assegnare all'oggetto `Page` che si va a creare.
- **options:Object**
Contiene le opzioni aggiuntive da assegnare alla pagina.

+ addGraphToPage(pageID:Integer,graph:Graph):void

Questo metodo aggiunge un grafico ad una pagina.

- **pageID:Integer**
Intero che contiene l'id dell'oggetto `Page` a cui si vuole aggiungere il grafico.
- **graph:Graph**
Il grafico che si vuole aggiungere alla pagina.

+ getPageInfo(pageID:Integer):Object

Metodo che ritorna tutte le informazioni sulla pagina: attributi e grafici contenuti.

- **pageID:Integer**
Intero che contiene l'id dell'oggetto `Page` di cui si vogliono le informazioni.

- createPageOptions(options:Object):Object

Questo metodo controlla che le opzioni fornite per la creazione di una pagina siano corrette, e ritorna delle opzioni completate con valori di default nel caso le originali fossero mancanti completamente o parzialmente.

- **options:Object**
Contiene le opzioni aggiuntive da controllare.

4.1.7 Classe SocketController

SocketController
<u>+ setSocket(nsp:Object):void</u> <u>+ sendUpdate(room:String, info:Object):void</u> <u>+ socketNamespace():String</u>

Tabella 9: Classe `SocketController`

Descrizione

Questa classe si occupa di gestire le interazioni con la componente *socket.io_G* utilizzata all'interno della libreria Norris nella classe *SocketService*. Per un per la spiegazione di dettaglio delle operazioni interne, si rimanda alla descrizione dei metodi della classe *SocketService*.

Utilizzo

La classe verrà utilizzata per assegnare e predisporre correttamente la risorsa *socket.io_G* della libreria e per la gestione degli aggiornamenti verso i vari *client_G*.

Attributi

Assenti.

Metodi

+ *setSocket(nsp:Object):void*

Questo metodo imposta il *namespace_G* di *Socket.io_G* che verrà utilizzato dalla libreria Norris e avvia il gestore di connessioni tra il *server_G* e il *client_G*.

- ***nsp:Object***

Il *namespace_G* di *Socket.io_G* che verrà utilizzato dalla libreria Norris.

+ *sendUpdate(room:String,info:Object):void*

Questo metodo invia un *JSON_G* dal *server_G* al *client_G*.

- ***room:String***

Contiene l'id della room.

- ***info:Object***

I dati che vengono inviati al *client_G* per l'aggiornamento.

+ *socketNamespace():String*

Questo metodo ritorna il *namespace_G* di *Socket.io_G* correntemente utilizzato da Norris.

4.1.8 Classe TableController

TableController
<pre> + createTable(title:String, header:Array, data:Array, options:Object):Integer + getChartInfo(graphID:Integer):Object + updateStream(graphID:Integer, data:Array, options:Object) : void + updateInPlace(graphID:Integer, row:Integer, column:Integer, newValue:Array, options:Object):void - buildTempl():Object - fillDefaultOpts(tableOptions:Object):void - fillDevOpts(options:Object, tableOptions:Object, columns:Integer):void - fillDefaultColorOpts(tableOptions:Object, rows:Integer, columns:Integer):void - fillDevBgColorOpts(options:Object, tableOptions:Object, rows:Integer, columns:Integer):void - fillDevFontColorOpts(options:Object, tableOptions:Object, rows:Integer, columns:Integer):void - fillDevUpdateOpts(options:Object, columns:Integer, fontColor:String, cellColor:String, rowColor:String, rowFontColor:String):void - dataModelPush(_tableModel:Table, data:Array, fontColor:String, cellColor:String, rowColor:String, rowFontColor:String):void - fillClientColors(_tableModel:Table, columns:Integer) :Array - removeRow(_tableModel:Table):void - getColors(graphID:Integer, rows:Integer, columns:Integer) :Array </pre>

Tabella 10: Classe TableController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Table_G*.

Utilizzo

Sarà utilizzata da *PresentationLayer::Table* per l'utilizzo delle *API_G* per la creazione e l'aggiornamento dei grafici di tipo *Table_G*.

Attributi

Assenti.

Metodi

+ `createTable(title:String,header:Array,data:Array,options:Object):Integer`

Questo metodo crea un oggetto di tipo `Table` e ritorna un intero contenente l'id della tabella appena creata.

- **`title:String`**
Stringa che contiene il titolo della tabella.
- **`header:Array`**
Array che contiene gli header delle colonne della tabella.
- **`data:Array`**
Array che contiene i dati iniziali della tabella.
- **`options:Object`**
Parametro opzionale, oggetto *JSON_G* che contiene le opzioni aggiuntive da assegnare alla tabella. Se non presente verranno impostate le opzioni di default.

+ `getChartInfo(graphID:Integer):Object`

Metodo che ritorna tutte le informazioni sulla tabella: attributi, dati ed opzioni.

- **`graphID:Integer`**
Intero che contiene l'id del grafico di cui si vogliono le informazioni.

+ `updateStream(graphID:Integer,data:Array,options:Object):void`

Aggiunge una nuova riga alla tabella in modalità stream.

- **`graphID:Integer`**
Intero che contiene l'id della tabella che si vuole aggiornare.
- **`data:Array`**
Array che contiene tutti i dati della nuova riga che verrà aggiunta alla tabella.
- **`options:Object`**
Opzioni aggiuntive.

+ `updateInPlace(graphID:Integer,row:Integer,column:Integer,newValue:Array,options:Object):void`

Aggiorna una cella della tabella in modalità in place.

- **`graphID:Integer`**
Intero che contiene l'id del grafico che si vuole aggiornare.
- **`row:Integer`**
Intero che indica il numero di riga della cella da aggiornare.
- **`column:Integer`**
Intero che indica il numero di colonna della cella da aggiornare.
- **`newValue:Array`**
Array che contiene i nuovi valori della cella da aggiornare.
- **`options:Object`**
Opzioni aggiuntive.

- **buildTempl():Object**

Costruisce un *template_G* per le opzioni aggiuntive dei grafici di tipo *Table_G*.

- **fillDefaultOpts(tableOptions:Object):void**

Riempie il *template_G* passato come parametro con le opzioni di default.

- **tableOptions:Object**

Template_G delle opzioni aggiuntive di un grafico di tipo *Table_G*.

- **fillDevOpts(options:Object,tableOptions:Object,columns:Integer):void**

Riempie il *template_G* di opzioni aggiuntive di un grafico di tipo *Table_G* con le opzioni impostate dallo sviluppatore.

- **options:Object**

Opzioni impostate dallo sviluppatore.

- **tableOptions:Object**

Template_G di opzioni aggiuntive di un grafico di tipo *Table_G* da riempire.

- **columns:Integer**

Numero di colonne nella tabella.

- **fillDefaultColorOpts(tableOptions:Object,rows:Integer,columns:Integer):void**

Riempie 'tableOptions' con le opzioni di colore di default.

- **tableOptions:Object**

Template_G di opzioni aggiuntive di un grafico di tipo *Table_G* da riempire.

- **rows:Integer**

Numero delle righe della tabella.

- **columns:Integer**

Numero di colonne della tabella.

- **fillDevBgColorOpts(options:Object,tableOptions:Object,rows:Integer,columns:Integer):void**

Riempie 'tableOptions' con le opzioni 'cellBackgroundColor' impostate dallo sviluppatore.

- **options:Object**

Opzioni impostate dallo sviluppatore.

- **tableOptions:Object**

Template_G di opzioni aggiuntive di un grafico di tipo *Table_G* da riempire.

- **rows:Integer**

Numero di righe della tabella.

- **columns:Integer**

Numero di colonne della tabella.

- **fillDevFontColorOpts(options:Object,tableOptions:Object,rows:Integer,columns:Integer):void**

Riempie 'tableOptions' con le opzioni di colore del font impostate dallo sviluppatore.

- **options:Object**

Opzioni impostate dallo sviluppatore.

- **tableOptions:Object**
Template di opzioni aggiuntive di un grafico di tipo *Table_G* da riempire.
- **rows:Integer**
Numero di righe della tabella.
- **columns:Integer**
Numero di colonne della tabella.

- fillDevUpdateOpts(options:Object,columns:Integer,fontColor:String,cellColor:String,rowColor:String,rowFontColor:String):void

Riempie gli array di colori con le opzioni impostate dallo sviluppatore.

- **options:Object**
Opzioni da riempire.
- **columns:Integer**
Numero di colonne della tabella.
- **fontColor:String**
Colore del font.
- **cellColor:String**
Colore di una cella.
- **rowColor:String**
Colore delle celle di una riga della tabella.
- **rowFontColor:String**
Colore del font di una riga della tabella.

- dataModelPush(_tableModel:Table,data:Array,fontColor:String,cellColor:String,rowColor:String,rowFontColor:String):void

Inserisce i nuovi dati nel modello di un grafico di tipo *Table_G*.

- **_tableModel:Table**
Oggetto di tipo Table.
- **data:Array**
Valori della nuova riga da inserire nella tabella.
- **fontColor:String**
Opzione 'fontColor' della nuova riga da inserire nella tabella.
- **cellColor:String**
Opzione 'cellColor' della nuova riga da inserire nella tabella.
- **rowColor:String**
Opzione 'rowColor' della nuova riga da inserire nella tabella.
- **rowFontColor:String**
Opzione 'rowFontColor' della nuova riga da inserire nella tabella.

- fillClientColors(_tableModel:Table,columns:Integer):Array

Costruisce l'array dei colori per il *client_G*.

- **_tableModel:Table**
Oggetto di tipo Table.
- **columns:Integer**
Numero di colonne della tabella.

- removeRow(_tableModel:Table):void

Rimuove una riga dalla tabella.

- **_tableModel:Table**
Oggetto di tipo Table.

- getColors(graphID:Integer,rows:Integer,columns:Integer):Array

Crea un array di colori.

- **graphID:Integer**
ID del grafico.
- **rows:Integer**
Numero di righe della tabella.
- **columns:Integer**
Numero di colonne della tabella.

4.2 Componente Norris::Lib::DataLayer

4.2.1 Classe ActiveResources

ActiveResources	
-	graphs:Array
-	pages:Array
+	ActiveResources():Object

Tabella 11: Classe ActiveResources

Descrizione

Questa classe rappresenta le risorse attive, siano esse pagine o grafici.

Utilizzo

Sarà utilizzata da BusinessLayer::ActiveResourcesController per indicizzare e accedere alle risorse attive.

Attributi

- **graphs:Array**
Contiene i riferimenti ai grafici attivi in una determinata istanza di Norris;
- **pages:Array**
Contiene i riferimenti alle pagine attive in una determinata istanza di Norris.

Metodi

+ ActiveResources():Object

Restituisce i riferimenti ai due array contenenti i riferimenti alle risorse attive in una determinata istanza di Norris.

4.2.2 Classe BarChartModel

BarChartModel
<ul style="list-style-type: none"> - <code>_id:Integer</code> - <code>_title:String</code> - <code>_xAxisName:String</code> - <code>_yAxisName:String</code> - <code>_labels:Array</code> - <code>_data:Array</code> - <code>_orientation:String</code> - <code>_showGrid:String</code> - <code>_showLegend:String</code> - <code>_legendPosition:String</code> - <code>_colors:Array</code> - <code>_valueType:String</code> - <code>_decimals:Integer</code> - <code>_series:Array</code>
+ <code>BarChartModel(id:Integer, title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object) :BarChartModel</code>

Tabella 12: Classe BarChartModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Bar Chart_G*.

Utilizzo

Sarà utilizzata da `BusinessLayer::BarChartController` per la creazione di grafici di tipo *Bar Chart_G*.

Attributi

- `_id:Integer`

Codice ID univoco che identifica l'oggetto ;

- `_title:String`

Titolo del grafico;

- `_xAxisName:String`

Nome dell'asse delle ascisse;

- `_yAxisName:String`

Nome dell'asse delle ordinate;

- `_labels:Array`

Array che contiene i valori che verranno riportati nell'asse delle ascisse;

- `_data:Array`

Array che contiene le serie di valori che verranno visualizzati nell'asse delle ordinate;

- `_orientation:String`

Indica l'orientamento delle barre del grafico, può assumere il valore vertical o horizontal ;

- `_showGrid:String`

Indica se la griglia del grafico è visibile o meno, può assumere il valore shown o hidden;

- `_showLegend:String`

Indica se la legenda del grafico è visibile o meno, può assumere il valore shown o hidden;

- `_legendPosition:String`

Indica la posizione della legenda del grafico, può assumere il valore right, left, bottom o top;

- `_colors:Array`

Indica i colori delle serie di dati del grafico;

- `_valueType:String`

Indica se i valori del grafico rappresentano un tipo di valuta, può assumere valore euro, dollars, pounds o null;

- `_decimals:Integer`

Indica quante cifre decimali vengono visualizzate per ogni valore, può assumere valore da 0 a 6;

- `_series:Array`

Indica i nomi delle serie di dati presenti nel grafico.

Metodi

+ `BarChartModel(id:Integer,title:String,xAxisName:String,yAxisName:String,labels:Array,data:Array,options:Object):BarChartModel`

Costruttore della classe BarChartModel.

- o **`id:Integer`**

Intero che contiene l'id del grafico.

- o **`title:String`**

Stringa che contiene il titolo del grafico.

- o **`xAxisName:String`**

Stringa che contiene l'etichetta dell'asse delle X.

- o **`yAxisName:String`**

Stringa che contiene l'etichetta dell'asse delle Y.

- o **`labels:Array`**

Array che contiene le etichette dei set di dati.

- o **`data:Array`**

Array che contiene i dati iniziali del grafico.

- **options:Object**
Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.

4.2.3 Classe LineChartModel

LineChartModel
<ul style="list-style-type: none"> - <code>_id:Integer</code> - <code>_title:String</code> - <code>_xAxisName:String</code> - <code>_yAxisName:String</code> - <code>_labels:Array</code> - <code>_data:Array</code> - <code>_showGrid:String</code> - <code>_showLegend:String</code> - <code>_legendPosition:String</code> - <code>_colors:Array</code> - <code>_valueType:String</code> - <code>_decimals:Integer</code> - <code>_labelsLimit:Integer</code> - <code>_series:Array</code>
+ <code>LineChartModel(id:Integer, title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object):LineChartModel</code>

Tabella 13: Classe LineChartModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Line Chart_G*.

Utilizzo

Sarà utilizzata da `BusinessLayer::LineChartController` per la creazione e la modifica dei grafici di tipo *Line Chart_G*.

Attributi

- **`_id:Integer`**
Codice ID univoco che identifica l'oggetto;
- **`_title:String`**
Titolo del grafico;
- **`_xAxisName:String`**
Nome dell'asse delle ascisse;
- **`_yAxisName:String`**
Nome dell'asse delle ordinate;

- `_labels:Array`

Array che contiene i valori che verranno riportati nell'asse delle ascisse;

- `_data:Array`

Array che contiene le serie di valori che verranno visualizzati nell'asse delle ordinate;

- `_showGrid:String`

Indica se la griglia del grafico è visibile o meno, può assumere il valore `shown` o `hidden`;

- `_showLegend:String`

Indica se la legenda del grafico è visibile o meno, può assumere il valore `shown` o `hidden`;

- `_legendPosition:String`

Indica la posizione della legenda del grafico, può assumere il valore `right`, `left`, `bottom` o `top`;

- `_colors:Array`

Indica i colori delle serie di dati del grafico;

- `_valueType:String`

Indica se i valori del grafico rappresentano un tipo di valuta, può assumere valore `euro`, `dollars`, `pounds` o `null`;

- `_decimals:Integer`

Indica quante cifre decimali vengono visualizzate per ogni valore, può assumere valore da 0 a 6;

- `_labelsLimit:Integer`

Indica la lunghezza massima delle serie di valori, può assumere valore maggiore di zero;

- `_series:Array`

Indica i nomi delle serie di dati presenti nel grafico.

Metodi

```
+ LineChartModel(id:Integer,title:String,xAxisName:String,yAxisName:String,
labels:Array,data:Array,options:Object):LineChartModel
```

Costruttore della classe LineChartModel.

- o **`id:Integer`**

Intero che contiene l'id del grafico.

- o **`title:String`**

Stringa che contiene il titolo del grafico.

- o **`xAxisName:String`**

Stringa che contiene l'etichetta dell'asse delle X.

- o **`yAxisName:String`**

Stringa che contiene l'etichetta dell'asse delle Y.

- o **`labels:Array`**

Array che contiene le etichette dei set di dati.

- **data:Array**
Array che contiene i dati iniziali del grafico.
- **options:Object**
Oggetto *JSON_G* che contiene le opzioni aggiuntive da assegnare al grafico.

4.2.4 Classe MapChartModel

MapChartModel
<ul style="list-style-type: none"> - <code>_id:Integer</code> - <code>_title:String</code> - <code>_points:Array</code> - <code>_paths:Array</code> - <code>_center:Object</code> - <code>_colors:Array</code> - <code>_zoom:Integer</code> - <code>_showLegend:String</code> - <code>_mapLegendPosition:String</code> - <code>_pathName:Array</code> - <code>_pathMode:String</code>
+ <code>MapChartModel(id:Integer, title:String, paths:Array, points:Array, center:Object, options:Object):MapChartModel</code>

Tabella 14: Classe MapChartModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Map Chart_G*.

Utilizzo

Sarà utilizzata da `BusinessLayer::MapChartController` per la creazione e la modifica dei grafici di tipo *Map Chart_G*.

Attributi

- **`_id:Integer`**
Codice ID univoco che identifica l'oggetto;
- **`_title:String`**
Titolo del grafico;
- **`_points:Array`**
Array che contiene i punti che verranno rappresentati sul grafico;
- **`_paths:Array`**
Array che contiene i percorsi che verranno rappresentati sul grafico;

- `_center:Object`

*JSON*_G che contiene i valori di latitudine e longitudine del punto centrale della mappa;

- `_colors:Array`

Indica i colori dei percorsi del grafico;

- `_zoom:Integer`

Indica il livello di zoom della mappa e può assumere un valore compreso tra 0 e 19;

- `_showLegend:String`

Indica se la legenda del grafico è visibile o meno, può assumere il valore `shown` o `hidden`;

- `_mapLegendPosition:String`

Indica la posizione della legenda del grafico, può assumere il valore `top-right`, `top-left`, `bottom-right` o `bottom-left`;

- `_pathName:Array`

Array che contiene i nomi dei percorsi che verranno rappresentati sulla legenda;

- `_pathMode:String`

Stringa che identifica il metodo di calcolo del percorso.

Metodi

+ `MapChartModel(id:Integer,title:String,paths:Array,points:Array,center:Object,options:Object):MapChartModel`

Costruttore della classe `MapChartModel`.

- **`id:Integer`**

Intero che contiene l'id del grafico.

- **`title:String`**

Stringa che contiene il titolo del grafico.

- **`paths:Array`**

Array che contiene i percorsi da tracciare sulla mappa.

- **`points:Array`**

Array che contiene le coordinate dei punti da visualizzare sulla mappa.

- **`center:Object`**

Oggetto che contiene latitudine e longitudine del punto centrale della mappa.

- **`options:Object`**

Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.



4.2.5 Classe PageModel

PageModel
<ul style="list-style-type: none">- <code>_id:Integer</code>- <code>_title:String</code>- <code>_data:Array</code>- <code>_pageWidth:Integer</code>- <code>_columns:Integer</code>
+ <code>PageModel(id:Integer, title:String):PageModel</code>

Tabella 15: Classe PageModel

Descrizione

Questa classe rappresenta il modello dei dati di una pagina.

Utilizzo

Sarà utilizzata da `BusinessLayer::PageController` per la creazione delle pagine.

Attributi

- `_id:Integer`

Codice ID univoco che identifica l'oggetto;

- `_title:String`

Titolo della pagina;

- `_data:Array`

Array contenente gli ID dei grafici associati a quella pagina;

- `_pageWidth:Integer`

Indica la larghezza di visualizzazione della pagina *HTML_G* in pixel, può assumere un valore maggiore di 800;

- `_columns:Integer`

Indica il numero massimo di grafici per riga, può assumere un valore tra 1 e 12.

Metodi

+ `PageModel(id:Integer, title:String):PageModel`

Costruttore della classe PageModel.

o `id:Integer`

Intero che contiene l'id della pagina.

o `title:String`

Stringa che contiene il titolo della pagina.

4.2.6 Classe TableModel

TableModel
<ul style="list-style-type: none"> - <code>_id:Integer</code> - <code>_title:String</code> - <code>_headers:Array</code> - <code>_data:Array</code> - <code>_insertPosition:String</code> - <code>_orderBy:Object</code> - <code>_displayedLines:Integer</code> - <code>_showBorder:String</code> - <code>_colorColumn:Array</code> - <code>_colorRow:Array</code> - <code>_colorCell:Array</code>
+ TableModel(id:Integer, title:String, headers:Array, data:Array, options:Object):TableModel

Tabella 16: Classe TableModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Table_G*.

Utilizzo

Sarà utilizzata da `BusinessLayer::TableController` per la creazione e la modifica dei grafici di tipo *Table_G*.

Attributi

- **`_id:Integer`**
Codice ID univoco che identifica l'oggetto;
- **`_title:String`**
Titolo del grafico;
- **`_headers:Array`**
Array che contiene i valori che verranno riportati negli header delle colonne;
- **`_data:Array`**
Array che contiene i valori che verranno visualizzati nelle celle della tabella;
- **`_insertPosition:String`**
Indica il punto in cui inserire i nuovi dati, può assumere il valore top o bottom;
- **`_orderBy:Object`**
Indica il criterio di ordinamento dei dati della tabella e la colonna di riferimento;
- **`_displayedLines:Integer`**
Indica il massimo numero di elementi visualizzati in ogni pagina della tabella;

**- `_showBorder:String`**

Indica se i bordi della tabella sono visibili o meno, può assumere il valore shown o hidden;

- `_colorColumn:Array`

Array che contiene l'indice della colonna e il colore da impostare;

- `_colorRow:Array`

Array che contiene l'indice della riga e il colore da impostare;

- `_colorCell:Array`

Array che contiene oggetti *JSON_G*, i quali contengono gli indici di riga e colonna, e il colore da impostare.

Metodi

```
+ TableModel(id:Integer,title:String,headers:Array,data:Array,options:Object)
:TableModel
```

Costruttore della classe TableModel.

- **`id:Integer`**
Intero che contiene l'id della tabella.
- **`title:String`**
Stringa che contiene il titolo della tabella.
- **`headers:Array`**
Array che contiene gli header delle colonne della tabella.
- **`data:Array`**
Array che contiene i dati iniziali della tabella.
- **`options:Object`**
Oggetto *JSON_G* che contiene le opzioni aggiuntive da assegnare alla tabella.

4.3 Componente Norris::Lib::PresentationLayer**4.3.1 Classe BarChart**

BarChart
<pre>+ BarChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object) :BarChart + getChartInfo():Object + updateInPlace(label:String, set:Integer, newValue:Integer):void</pre>

Tabella 17: Classe BarChart

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Bar Chart*_G.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Bar Chart*_G.

Attributi

Assenti.

Metodi

```
+ BarChart(title:String,xAxisName:String,yAxisName:String,labels:Array,  
data:Array,options:Object):BarChart
```

Costruttore della classe BarChart.

- **title:String**
Stringa che contiene il titolo del grafico.
- **xAxisName:String**
Stringa che contiene l'etichetta dell'asse delle X.
- **yAxisName:String**
Stringa che contiene l'etichetta dell'asse delle Y.
- **labels:Array**
Array che contiene le etichette dei set di dati.
- **data:Array**
Array che contiene i dati iniziali del grafico.
- **options:Object**
Parametro opzionale, oggetto *JSON*_G che contiene le opzioni aggiuntive da assegnare al grafico. Se non presente verranno impostate le opzioni di default.

```
+ getChartInfo():Object
```

Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

```
+ updateInPlace(label:String,set:Integer,newValue:Integer):void
```

Aggiorna un dato del grafico con modalità in place.

- **label:String**
Indica il nome della label del dato da cambiare.
- **set:Integer**
Intero che indica l'indice della serie a cui appartiene il dato da cambiare.
- **newValue:Integer**
Intero che contiene il nuovo valore da assegnare al dato.

4.3.2 Classe LineChart

LineChart
<pre> + LineChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object) :LineChart + getChartInfo():Object + updateInPlace(label:String, set:Integer, newValue:Array) :void + updateStream(newLabel:String, newValue:Array):void </pre>

Tabella 18: Classe LineChart

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Line Chart_G*.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Line Chart_G*.

Attributi

Assenti.

Metodi

```
+ LineChart(title:String,xAxisName:String,yAxisName:String,labels:Array,
data:Array,options:Object):LineChart
```

Costruttore della classe LineChart.

- **title:String**
Stringa che contiene il titolo del grafico.
- **xAxisName:String**
Stringa che contiene l'etichetta dell'asse delle X.
- **yAxisName:String**
Stringa che contiene l'etichetta dell'asse delle Y.
- **labels:Array**
Array che contiene le etichette dei set di dati.
- **data:Array**
Array che contiene i dati iniziali del grafico.
- **options:Object**
Parametro opzionale, oggetto *JSON_G* che contiene le opzioni aggiuntive

da assegnare al grafico. Se non presente verranno impostate le opzioni di default.

+ `getChartInfo():Object`

Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

+ `updateInPlace(label:String,set:Integer,newValue:Array):void`

Aggiorna un dato del grafico con modalità in place.

- **`label:String`**

Indica il nome della label del dato da cambiare.

- **`set:Integer`**

Indica l'indice della serie a cui appartiene il dato da cambiare.

- **`newValue:Array`**

Array che contiene i nuovi valori del dato.

+ `updateStream(newLabel:String,newValue:Array):void`

Aggiunge una nuova label e un valore per ogni set di dati con modalità stream.

- **`newLabel:String`**

Stringa che contiene il nome della label che verrà aggiunta.

- **`newValue:Array`**

Array che contiene i nuovi valori per ogni set di dati.

4.3.3 Classe MapChart

MapChart
<pre> + MapChart(title:String, paths:Array, points:Array, centerLatitude:Float, centerLongitude:Float, options:Object):MapChart + getChartInfo():Object + updateInPlace(point:Array, latitude:Float, longitude:Float):void + updateMovie(newPoints:Array):void </pre>

Tabella 19: Classe MapChart

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Map Chart_G*.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Map Chart_G*.

Attributi

Assenti.

Metodi

+ `MapChart(title:String,paths:Array,points:Array,centerLatitude:Float,centerLongitude:Float,options:Object):MapChart`

Costruttore della classe MapChart.

- `title:String`
Stringa che contiene il titolo del grafico.
- `paths:Array`
Array che contiene i percorsi da tracciare sulla mappa.
- `points:Array`
Array che contiene le coordinate dei punti da visualizzare sulla mappa.
- `centerLatitude:Float`
Latitudine del punto centrale della mappa.
- `centerLongitude:Float`
Longitudine del punto centrale della mappa.
- `options:Object`
Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.

+ `getChartInfo():Object`

Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

+ `updateInPlace(point:Array,latitude:Float,longitude:Float):void`

Aggiorna un punto della mappa con modalità in place.

- `point:Array`
Array contenente i nuovi punti.
- `latitude:Float`
Nuova latitudine del punto da modificare.
- `longitude:Float`
Nuova longitudine del punto da modificare.

+ `updateMovie(newPoints:Array):void`

Aggiorna i punti della mappa con modalità movie.

- `newPoints:Array`
Array contenente i nuovi punti.

4.3.4 Classe Norris

Norris
+ Norris(nsp:Socket):Object

Tabella 20: Classe Norris

Descrizione

Questa classe fornisce un punto d'accesso allo sviluppatore alla libreria Norris..

Utilizzo

Sarà utilizzata dallo sviluppatore per poter creare ed utilizzare gli altri oggetti forniti dalla libreria Norris. La classe, inoltre, istanzierà un subrouting sul punto di *mount_G* designato dallo sviluppatore.

Attributi

Assenti.

Metodi

+ Norris(nsp:Socket):Object

Costruttore della classe Norris.

- o **nsp:Socket**

Il *namespace_G* di *Socket.io* che verrà utilizzato dalla libreria Norris.

4.3.5 Classe Page

Page
+ Page(title:String, options:Object):Page + getPageInfo():Object + addGraph(graph:Graph):void

Tabella 21: Classe Page

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione alle pagine.

**Utilizzo**

Sarà utilizzata dallo sviluppatore per la creazione e la gestione delle pagine.

Attributi

Assenti.

Metodi

+ Page(title:String,options:Object):Page

Costruttore della classe Page.

- **title:String**

Stringa che contiene il titolo della nuova pagina.

- **options:Object**

Oggetto che contiene le opzioni aggiuntive da assegnare alla pagina.

+ getPageInfo():Object

Metodo che ritorna tutte le informazioni sulla pagina: attributi e grafici contenuti.

+ addGraph(graph:Graph):void

Aggiunge un grafico alla pagina.

- **graph:Graph**

Grafico di tipo *Bar Chart_G*, *Line Chart_G*, *Map Chart_G* o *Table_G* che si vuole aggiungere alla pagina.

4.3.6 Classe PageRouter

PageRouter
+ PageRouter(page:Page):express

Tabella 22: Classe PageRouter

Descrizione

Questa classe gestisce la creazione e l'utilizzo del *subrouting_G*.

Utilizzo

La classe verrà utilizzata per la creazione di un *subrouting_G*, sul punto di *mount_G* designato dallo sviluppatore, che si occuperà di gestire le richieste GET dei *client_G*.

Attributi

Assenti.



Metodi

+ `PageRoute(page:Page):express`

Questo metodo crea un oggetto PageRouter.

- `page:Page`

Un oggetto di tipo Page.

4.3.7 Classe Table

Table
<pre>+ Table(title:String, headers:Array, data:Array, options:Object):Table + getChartInfo():Object + updateInPlace(row:Integer, column:Integer, newValue:Array):void + updateStream(data:Array):void</pre>

Tabella 23: Classe Table

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Table_G*.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Table_G*.

Attributi

Assenti.

Metodi

+ `Table(title:String,headers:Array,data:Array,options:Object):Table`

Costruttore della classe Table.

- `title:String`

Stringa che contiene il titolo della tabella.

- `headers:Array`

Array che contiene gli header delle colonne della tabella.

- `data:Array`

Array che contiene i dati iniziali della tabella.

- **options:Object**

Parametro opzionale, oggetto *JSON_G* che contiene le opzioni aggiuntive da assegnare alla tabella. Se non presente verranno impostate le opzioni di default.

- + **getChartInfo():Object**

Metodo che ritorna tutte le informazioni sulla tabella: attributi, dati ed opzioni.

- + **updateInPlace(row:Integer,column:Integer,newValue:Array):void**

Aggiorna una cella della tabella in modalità in place.

- **row:Integer**

Intero che indica il numero di riga della cella da aggiornare.

- **column:Integer**

Intero che indica il numero di colonna della cella da aggiornare.

- **newValue:Array**

Array che contiene i nuovi valori della cella da aggiornare.

- + **updateStream(data:Array):void**

Aggiunge una nuova riga alla tabella in modalità stream.

- **data:Array**

Array che contiene tutti i dati della nuova riga che verrà aggiunta alla tabella.

4.4 Componente Norris::Lib::Utils

4.4.1 Classe ColorManager

ColorManager
<ul style="list-style-type: none"> + <u>ColorGenerator(num:Integer):Array</u> + <u>hexColorParse(color:String):Object</u>

Tabella 24: Classe ColorManager

Descrizione

Questa classe si occupa di generare dei colori di default per i grafici.

Utilizzo

Sarà utilizzata da *BusinessLayer::BarChartController*, *BusinessLayer::LineChartController* e *BusinessLayer::MapChartController* per generare dei colori di default.

Attributi

Assenti.



Metodi

+ `ColorGenerator(num:Integer):Array`

Questo metodo genera dei colori di default per i dati dei grafici e li ritorna in un array.

- `num:Integer`

Intero che indica il numero di colori da generare.

+ `hexColorParse(color:String):Object`

Controllare la correttezza del colore esadecimale e ritorna un oggetto con la rappresentazione RGB del medesimo colore; altrimenti ritorna un oggetto vuoto.

- `color:String`

Il colore esadecimale da trasformare.

4.4.2 Classe `NorrisError`

NorrisError	
+ <code>NorrisError(err:Integer):void</code> + <code>toString():String</code>	

Tabella 25: Classe `NorrisError`

Descrizione

Questa classe si occupa di gestire i messaggi di errore.

Utilizzo

Sarà utilizzata dalle classi del *package_G* `BusinessLayer` quando si verifica un errore.

Attributi

Assenti.

Metodi

+ `NorrisError(err:Integer):void`

La funzione crea un oggetto di tipo `NorrisError`.

- `err:Integer`

Intero che contiene il codice dell'errore da sollevare.

+ `toString():String`

La funzione crea la stringa `NorrisError`.

4.4.3 Classe ProgressiveID

ProgressiveID
- id:Integer
+ ProgressiveID():Integer

Tabella 26: Classe ProgressiveID

Descrizione

Questa classe si occupa di generare un ID univoco per ogni risorsa, ne esisterà un'unica istanza all'interno di una istanza di Norris.

Utilizzo

Sarà utilizzata dalle classi di BusinessLayer che si occupano della creazione di pagine e grafici.

Questa classe non è soggetta ad accessi concorrenti in quanto le operazioni di creazione avvengono in modo sincrono e non concorrente. Questo inoltre previene la possibilità di creare due risorse distinte con il medesimo ID.

Attributi

- id:Integer

Contatore progressivo che viene incrementato ad ogni invocazione.

Metodi

+ ProgressiveID():Integer

Quando chiamato, questo metodo ritorna il valore corrente dell'id progressivo e poi lo incrementa.

4.4.4 Classe SocketService

SocketService
+ <u>setSocket(socket:Object):void</u>
+ <u>connectionManager():void</u>
+ <u>sendUpdate(room:Integer, info:Object):void</u>
+ <u>getSocketNamespace():String</u>
+ <u>isSocketNamespace():Boolean</u>

Tabella 27: Classe SocketService

Descrizione

Questa classe contiene i metodi riguardanti *socket.io_G* che verranno utilizzati dalla libreria Norris.

Utilizzo

Sarà utilizzata dalla classe `BusinessLayer::SocketController` che si occupa della gestione del *socket_G*.

Attributi

Assenti.

Metodi

+ setSocket(socket:Object):void

Questo metodo imposta il *namespace_G* di *Socket.io_G* che verrà utilizzato dalla libreria Norris.

- **socket:Object**
Oggetto di tipo Socket.

+ connectionManager():void

Questo metodo avvia il gestore di connessioni tra il *server_G* e il *client_G*.
Nello specifico, assegna una nuova funzione all'evento *connection* del *socket_G* principale di Norris. Questa funzione crea un ulteriore evento, in modo che per ogni *socket* che si conetterà a quello principale, all'emissione dell'evento *joinRoom*, venga assegnato ad un'apposita stanza identificata dal codice identificativo del grafico che rappresenta, che verrà poi utilizzata per inviare le notifiche *push_G* esclusivamente ai *socket* interessati.

+ sendUpdate(room:Integer,info:Object):void

Questo metodo invia un *JSON_G* dal *server_G* al *client_G*. L'update verrà inviato esclusivamente ai *socket* interessati all'aggiornamento, tramite l'emissione di quest'ultimo sulla stanza identificata dal codice che viene passato tramite il parametro *room*.

- **room:Integer**
Integer che contiene l'id della room.
- **info:Object**
I dati che vengono inviati al *client_G* per l'aggiornamento sottoforma di *JSON_G*.

+ getSocketNamespace():String

Questo metodo ritorna il *namespace_G* di *Socket.io_G* correntemente utilizzato da Norris.

+ isSocketNamespace():Boolean

Questo metodo controlla che un determinato oggetto sia un'istanza di *namespace_G* di *socket.io_G*.

5 Specifica classi del *front-end_G*

5.1 Componente Norris::NorrisApp::Controllers

5.1.1 Classe BarLineChartCtrl

BarLineChartCtrl
+ scope:Object + window:Object + scope.graph:Object
+ BarLineChartCtrl(scope:Object, SocketsSvc:Object, BarLineSvc:Object, window:Object):BarLineChartCtrl

Tabella 28: Classe BarLineChartCtrl

Descrizione

Questa classe contiene i metodi necessari per gestire i grafici di tipo *Bar Chart_G* e *Line Chart_G* lato *front-end_G*.

Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end_G* che riguardano i grafici di tipo *Bar Chart_G* e *Line Chart_G*.

Attributi

+ scope:Object

Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato;

+ window:Object

Questo campo dati rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri;

+ scope.graph:Object

Questo campo dati rappresenta il grafico generato dalla libreria Norris, con tutte le sue proprietà. Contiene l'id del grafico, il titolo, le serie di dati, le etichette ad esse associate e le proprietà del grafico stesso.

Metodi

+ BarLineChartCtrl(scope:Object, SocketsSvc:Object, BarLineSvc:Object, window:Object):BarLineChartCtrl

Questo metodo costruisce la classe BarLineChartCtrl.

- **scope:Object**
Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato.
- **SocketsSvc:Object**
Questo parametro rappresenta un riferimento al servizio SocketsSvc.
- **BarLineSvc:Object**
Questo parametro rappresenta un riferimento al servizio BarLineSvc.
- **window:Object**
Questo parametro rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

5.1.2 Classe FrontCtrl

FrontCtrl
+ scope:Object - _data:Object - colClass:Integer
+ FrontCtrl(scope:Object, FrontSvc:Object, FirstConnectSvc:Object):FrontCtrl

Tabella 29: Classe FrontCtrl

Descrizione

Questa classe si occupa di gestire il *front-end_G*.

Utilizzo

Sarà utilizzata allo scopo di eseguire le prime chiamate necessarie al *front-end_G*.

Attributi

- + **scope:Object**
Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato;
- **_data:Object**
Questo campo dati contiene i dati di un oggetto pagina, comprensivi del titolo della pagina, del *namespace_G* del *socket_G* e dei dati dei grafici;
- **colClass:Integer**
Questo campo dati è calcolato per ottenere il valore da 1 a 12 che rappresenta la larghezza delle colonne secondo le specifiche di Bootstrap.

Metodi

+ FrontCtrl(scope:Object,FrontSvc:Object,FirstConnectSvc:Object):FrontCtrl

Questo metodo costruisce la classe FrontCtrl.

- **scope:Object**
Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .
- **FrontSvc:Object**
Questo parametro rappresenta un riferimento al servizio FrontSvc.
- **FirstConnectSvc:Object**
Questo parametro rappresenta un riferimento al servizio FirstConnectSvc.

5.1.3 Classe MapChartCtrl

MapChartCtrl
+ scope:Object + scope.graph:Object + window:Object
+ MapChartCtrl(scope:Object, SocketsSvc:Object, MapSvc:Object, window:Object):MapChartCtrl

Tabella 30: Classe MapChartCtrl

Descrizione

Questa classe contiene i metodi necessari per gestire il grafico di tipo *Map Chart_G* lato *front-end_G* .

Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end_G* che riguardano il grafico di tipo *Map Chart_G*.

Attributi

- + scope:Object**
Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato ;
- + scope.graph:Object**
Questo campo dati rappresenta il grafico generato dalla libreria Norris, con tutte le sue proprietà. Contiene l'id del grafico, il titolo, i punti e i percorsi da rappresentare nonché le proprietà del grafico stesso ;

+ **window:Object**

Questo campo dati rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

Metodi

+ MapChartCtrl(scope:Object, SocketsSvc:Object, MapSvc:Object, window:Object):MapChartCtrl

Questo metodo costruisce la classe MapChartCtrl.

- **scope:Object**

Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .

- **SocketsSvc:Object**

Questo parametro rappresenta un riferimento al servizio SocketsSvc .

- **MapSvc:Object**

Questo parametro rappresenta un riferimento al servizio MapSvc .

- **window:Object**

Questo parametro rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

5.1.4 Classe TableCtrl

TableCtrl
+ scope:Object + window:Object + scope.graph:Object
<u>+ TableCtrl(scope:Object, SocketsSvc:Object, TableSvc:Object, window:Object):TableCtrl</u>

Tabella 31: Classe TableCtrl

Descrizione

Questa classe contiene i metodi necessari per gestire il grafico di tipo *Table_G* lato *front-end_G* .

Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end_G* che riguardano il grafico di tipo *Table_G*.



Attributi

+ `scope:Object`

Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato ;

+ `window:Object`

Questo campo dati rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri ;

+ `scope.graph:Object`

Questo campo dati rappresenta il grafico generato dalla libreria Norris, con tutte le sue proprietà. Contiene l'id del grafico, il titolo, le serie di dati, le etichette ad esse associate e le proprietà del grafico stesso .

Metodi

+ `TableCtrl(scope:Object, SocketsSvc:Object, TableSvc:Object, window:Object):TableCtrl`

Questo metodo costruisce la classe `TableCtrl` .

o `scope:Object`

Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .

o `SocketsSvc:Object`

Questo parametro rappresenta un riferimento al servizio `SocketsSvc` .

o `TableSvc:Object`

Questo parametro rappresenta un riferimento al servizio `TableSvc` .

o `window:Object`

Questo parametro rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

5.2 Componente Norris::NorrisApp::Model

5.2.1 Classe `BarChartMdl`

BarChartMdl
+ <code>graphG:Object</code>

Tabella 32: Classe `BarChartMdl`

Descrizione

Questa classe conterrà i valori relativi al grafico *Bar Chart_G* dal lato *front-end_G*.



Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS_G* prevede l'inserimento dei valori all'interno delle classi del *package_G* *Controllers*. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end_G* di Norris.

Attributi

+ **graphG:Object**

Questo campo dati, istanziato in *BarLineChartCtrl* rappresenta il modello dei dati di un grafico di tipo *Bar Chart_G*.

Metodi

Assenti.

5.2.2 Classe FrontMdl

FrontMdl
+ colMdClass:String + nspSock:String

Tabella 33: Classe FrontMdl

Descrizione

Questa classe conterrà i valori relativi ai grafici di una pagina dal lato *front-end_G*.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS_G* prevede l'inserimento dei valori all'interno delle classi del *package_G* *Controllers*. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end_G* di Norris.

Attributi

+ **colMdClass:String**

Questo campo dati rappresenta la proprietà CSS da assegnare alle colonne per rappresentarne la larghezza in modalità schermo medio secondo le specifiche di Bootstrap;

+ **nspSock:String**

Rappresenta il *namespace_G* a cui sarà necessario connettersi per ottenere dati dal *socket_G* del *server_G*.

**Metodi**

Assenti.

5.2.3 Classe LineChartMdl

LineChartMdl
+ graphG:Object

Tabella 34: Classe LineChartMdl

Descrizione

Questa classe conterrà i valori relativi al grafico *Line Chart_G* dal lato *front-end_G*.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS_G* prevede l'inserimento dei valori all'interno delle classi del *package_G* Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end_G* di Norris.

Attributi

+ graphG:Object

Questo campo dati, istanziato in BarLineChartCtrl rappresenta il modello dei dati di un grafico di tipo *Line Chart_G*.

Metodi

Assenti.

5.2.4 Classe MapChartMdl

MapChartMdl
+ title:String + center:Array + zoom:Number + markers:Array + polylines:Array + colors:Array + legendItems:Array

Tabella 35: Classe MapChartMdl

Descrizione

Questa classe conterrà i valori relativi al grafico *Map Chart_G* dal lato *front-end_G*.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS_G* prevede l'inserimento dei valori all'interno delle classi del *package_G* Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end_G* di Norris.

Attributi

+ title:String

Rappresenta il titolo del grafico di tipo *Map Chart_G*;

+ center:Array

Rappresenta il punto centrale del *Map Chart_G*;

+ zoom:Number

Rappresenta il livello di zoom iniziale del *Map Chart_G*;

+ markers:Array

Rappresenta i punti che verranno disegnati sul *Map Chart_G*;

+ polylines:Array

Rappresenta i percorsi che verranno tracciati sul *Map Chart_G*;

+ colors:Array

Rappresenta i colori dei percorsi rappresentati in polylines;

+ legendItems:Array

Rappresenta i nomi dei percorsi che verranno inseriti nella legenda, se attiva.

**Metodi**

Assenti.

5.2.5 Classe TableMdl

TableMdl
+ graphG:Object

Tabella 36: Classe TableMdl

Descrizione

Questa classe conterrà i valori relativi al grafico *Table_G* dal lato *front-end_G*.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS_G* prevede l'inserimento dei valori all'interno delle classi del *package_G* Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end_G* di Norris.

Attributi

+ graphG:Object

Questo campo dati, istanziato in TableCtrl rappresenta il modello dei dati di un grafico di tipo *Table_G* .

Metodi

Assenti.

5.3 Componente Norris::NorrisApp::Services

5.3.1 Classe BarLineSvc

BarLineSvc
<pre> + BarLineSvc(ColorsSvc:Object):BarLineSvc + fillLineData(series:Array, labels:Array, inData:Array, outData:Array):Array + setColors(colors:Array):Array + setOpts(title:String, xAxisName:String, yAxisName:String, showGrid:boolean, showLegend:boolean, legendPosition:String, seriesCount:String):Object </pre>

Tabella 37: Classe BarLineSvc

Descrizione

Questa classe contiene i metodi che effettuano operazioni sui dati dei grafici in `Controllers::BarLineChartCtrl`.

Utilizzo

Sarà utilizzata dalla classe `Controllers::BarLineChartCtrl` per operazioni quali il popolamento delle strutture dati e la definizione delle proprietà.

Attributi

Assenti.

Metodi

`+ BarLineSvc(ColorsSvc:Object):BarLineSvc`

Costruisce la classe `BarLineSvc`.

- **ColorsSvc:Object**

Rappresenta un riferimento alla classe `ColorsSvc`.

`+ fillLineData(series:Array, labels:Array, inData:Array, outData:Array):Array`

Riempie l'array dei dati di un grafico di tipo *Line Chart_G* o *Bar Chart_G*.

- **series:Array**

Rappresenta l'array dei nomi delle serie di dati.

- **labels:Array**

Rappresenta le etichette dell'asse X per i dati.

- **inData:Array**

Dati grezzi in arrivo dal *back-end_G*.

- **outData:Array**
Array dei dati in formato corretto da inserire nel modello del grafico.

+ setColors(colors:Array):Array

Il metodo scorre l'array dei colori delle serie e li inserisce nel modello dati.

- **colors:Array**
Colori delle varie serie di dati.

+ setOpts(title:String,xAxisName:String,yAxisName:String,showGrid:boolean,showLegend:boolean,legendPosition:String,seriesCount:String):Object

Il metodo crea l'oggetto contenente le opzioni del grafico *Line Chart_G* o *Bar Chart_G*.

- **title:String**
Titolo del grafico.
- **xAxisName:String**
Etichetta dell'asse delle X.
- **yAxisName:String**
Etichetta dell'asse delle Y.
- **showGrid:boolean**
Stato di attivazione della griglia.
- **showLegend:boolean**
Stato di attivazione della legenda del grafico .
- **legendPosition:String**
Posizione della legenda rispetto all'area del grafico.
- **seriesCount:String**
Numero massimo di serie di valori da mantenere nel modello.

5.3.2 Classe ColorsSvc

ColorsSvc
<u>+ ColorsSvc():ColorsSvc</u> <u>- componentToHex(component:Number):String</u> <u>+ rgbToHex(r:Number, g:Number, b:Number):String</u>

Tabella 38: Classe ColorsSvc

Descrizione

Questa classe contiene i metodi necessari a convertire un colore da formato RGB a esadecimale.

Utilizzo

Servirà per convertire i colori dal formato ricevuto dal *server_G* a quello conforme alle librerie utilizzate lato *front-end_G*.

**Attributi**

Assenti.

Metodi

+ ColorsSvc():ColorsSvc

Costruisce la classe ColorsSvc.

- componentToHex(component:Number):String

Converte un valore numerico in esadecimale.

- **component:Number**

Rappresenta un valore decimale da convertire in esadecimale.

+ rgbToHex(r:Number,g:Number,b:Number):String

Converte un tre valori numerici in una stringa esadecimale.

- **r:Number**

Rappresenta il valore per il colore rosso (0-255).

- **g:Number**

Rappresenta il valore per il colore verde (0-255).

- **b:Number**

Rappresenta il valore per il colore blu (0-255).

5.3.3 Classe FirstConnectSvc

FirstConnectSvc
+ FirstConnectSvc(resource:Object, location:Object):Object

Tabella 39: Classe FirstConnectSvc

Descrizione

Questa classe contiene il metodo necessario al recupero dell'*URL_G* del *server_G*.

Utilizzo

Sarà utilizzata come classe di supporto alla classe *Controllers::FirstConnectCtrl*.

Attributi

Assenti.



Metodi

+ FirstConnectSvc(resource:Object, location:Object):Object

Il metodo legge l'*URL_G* corrente ed effettua la richiesta al *back-end_G*.

- **resource:Object**
Rappresenta una risorsa *REST_G*, è un oggetto *AngularJS_G*.
- **location:Object**
Rappresenta una *URL_G*, è un oggetto fornito da *AngularJS_G*.

5.3.4 Classe FrontSvc

FrontSvc
<u>+ FrontSvc():FrontSvc</u> <u>+ createRows(data:Array, columns:Number):Array</u>

Tabella 40: Classe FrontSvc

Descrizione

Questa classe contiene i metodi che effettuano operazioni sui dati grezzi della pagina in `Controllers::FrontCtrl`.

Utilizzo

Sarà utilizzata dalla classe `Controllers::FrontCtrl` per calcolare proprietà della pagina a partire dai dati grezzi.

Attributi

Assenti.

Metodi

+ FrontSvc():FrontSvc

Il metodo costruisce la classe `FrontSvc`.

+ createRows(data:Array, columns:Number):Array

Divide i dati grezzi in righe e colonne.

- **data:Array**
Rappresenta i dati grezzi dei grafici nella pagina.
- **columns:Number**
Rappresenta il numero di colonne in cui verranno disposti i grafici, e pertanto il numero massimo di grafici per riga.

5.3.5 Classe MapSvc

MapSvc
<pre> + MapSvc(ColorsSvc:Object):MapSvc - cnvLatLng(x:Array):Object - setPathMode(mode:String):Object + setColors(colors:Array):Array + createPolyline(pathLine:Array, colors:String, map:Object):Object + createMarker(point:Array, map:Object):Object + buildPath(path:Array, color:String, map:Object, polyLines:Array, method:String):void + buildLegend(map:Object, position:Array, id:String):void + updateMovie(markers:Array, newData:Array, map:Object):void </pre>

Tabella 41: Classe MapSvc

Descrizione

Questa classe contiene i metodi che effettuano operazioni sui dati dei grafici in `Controllers::MapChartCtrl`.

Utilizzo

Sarà utilizzata dalla classe `Controllers::MapChartCtrl` per operazioni sulle mappe, quali calcolo di percorsi e inserimento di segnalini.

Attributi

Assenti.

Metodi

+ MapSvc(ColorsSvc:Object):MapSvc

Crea la classe MapSvc.

- **ColorsSvc:Object**

Rappresenta un riferimento alla classe ColorsSvc.

- cnvLatLng(x:Array):Object

Converte un array di due punti in un oggetto LatLng da usare come punto in *Google Maps API_G*.

- **x:Array**

Rappresenta l'array da convertire.

- `setPathMode(mode:String):Object`

Converte una stringa in un oggetto per poter essere utilizzato come parametro nel calcolo dei percorsi.

- `mode:String`

Rappresenta il metodo con il quale calcolare il percorso.

+ `setColors(colors:Array):Array`

Metodo di utilità che converte i colori in formato esadecimale.

- `colors:Array`

Rappresenta i colori dei vari percorsi.

+ `createPolyline(pathLine:Array,colors:String,map:Object):Object`

A partire da una serie di punti il metodo crea una linea colorata sulla mappa.

- `pathLine:Array`

Rappresenta i punti per i quali passa la linea.

- `colors:String`

Il colore della linea.

- `map:Object`

È il riferimento alla mappa sulla quale inserire la linea.

+ `createMarker(point:Array,map:Object):Object`

Il metodo crea un segnalino sulla mappa a partire dalle coordinate di un punto.

- `point:Array`

Rappresenta le coordinate sulle quali inserire il segnalino.

- `map:Object`

Riferimento alla mappa sulla quale inserire il punto.

+ `buildPath(path:Array,color:String,map:Object,polylines:Array,method:String):void`

Il metodo, tramite una chiamata al servizio Navigazione di Google, a partire da due punti genera un percorso stradale tra i punti, e li disegna sulla mappa.

- `path:Array`

Rappresenta i punti di inizio e fine del percorso.

- `color:String`

Rappresenta il colore del percorso da tracciare.

- `map:Object`

Rappresenta il riferimento alla mappa sulla quale disegnare il percorso.

- `polylines:Array`

È l'array del modello dati nel quale inserire i valori dei percorsi.

- `method:String`

Rappresenta il metodo di calcolo da utilizzare per i percorsi.

+ `buildLegend(map:Object,position:Array,id:String):void`

Il metodo posiziona la legenda dentro la mappa.

- `map:Object`

Rappresenta il riferimento alla mappa.

- **position:Array**
Il punto nel quale inserire la legenda.
- **id:String**
È L'identificativo HTML dell'elemento contenente la legenda.

+ updateMovie(markers:Array,newData:Array,map:Object):void

Il metodo aggiorna i punti presenti sulla mappa con il metodo stream, ne aggiunge nel caso ne esistano di nuovi, rimuove quelli non più presenti e aggiorna quelli che hanno cambiato posizione.

- **markers:Array**
Array dei segnalini presenti nella mappa.
- **newData:Array**
Dati aggiornati dei punti da rappresentare.
- **map:Object**
Rappresenta il riferimento alla mappa .

5.3.6 Classe SocketsSvc

SocketsSvc
- sockets:Array
+ SocketsSvc(scope:Object):SocketsSvc
+ open(id:integer, namespace:String):void
+ on(id:Number, eventName:String, callback:function):void
+ emit(id:Number, eventName:String, data:Object, callback:function):void

Tabella 42: Classe SocketsSvc

Descrizione

Questa classe contiene i metodi necessari alla gestione dei servizi di *Socket.io_G* dal lato del *front-end_G*.

Utilizzo

Sarà utilizzata allo scopo di gestire le connessioni con il *socket_G* lato *server_G* dalle classi del *package_G* Controllers.

Attributi

- **sockets:Array**
Rappresenta l'array dei *socket_G*, uno per ciascun grafico della pagina.

Metodi

+ SocketsSvc(scope:Object):SocketsSvc

Costruisce la classe SocketsSvc.

- **scope:Object**

Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .

+ open(id:integer,namespace:String):void

Apri una connessione al *namespace_G* specificato.

- **id:integer**

Rappresenta il codice identificativo del grafico del quale aprire un *socket_G*.

- **namespace:String**

Rappresenta il *namespace_G* sul quale aprire la connessione.

+ on(id:Number,eventName:String,callback:function):void

Attiva una callback che ascolta un determinato evento.

- **id:Number**

Rappresenta il codice identificativo del grafico del quale aprire un *socket_G* .

- **eventName:String**

Rappresenta l'evento da mantenere osservato.

- **callback:function**

Rappresenta la funzione da chiamare al verificarsi dell'evento.

+ emit(id:Number,eventName:String,data:Object,callback:function):void

Emette un evento sul *socket_G* specificato.

- **id:Number**

Rappresenta il codice identificativo del grafico del quale aprire un *socket_G* .

- **eventName:String**

Rappresenta l'evento da emettere.

- **data:Object**

Rappresenta i dati da emettere.

- **callback:function**

Rappresenta la funzione da chiamare al una volta emesso l'evento .

5.3.7 Classe TableSvc

TableSvc
<pre> + TableSvc(ColorsSvc:Object):TableSvc + fillData(headers:Array, inData:Array, colors:Array, border:Array):Array + inplaceUpd(inData:Array, outData:Array):void + streamUpd(inData:Array, outData:Array, limit:Number):void </pre>

Tabella 43: Classe TableSvc

Descrizione

Questa classe contiene i metodi che effettuano operazioni sui dati dei grafici in `Controllers::TableCtrl`.

Utilizzo

Sarà utilizzata dalla classe `Controllers::TableCtrl` per alcune operazioni sui dati, quali popolamento delle strutture dati e inserimento delle proprietà.

Attributi

Assenti.

Metodi

[+ TableSvc\(ColorsSvc:Object\):TableSvc](#)

Il metodo costruisce la classe TableSvc.

- **ColorsSvc:Object**

Questo parametro rappresenta un riferimento al servizio ColorsSvc.

[+ fillData\(headers:Array, inData:Array, colors:Array, border:Array\):Array](#)

Il metodo genera, a partire dai dati grezzi forniti dal *back-end_G*, i dati per il modello.

- **headers:Array**
Rappresenta le intestazioni delle colonne della tabella.
- **inData:Array**
Rappresenta l'array dei dati in arrivo.
- **colors:Array**
Rappresenta l'array dei colori dei dati.
- **border:Array**
Rappresenta lo stato dei bordi delle varie celle della tabella.



+ inplaceUpd(inData:Array,outData:Array):void

Il metodo aggiorna i dati del modello con modalità in place.

- **inData:Array**
Rappresenta l'array dei nuovi dati.
- **outData:Array**
Rappresenta l'array dei dati aggiornati.

+ streamUpd(inData:Array,outData:Array,limit:Number):void

Il metodo aggiorna i dati del modello con metodo stream e controlla che non venga superato il limite massimo di valori.

- **inData:Array**
Rappresenta l'array dei dati in arrivo.
- **outData:Array**
Rappresenta l'array dei dati aggiornati.
- **limit:Number**
Rappresenta il limite massimo di righe di dati da salvare.

5.4 Componente Norris::NorrisApp::Views

5.4.1 Classe BarChartView

BarChartView

Tabella 44: Classe BarChartView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Bar Chart_G* nel *front-end_G*.

Utilizzo

Sarà utilizzata da *AngularJS_G* per visualizzare un grafico di tipo *Bar Chart_G* all'interno della pagina finale.

Attributi

Assenti.

Metodi

Assenti.

5.4.2 Classe Index

Index

Tabella 45: Classe Index

Descrizione

Questa classe importa tutti gli script e i file necessari al *front-end_G*. .

Utilizzo

Sarà inviata ai browser in seguito alla loro prima richiesta e si occuperà di importare gli script necessari al funzionamento del *front-end_G*..

Attributi

Assenti.

Metodi

Assenti.

5.4.3 Classe LineChartView

LineChartView

Tabella 46: Classe LineChartView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Line Chart_G* nel *front-end_G* .

Utilizzo

Sarà utilizzata da *AngularJS_G* per visualizzare un grafico di tipo *Line Chart_G* all'interno della pagina finale .

Attributi

Assenti.

Metodi

Assenti.

5.4.4 Classe MapChartView

MapChartView

Tabella 47: Classe MapChartView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Map Chart_G* dal lato *front-end_G*.

Utilizzo

Sarà utilizzata da *AngularJS_G* visualizzare un grafico di tipo *Map Chart_G* all'interno della pagina finale .

Attributi

Assenti.

Metodi

Assenti.

5.4.5 Classe TableView

TableView

Tabella 48: Classe TableView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Table_G* dal lato *front-end_G*.

Utilizzo

Sarà utilizzata da *AngularJS_G* per visualizzare un grafico di tipo *Table_G* all'interno della pagina finale.

Attributi

Assenti.

Metodi

Assenti.

6 Tracciamento

6.1 Tracciamento requisiti-classi

Requisiti	Classi
RAF1	Norris::Lib::PresentationLayer::Norris
RAF1.1	Norris::Lib::Utils::SocketService
RAF1.2	Norris::Lib::BusinessLayer::SocketController
RAF1.2.1	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::SocketController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::TableModel
RAF1.2.2	Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::SocketController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::TableModel
RAF1.2.3	Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::SocketController Norris::Lib::DataLayer::MapChartModel
RAF1.3	Norris::Lib::PresentationLayer::Page
RAF2	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF2.1	Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::PageModel

Requisiti	Classi
RAF3	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF3.1	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::PageModel
RAF3.2	Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::PageModel
RAF3.3	Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel
RAF3.4	Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF3.5	Norris::NorrisApp::Controllers::BarLineChartCtrl Norris::NorrisApp::Controllers::FrontCtrl Norris::NorrisApp::Controllers::MapChartCtrl Norris::NorrisApp::Controllers::TableCtrl Norris::NorrisApp::Model::BarChartMdl Norris::NorrisApp::Model::LineChartMdl Norris::NorrisApp::Model::MapChartMdl Norris::NorrisApp::Model::TableMdl Norris::NorrisApp::Services::BarLineSvc Norris::NorrisApp::Services::ColorsSvc Norris::NorrisApp::Services::FirstConnectSvc Norris::NorrisApp::Services::FrontSvc Norris::NorrisApp::Services::MapSvc Norris::NorrisApp::Services::SocketsSvc Norris::NorrisApp::Services::TableSvc
RAF3.5.1	Norris::NorrisApp::Views::BarChartView Norris::NorrisApp::Views::Index Norris::NorrisApp::Views::LineChartView Norris::NorrisApp::Views::MapChartView Norris::NorrisApp::Views::TableView

Requisiti	Classi
RAF4	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF4.1	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF4.1.1	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.2	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.3	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.4	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.5	Norris::Lib::PresentationLayer::Table
RAF4.1.6	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart

Requisiti	Classi
RAF4.1.7	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart
RAF4.1.8	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart
RAF4.1.9	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart
RAF4.1.10	Norris::Lib::PresentationLayer::BarChart
RAF4.1.11	Norris::Lib::PresentationLayer::MapChart
RAF4.1.12	Norris::Lib::PresentationLayer::MapChart
RAF4.1.13	Norris::Lib::PresentationLayer::Table
RAF4.1.14	Norris::Lib::PresentationLayer::Table
RAF4.1.15	Norris::Lib::PresentationLayer::Table
RAF4.1.16	Norris::Lib::PresentationLayer::Table
RAF4.1.17	Norris::Lib::PresentationLayer::Table
RAF4.2	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF4.2.1	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::PageRouter Norris::Lib::PresentationLayer::Table
RAV3	Norris::Lib::Utils::SocketService

Tabella 49: Tracciamento Requisiti - Classi

6.2 Tracciamento classi-requisiti

Classi	Requisiti
Norris::Lib::BusinessLayer::ActiveResourcesController	
Norris::Lib::BusinessLayer::BarChartController	RAF1.2.1 RAF2 RAF3.1 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::DataConsistency	
Norris::Lib::BusinessLayer::LineChartController	RAF1.2.1 RAF1.2.2 RAF2 RAF3.2 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::MapChartController	RAF1.2.1 RAF1.2.3 RAF2 RAF3.3 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::PageController	RAF2 RAF2.1 RAF3.1 RAF3.2 RAF3.3 RAF3.4 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::SocketController	RAF1.2 RAF1.2.1 RAF1.2.2 RAF1.2.3

Classi	Requisiti
Norris::Lib::BusinessLayer::TableController	RAF1.2.1 RAF1.2.2 RAF2 RAF3.4 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::ActiveResources	
Norris::Lib::DataLayer::BarChartModel	RAF1.2.1 RAF2 RAF3.1 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::LineChartModel	RAF1.2.1 RAF1.2.2 RAF2 RAF3.2 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::MapChartModel	RAF1.2.1 RAF1.2.3 RAF2 RAF3.3 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::PageModel	RAF2 RAF2.1 RAF3.1 RAF3.2 RAF3.3 RAF3.4 RAF4 RAF4.1 RAF4.2



Classi	Requisiti
Norris::Lib::DataLayer::TableModel	RAF1.2.1 RAF1.2.2 RAF2 RAF3.4 RAF4 RAF4.1 RAF4.2
Norris::Lib::PresentationLayer::BarChart	RAF3 RAF4.1.1 RAF4.1.10 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.6 RAF4.1.7 RAF4.1.8 RAF4.1.9 RAF4.2.1
Norris::Lib::PresentationLayer::LineChart	RAF3 RAF4.1.1 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.6 RAF4.1.7 RAF4.1.8 RAF4.1.9 RAF4.2.1
Norris::Lib::PresentationLayer::MapChart	RAF3 RAF4.1.1 RAF4.1.11 RAF4.1.12 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.6 RAF4.1.7 RAF4.1.8 RAF4.2.1
Norris::Lib::PresentationLayer::Norris	RAF1
Norris::Lib::PresentationLayer::Page	RAF1.3

Classi	Requisiti
Norris::Lib::PresentationLayer::PageRouter	RAF4.2.1
Norris::Lib::PresentationLayer::Table	RAF3 RAF4.1.1 RAF4.1.13 RAF4.1.14 RAF4.1.15 RAF4.1.16 RAF4.1.17 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.5 RAF4.2.1
Norris::Lib::Utils::ColorManager	
Norris::Lib::Utils::NorrisError	
Norris::Lib::Utils::ProgressiveID	
Norris::Lib::Utils::SocketService	RAF1.1 RAV3
Norris::NorrisApp::Controllers::BarLineChartCtrl	RAF3.5
Norris::NorrisApp::Controllers::FrontCtrl	RAF3.5
Norris::NorrisApp::Controllers::MapChartCtrl	RAF3.5
Norris::NorrisApp::Controllers::TableCtrl	RAF3.5
Norris::NorrisApp::Model::BarChartMdl	RAF3.5
Norris::NorrisApp::Model::FrontMdl	
Norris::NorrisApp::Model::LineChartMdl	RAF3.5
Norris::NorrisApp::Model::MapChartMdl	RAF3.5
Norris::NorrisApp::Model::TableMdl	RAF3.5
Norris::NorrisApp::Services::BarLineSvc	RAF3.5
Norris::NorrisApp::Services::ColorsSvc	RAF3.5
Norris::NorrisApp::Services::FirstConnectSvc	RAF3.5
Norris::NorrisApp::Services::FrontSvc	RAF3.5

Classi	Requisiti
Norris::NorrisApp::Services::MapSvc	RAF3.5
Norris::NorrisApp::Services::SocketsSvc	RAF3.5
Norris::NorrisApp::Services::TableSvc	RAF3.5
Norris::NorrisApp::Views::BarChartView	RAF3.5.1
Norris::NorrisApp::Views::Index	RAF3.5.1
Norris::NorrisApp::Views::LineChartView	RAF3.5.1
Norris::NorrisApp::Views::MapChartView	RAF3.5.1
Norris::NorrisApp::Views::TableView	RAF3.5.1

Tabella 50: Tracciamento Classi - Requisiti

6.3 Tracciamento test di unità - metodi

Test	Metodi
TU1	Norris::Lib::PresentationLayer::Page::getPageInfo() Norris::Lib::BusinessLayer::BarChartController::getChartInfo() Norris::Lib::BusinessLayer::LineChartController::getChartInfo() Norris::Lib::BusinessLayer::MapChartController::getChartInfo() Norris::Lib::BusinessLayer::SocketController::setSocket() Norris::Lib::BusinessLayer::SocketController::socketNamespace() Norris::Lib::BusinessLayer::TableController::getChartInfo() Norris::Lib::Utils::SocketService::setSocket() Norris::Lib::Utils::SocketService::getSocketNamespace() Norris::Lib::PresentationLayer::BarChart::getChartInfo() Norris::Lib::PresentationLayer::LineChart::getChartInfo() Norris::Lib::PresentationLayer::Table::getChartInfo() Norris::Lib::BusinessLayer::PageController::getPageInfo() Norris::Lib::PresentationLayer::MapChart::getChartInfo()
TU2	Norris::Lib::PresentationLayer::PageRouter::PageRouter()
TU3	Norris::Lib::BusinessLayer::PageController::createPage()
TU4	Norris::Lib::PresentationLayer::PageRouter::PageRouter()
TU5	Norris::Lib::BusinessLayer::BarChartController::createBarChart() Norris::Lib::BusinessLayer::LineChartController::createLineChart() Norris::Lib::BusinessLayer::MapChartController::createMapChart() Norris::Lib::BusinessLayer::TableController::createTable()
TU6	Norris::Lib::Utils::NorrisError::NorrisError()
TU7	Norris::Lib::BusinessLayer::PageController::createPage()
TU9	Norris::Lib::PresentationLayer::Page::addGraph() Norris::Lib::BusinessLayer::PageController::addGraphToPage()
TU10	Norris::Lib::BusinessLayer::PageController::createPage()

Test	Metodi
TU11	Norris::Lib::BusinessLayer::BarChartController::updateInPlace() Norris::Lib::BusinessLayer::LineChartController::updateStream() Norris::Lib::BusinessLayer::MapChartController::updateMovie() Norris::Lib::BusinessLayer::TableController::updateStream() Norris::Lib::PresentationLayer::BarChart::updateInPlace() Norris::Lib::PresentationLayer::LineChart::updateInPlace() Norris::Lib::PresentationLayer::LineChart::updateStream() Norris::Lib::PresentationLayer::Table::updateInPlace() Norris::Lib::PresentationLayer::Table::updateStream() Norris::Lib::BusinessLayer::LineChartController::updateInPlace() Norris::Lib::BusinessLayer::TableController::updateInPlace() Norris::Lib::PresentationLayer::MapChart::updateInPlace() Norris::Lib::PresentationLayer::MapChart::updateMovie() Norris::Lib::BusinessLayer::MapChartController::updateInPlace()
TU13	Norris::Lib::BusinessLayer::PageController::addGraphToPage()
TU14	Norris::Lib::BusinessLayer::TableController::updateStream()
TU15	Norris::Lib::BusinessLayer::TableController::updateStream()
TU16	Norris::Lib::BusinessLayer::TableController::createTable()
TU20	Norris::Lib::BusinessLayer::PageController::createPage()
TU27	Norris::Lib::BusinessLayer::PageController::createPage()
TU29	Norris::Lib::Utils::NorrisError::NorrisError()
TU30	Norris::Lib::BusinessLayer::SocketController::sendUpdate() Norris::Lib::Utils::SocketService::sendUpdate()
TU32	Norris::Lib::Utils::ProgressiveID::ProgressiveID()
TU37	Norris::Lib::BusinessLayer::PageController::addGraphToPage()
TU39	Norris::Lib::Utils::ColorManager::ColorGenerator()
TU40	Norris::Lib::Utils::SocketService::connectionManager()
TU41	Norris::Lib::BusinessLayer::ActiveResourcesController::storePage()
TU42	Norris::Lib::BusinessLayer::ActiveResourcesController::retrieveGraph() Norris::Lib::BusinessLayer::ActiveResourcesController::retrievePage()
TU43	Norris::Lib::PresentationLayer::Norris::Norris()
TU44	Norris::Lib::PresentationLayer::PageRouter::PageRouter()

Test	Metodi
TU45	Norris::Lib::BusinessLayer::PageController::createPageOptions()
TU46	Norris::Lib::DataLayer::ActiveResources::ActiveResources()
TU47	Norris::Lib::BusinessLayer::ActiveResourcesController::storeGraph()
TU48	Norris::Lib::BusinessLayer::DataConsistency::checkOrientation()
TU49	Norris::Lib::BusinessLayer::DataConsistency::checkLegendPosition()
TU50	Norris::Lib::BusinessLayer::DataConsistency::checkMapLegendPosition()
TU51	Norris::Lib::BusinessLayer::DataConsistency::checkMapZoom()
TU52	Norris::Lib::BusinessLayer::DataConsistency::pathMode()
TU53	Norris::Lib::BusinessLayer::DataConsistency::checkOrderBy()
TU54	Norris::Lib::BusinessLayer::DataConsistency::checkInsertPosition()
TU55	Norris::Lib::BusinessLayer::DataConsistency::checkDisplayedLines()
TU56	Norris::Lib::BusinessLayer::DataConsistency::checkColors()
TU57	Norris::Lib::BusinessLayer::DataConsistency::checkHex()
TU58	Norris::Lib::BusinessLayer::DataConsistency::checkColorArray()
TU59	Norris::Lib::BusinessLayer::DataConsistency::checkColorMatrix()
TU60	Norris::Lib::BusinessLayer::DataConsistency::checkSeries()
TU61	Norris::Lib::BusinessLayer::DataConsistency::checkValueType()
TU62	Norris::Lib::BusinessLayer::DataConsistency::checkDecimals()
TU63	Norris::Lib::BusinessLayer::DataConsistency::checkTableFormat()
TU64	Norris::Lib::BusinessLayer::DataConsistency::checkBounds()
TU65	Norris::NorrisApp::Services::TableSvc::fillData()
TU66	Norris::NorrisApp::Services::TableSvc::inPlaceUpd()
TU67	Norris::NorrisApp::Services::TableSvc::streamUpd()
TU68	Norris::NorrisApp::Services::SocketsSvc::open()
TU69	Norris::NorrisApp::Services::SocketsSvc::on()
TU70	Norris::NorrisApp::Services::FrontSvc::createRows()

Test	Metodi
TU71	Norris::NorrisApp::Services::MapSvc::cnvLatLong()
TU72	Norris::NorrisApp::Services::MapSvc::createPolyline()
TU73	Norris::NorrisApp::Services::MapSvc::createMarker()
TU74	Norris::NorrisApp::Services::MapSvc::buildPath()
TU75	Norris::NorrisApp::Services::MapSvc::buildLegend()
TU76	Norris::NorrisApp::Services::FirstConnectSvc::FirstConnectSvc()
TU77	Norris::NorrisApp::Services::MapSvc::updateMovie()
TU78	Norris::Lib::Utils::SocketService::isSocketNamespace()
TU79	Norris::Lib::DataLayer::PageModel::PageModel()
TU80	Norris::Lib::DataLayer::LineChartModel::LineChartModel()
TU81	Norris::Lib::DataLayer::BarChartModel::BarChartModel()
TU82	Norris::Lib::DataLayer::MapChartModel::MapChartModel()
TU83	Norris::Lib::DataLayer::TableModel::TableModel()
TU84	Norris::Lib::BusinessLayer::DataConsistency::labelConsistency()
TU85	Norris::Lib::BusinessLayer::DataConsistency::seriesConsistency()
TU86	Norris::Lib::BusinessLayer::DataConsistency::inPlaceTableOptionsConsistency()
TU87	Norris::Lib::BusinessLayer::DataConsistency::streamTableOptionsConsistency()
TU88	Norris::Lib::Utils::ColorManager::hexColorParse()
TU90	Norris::Lib::BusinessLayer::DataConsistency::checkPropertyShowing()

Tabella 51: Tracciamento test di unità-metodi

6.4 Tracciamento metodi-test di unità

Metodo	Test
Norris::Lib::BusinessLayer::ActiveResourcesController::storeGraph()	TU47
Norris::Lib::BusinessLayer::ActiveResourcesController::retrieveGraph()	TU42
Norris::Lib::BusinessLayer::ActiveResourcesController::retrievePage()	TU42
Norris::Lib::BusinessLayer::ActiveResourcesController::storePage()	TU41
Norris::Lib::BusinessLayer::BarChartController::createBarChart()	TU5
Norris::Lib::BusinessLayer::BarChartController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::BarChartController::updateInPlace()	TU11
Norris::Lib::BusinessLayer::DataConsistency::checkTemplate()	
Norris::Lib::BusinessLayer::DataConsistency::checkOrientation()	TU48
Norris::Lib::BusinessLayer::DataConsistency::checkLegendPosition()	TU49
Norris::Lib::BusinessLayer::DataConsistency::checkMapLegendPosition()	TU50
Norris::Lib::BusinessLayer::DataConsistency::checkMapZoom()	TU51
Norris::Lib::BusinessLayer::DataConsistency::pathMode()	TU52
Norris::Lib::BusinessLayer::DataConsistency::checkOrderBy()	TU53
Norris::Lib::BusinessLayer::DataConsistency::checkInsertPosition()	TU54
Norris::Lib::BusinessLayer::DataConsistency::checkDisplayedLines()	TU55
Norris::Lib::BusinessLayer::DataConsistency::checkColors()	TU56
Norris::Lib::BusinessLayer::DataConsistency::checkHex()	TU57
Norris::Lib::BusinessLayer::DataConsistency::checkColorArray()	TU58
Norris::Lib::BusinessLayer::DataConsistency::checkColorMatrix()	TU59
Norris::Lib::BusinessLayer::DataConsistency::checkSeries()	TU60
Norris::Lib::BusinessLayer::DataConsistency::checkValueType()	TU61
Norris::Lib::BusinessLayer::DataConsistency::checkDecimals()	TU62
Norris::Lib::BusinessLayer::DataConsistency::checkTableFormat()	TU63



Metodo	Test
Norris::Lib::BusinessLayer::DataConsistency::checkBounds()	TU64
Norris::Lib::BusinessLayer::DataConsistency::checkAllColors()	
Norris::Lib::BusinessLayer::DataConsistency::jsonConsistencyCheck()	
Norris::Lib::BusinessLayer::DataConsistency::inPlaceTableOptionsConsistency()	TU86
Norris::Lib::BusinessLayer::DataConsistency::streamTableOptionsConsistency()	TU87
Norris::Lib::BusinessLayer::DataConsistency::labelConsistency()	TU84
Norris::Lib::BusinessLayer::DataConsistency::seriesConsistency()	TU85
Norris::Lib::BusinessLayer::DataConsistency::checkUpdateHex()	
Norris::Lib::BusinessLayer::DataConsistency::checkPropertyShowing()	TU90
Norris::Lib::BusinessLayer::LineChartController::createLineChart()	TU5
Norris::Lib::BusinessLayer::LineChartController::updateStream()	TU11
Norris::Lib::BusinessLayer::LineChartController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::LineChartController::updateInPlace()	TU11
Norris::Lib::BusinessLayer::MapChartController::createMapChart()	TU5
Norris::Lib::BusinessLayer::MapChartController::updateMovie()	TU11
Norris::Lib::BusinessLayer::MapChartController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::MapChartController::updateInPlace()	TU11
Norris::Lib::BusinessLayer::MapChartController::pointsConsistency()	
Norris::Lib::BusinessLayer::PageController::createPage()	TU3 TU7 TU10 TU20 TU27
Norris::Lib::BusinessLayer::PageController::addGraphToPage()	TU9 TU13 TU37
Norris::Lib::BusinessLayer::PageController::getPageInfo()	TU1
Norris::Lib::BusinessLayer::PageController::createPageOptions()	TU45



Metodo	Test
Norris::Lib::BusinessLayer::SocketController::setSocket()	TU1
Norris::Lib::BusinessLayer::SocketController::sendUpdate()	TU30
Norris::Lib::BusinessLayer::SocketController::socketNamespace()	TU1
Norris::Lib::BusinessLayer::TableController::createTable()	TU5 TU16
Norris::Lib::BusinessLayer::TableController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::TableController::updateStream()	TU11 TU14 TU15
Norris::Lib::BusinessLayer::TableController::updateInPlace()	TU11
Norris::Lib::BusinessLayer::TableController::buildTempl()	
Norris::Lib::BusinessLayer::TableController::fillDefaultOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevOpts()	
Norris::Lib::BusinessLayer::TableController::fillDefaultColorOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevBgColorOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevFontColorOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevUpdateOpts()	
Norris::Lib::BusinessLayer::TableController::dataModelPush()	
Norris::Lib::BusinessLayer::TableController::fillClientColors()	
Norris::Lib::BusinessLayer::TableController::removeRow()	
Norris::Lib::BusinessLayer::TableController::getColors()	
Norris::Lib::DataLayer::ActiveResources::ActiveResources()	TU46
Norris::Lib::DataLayer::BarChartModel::BarChartModel()	TU81
Norris::Lib::DataLayer::LineChartModel::LineChartModel()	TU80
Norris::Lib::DataLayer::MapChartModel::MapChartModel()	TU82
Norris::Lib::DataLayer::PageModel::PageModel()	TU79
Norris::Lib::DataLayer::TableModel::TableModel()	TU83

Metodo	Test
Norris::Lib::PresentationLayer::BarChart::BarChart()	
Norris::Lib::PresentationLayer::BarChart::getChartInfo()	TU1
Norris::Lib::PresentationLayer::BarChart::updateInPlace()	TU11
Norris::Lib::PresentationLayer::LineChart::LineChart()	
Norris::Lib::PresentationLayer::LineChart::getChartInfo()	TU1
Norris::Lib::PresentationLayer::LineChart::updateInPlace()	TU11
Norris::Lib::PresentationLayer::LineChart::updateStream()	TU11
Norris::Lib::PresentationLayer::MapChart::MapChart()	
Norris::Lib::PresentationLayer::MapChart::getChartInfo()	TU1
Norris::Lib::PresentationLayer::MapChart::updateInPlace()	TU11
Norris::Lib::PresentationLayer::MapChart::updateMovie()	TU11
Norris::Lib::PresentationLayer::Norris::Norris()	TU43
Norris::Lib::PresentationLayer::Page::Page()	
Norris::Lib::PresentationLayer::Page::getPageInfo()	TU1
Norris::Lib::PresentationLayer::Page::addGraph()	TU9
Norris::Lib::PresentationLayer::PageRouter::PageRouter()	TU2 TU4 TU44
Norris::Lib::PresentationLayer::Table::Table()	
Norris::Lib::PresentationLayer::Table::getChartInfo()	TU1
Norris::Lib::PresentationLayer::Table::updateInPlace()	TU11
Norris::Lib::PresentationLayer::Table::updateStream()	TU11
Norris::Lib::Utils::ColorManager::ColorGenerator()	TU39
Norris::Lib::Utils::ColorManager::hexColorParse()	TU88
Norris::Lib::Utils::NorrisError::NorrisError()	TU6 TU29
Norris::Lib::Utils::NorrisError::toString()	



Metodo	Test
Norris::Lib::Utils::ProgressiveID::ProgressiveID()	TU32
Norris::Lib::Utils::SocketService::setSocket()	TU1
Norris::Lib::Utils::SocketService::connectionManager()	TU40
Norris::Lib::Utils::SocketService::sendUpdate()	TU30
Norris::Lib::Utils::SocketService::getSocketNamespace()	TU1
Norris::Lib::Utils::SocketService::isSocketNamespace()	TU78
Norris::NorrisApp::Controllers::BarLineChartCtrl::BarLineChartCtrl()	
Norris::NorrisApp::Controllers::FrontCtrl::FrontCtrl()	
Norris::NorrisApp::Controllers::MapChartCtrl::MapChartCtrl()	
Norris::NorrisApp::Controllers::TableCtrl::TableCtrl()	
Norris::NorrisApp::Services::BarLineSvc::BarLineSvc()	
Norris::NorrisApp::Services::BarLineSvc::fillLineData()	
Norris::NorrisApp::Services::BarLineSvc::setColors()	
Norris::NorrisApp::Services::BarLineSvc::setOpts()	
Norris::NorrisApp::Services::ColorsSvc::ColorsSvc()	
Norris::NorrisApp::Services::ColorsSvc::componentToHex()	
Norris::NorrisApp::Services::ColorsSvc::rgbToHex()	
Norris::NorrisApp::Services::FirstConnectSvc::FirstConnectSvc()	TU76
Norris::NorrisApp::Services::FrontSvc::FrontSvc()	
Norris::NorrisApp::Services::FrontSvc::createRows()	TU70
Norris::NorrisApp::Services::MapSvc::MapSvc()	
Norris::NorrisApp::Services::MapSvc::cnvLatLong()	TU71
Norris::NorrisApp::Services::MapSvc::setPathMode()	
Norris::NorrisApp::Services::MapSvc::setColors()	
Norris::NorrisApp::Services::MapSvc::createPolyline()	TU72
Norris::NorrisApp::Services::MapSvc::createMarker()	TU73

Metodo	Test
Norris::NorrisApp::Services::MapSvc::buildPath()	TU74
Norris::NorrisApp::Services::MapSvc::buildLegend()	TU75
Norris::NorrisApp::Services::MapSvc::updateMovie()	TU77
Norris::NorrisApp::Services::SocketsSvc::SocketsSvc()	
Norris::NorrisApp::Services::SocketsSvc::open()	TU68
Norris::NorrisApp::Services::SocketsSvc::on()	TU69
Norris::NorrisApp::Services::SocketsSvc::emit()	
Norris::NorrisApp::Services::TableSvc::TableSvc()	
Norris::NorrisApp::Services::TableSvc::fillData()	TU65
Norris::NorrisApp::Services::TableSvc::inPlaceUpd()	TU66
Norris::NorrisApp::Services::TableSvc::streamUpd()	TU67

Tabella 52: Tracciamento metodi-test di unità