

# NORRIS Framework



FLAMETECH Inc.

## Specifica Tecnica

### Informazioni sul documento

<b>Versione</b>	3.0.0
<b>Redazione</b>	Faggin Andrea Meneguzzo Francesco
<b>Verifica</b>	Sartor Michele
<b>Responsabile</b>	Merlo Gianluca
<b>Uso</b>	Esterno
<b>Lista di distribuzione</b>	<b>FlameTech Inc.</b> Prof. Vardanega Tullio Prof. Cardin Riccardo

### Descrizione

Specifica tecnica e architettura del prodotto sviluppato dal gruppo **FlameTech Inc.** per la realizzazione del progetto Norris.

Versione	Modifica	Autore	Ruolo	Data	Stato
3.0.0	Approvazione documento	Merlo Gianluca	Responsabile	2015/06/12	Approvato
2.1.0	Verifica documento	Sartor Michele	Verificatore	2015/06/12	Verificato
2.0.5	Aggiunte descrizioni di dettaglio riguardanti l'integrazione delle tecnologie con il prodotto nella sezione 2	Meneguzzo Francesco	Progettista	2015/06/09	In Lavorazione
2.0.4	Approfondimento descrizione pattern MVC nella sezione A	Faggin Andrea	Progettista	2015/06/09	In lavorazione
2.0.3	Rotazione fig. 10 per un leggero miglioramento della leggibilità	Faggin Andrea	Progettista	2015/06/08	In Lavorazione
2.0.2	Adeguate spiegazione di dettaglio del singleton ProgressiveID nel documento DefinizioneDi-Prodotto_ver2.0.0 in sede di descrizione della componente	Meneguzzo Francesco	Progettista	2015/06/08	In Lavorazione
2.0.1	Aggiunta spiegazione dipendenza circolare nella Fig. 1	Faggin Andrea	Progettista	2015/06/08	In Lavorazione
2.0.0	Approvazione documento	Cardin Andrea	Responsabile	2015/05/21	Approvato
1.2.0	Terminata verifica delle rimanenti sezioni e dell'applicazione delle correzioni	Merlo Gianluca	Verificatore	2015/05/20	Verificato
1.1.1	Corretti alcuni errori di ortografia rilevati tramite verifica	Zanetti Davide	Progettista	2015/05/19	In Lavorazione
1.1.0	Verifica sezioni 1, 2, 3 e 4 del documento	Persegona Mattia	Verificatore	2015/05/19	Verificato
1.0.7	Termine adeguamento descrizione componenti in seguito a ulteriore progettazione	Sartor Michele	Progettista	2015/05/11	In Lavorazione



Versione	Modifica	Autore	Ruolo	Data	Stato
1.0.6	Applicazione correzioni RP - Contestualizzazione dei pattern utilizzati con inserimento di diagrammi delle classi dedicati	Faggin Andrea	Progettista	2015/05/08	In Lavorazione
1.0.5	Inizio adeguamento descrizione componenti in seguito a ulteriore progettazione	Sartor Michele	Progettista	2015/05/08	In Lavorazione
1.0.4	Applicazione correzioni RP - Adeguamento sezione 5 e corretta sintassi dei messaggi trovati per le figure segnalate	Zanetti Davide	Progettista	2015/05/07	In Lavorazione
1.0.3	Applicazione correzioni RP - Inseriti alcuni svantaggi delle tecnologie utilizzate	Meneguzzo Francesco	Progettista	2015/05/07	In Lavorazione
1.0.2	Applicazione correzioni RP - Dettagliata maggiormente la modalità di identificazione delle risorse REST	Faggin Andrea	Progettista	2015/05/07	In Lavorazione
1.0.1	Sostituite tecnologie Chart.js e DataTables con Google Charts	Meneguzzo Francesco	Progettista	2015/05/07	In Lavorazione
1.0.0	Approvazione Documento	Persegona Mattia	Responsabile	2015/04/10	Approvato
0.2.0	Verifica Documento	Meneguzzo Francesco	Verificatore	2015/04/08	Verificato
0.1.1	Correzione errori grammaticali	Faggin Andrea	Progettista	2015/04/08	In Lavorazione
0.1.0	Verifica Documento	Meneguzzo Francesco	Verificatore	2015/04/07	Verificato
0.0.11	Stesa sezione Stime di Fattibilità	Merlo Gianluca	Progettista	2015/03/30	In Lavorazione
0.0.10	Stesa sezione Tracciamento	Cardin Andrea	Progettista	2015/03/24	In Lavorazione
0.0.9	Stesa sezione Design Pattern	Zanetti Davide	Progettista	2015/03/23	In Lavorazione

Versione	Modifica	Autore	Ruolo	Data	Stato
0.0.8	Stesa sezione Diagrammi di Attività	Sartor Michele	Progettista	2015/03/23	In Lavorazione
0.0.7	Termine stesura sezione Componenti e Classi	Merlo Gianluca	Progettista	2015/03/20	In Lavorazione
0.0.6	Stesa Appendice A	Zanetti Davide	Progettista	2015/03/17	In Lavorazione
0.0.5	Inizio stesura sezione Componenti e Classi	Merlo Gianluca	Progettista	2015/03/16	In Lavorazione
0.0.4	Stesa sezione Descrizione Architettura	Faggin Andrea	Progettista	2015/03/16	In Lavorazione
0.0.3	Stesa sezione Tecnologie Utilizzate	Cardin Andrea	Progettista	2015/03/11	In Lavorazione
0.0.2	Stesa sezione Introduzione	Sartor Michele	Progettista	2015/03/10	In Lavorazione
0.0.1	Inizio stesura scheletro documento	Faggin Andrea	Progettista	2015/03/10	In Lavorazione

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento . . . . .	1
1.2	Scopo del prodotto . . . . .	1
1.3	Glossario . . . . .	1
1.4	Riferimenti . . . . .	1
1.4.1	Normativi . . . . .	1
1.4.2	Informativi . . . . .	1
<b>2</b>	<b>Tecnologie utilizzate</b>	<b>2</b>
2.1	<i>Node.js<sub>G</sub></i> . . . . .	2
2.2	<i>Express<sub>G</sub></i> . . . . .	2
2.3	<i>AngularJS<sub>G</sub></i> . . . . .	3
2.4	<i>Google Charts<sub>G</sub></i> . . . . .	3
2.5	<i>Google Maps API<sub>G</sub></i> . . . . .	4
2.6	<i>MPAndroidChart<sub>G</sub></i> . . . . .	4
2.7	<i>Socket.io<sub>G</sub></i> . . . . .	4
2.8	<i>JSON<sub>G</sub></i> . . . . .	5
2.9	<i>HTML5<sub>G</sub></i> . . . . .	5
<b>3</b>	<b>Descrizione architettura</b>	<b>6</b>
3.1	Metodo e formalismo di specifica . . . . .	6
3.2	Architettura generale . . . . .	6
3.2.1	Interfaccia REST-like . . . . .	6
3.2.2	Norris . . . . .	8
3.2.2.1	Lib . . . . .	9
3.2.2.2	NorrisApp . . . . .	12
3.2.3	Applicativo <i>Android<sub>G</sub></i> . . . . .	13
<b>4</b>	<b>Componenti e classi</b>	<b>14</b>
4.1	Componente Norris <i>framework<sub>G</sub></i> . . . . .	14
4.1.1	Norris . . . . .	14
4.1.1.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	14
4.1.1.1.1	Descrizione . . . . .	14
4.1.1.1.2	<i>Package<sub>G</sub></i> contenuti . . . . .	14
4.1.2	Norris::Lib . . . . .	15
4.1.2.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	16
4.1.2.1.1	Descrizione . . . . .	16
4.1.2.1.2	<i>Package<sub>G</sub></i> contenuti . . . . .	16
4.1.3	Norris::Lib::BusinessLayer . . . . .	16
4.1.3.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	16
4.1.3.1.1	Descrizione . . . . .	16
4.1.3.2	Classi . . . . .	16
4.1.3.2.1	Norris::Lib::BusinessLayer::ActiveResourcesController . . . . .	16
4.1.3.2.2	Norris::Lib::BusinessLayer::BarChartController . . . . .	17
4.1.3.2.3	Norris::Lib::BusinessLayer::DataConsistency . . . . .	18
4.1.3.2.4	Norris::Lib::BusinessLayer::LineChartController . . . . .	19
4.1.3.2.5	Norris::Lib::BusinessLayer::MapChartController . . . . .	20
4.1.3.2.6	Norris::Lib::BusinessLayer::PageController . . . . .	21



4.1.3.2.7	Norris::Lib::BusinessLayer::SocketController . . . . .	22
4.1.3.2.8	Norris::Lib::BusinessLayer::TableController . . . . .	23
4.1.4	Norris::Lib::DataLayer . . . . .	24
4.1.4.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	24
4.1.4.1.1	Descrizione . . . . .	24
4.1.4.2	Classi . . . . .	24
4.1.4.2.1	Norris::Lib::DataLayer::ActiveResources . . . . .	24
4.1.4.2.2	Norris::Lib::DataLayer::BarChartModel . . . . .	25
4.1.4.2.3	Norris::Lib::DataLayer::LineChartModel . . . . .	25
4.1.4.2.4	Norris::Lib::DataLayer::MapChartModel . . . . .	26
4.1.4.2.5	Norris::Lib::DataLayer::PageModel . . . . .	26
4.1.4.2.6	Norris::Lib::DataLayer::TableModel . . . . .	26
4.1.5	Norris::Lib::PresentationLayer . . . . .	27
4.1.5.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	27
4.1.5.1.1	Descrizione . . . . .	27
4.1.5.2	Classi . . . . .	27
4.1.5.2.1	Norris::Lib::PresentationLayer::BarChart . . . . .	27
4.1.5.2.2	Norris::Lib::PresentationLayer::LineChart . . . . .	28
4.1.5.2.3	Norris::Lib::PresentationLayer::MapChart . . . . .	28
4.1.5.2.4	Norris::Lib::PresentationLayer::Norris . . . . .	29
4.1.5.2.5	Norris::Lib::PresentationLayer::Page . . . . .	30
4.1.5.2.6	Norris::Lib::PresentationLayer::PageRouter . . . . .	30
4.1.5.2.7	Norris::Lib::PresentationLayer::Table . . . . .	30
4.1.6	Norris::Lib::Utils . . . . .	31
4.1.6.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	31
4.1.6.1.1	Descrizione . . . . .	31
4.1.6.2	Classi . . . . .	31
4.1.6.2.1	Norris::Lib::Utils::ColorManager . . . . .	31
4.1.6.2.2	Norris::Lib::Utils::NorrisError . . . . .	32
4.1.6.2.3	Norris::Lib::Utils::ProgressiveID . . . . .	33
4.1.6.2.4	Norris::Lib::Utils::SocketService . . . . .	33
4.1.7	Norris::NorrisApp . . . . .	34
4.1.7.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	34
4.1.7.1.1	Descrizione . . . . .	34
4.1.7.1.2	<i>Package<sub>G</sub></i> contenuti . . . . .	34
4.1.8	Norris::NorrisApp::Controllers . . . . .	35
4.1.8.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	35
4.1.8.1.1	Descrizione . . . . .	35
4.1.8.2	Classi . . . . .	35
4.1.8.2.1	Norris::NorrisApp::Controllers::BarLineChartCtrl . . . . .	35
4.1.8.2.2	Norris::NorrisApp::Controllers::FrontCtrl . . . . .	36
4.1.8.2.3	Norris::NorrisApp::Controllers::MapChartCtrl . . . . .	37
4.1.8.2.4	Norris::NorrisApp::Controllers::TableCtrl . . . . .	37
4.1.9	Norris::NorrisApp::Model . . . . .	38
4.1.9.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	38
4.1.9.1.1	Descrizione . . . . .	38
4.1.9.2	Classi . . . . .	38
4.1.9.2.1	Norris::NorrisApp::Model::BarChartMdl . . . . .	38
4.1.9.2.2	Norris::NorrisApp::Model::FrontMdl . . . . .	39

4.1.9.2.3	Norris::NorrisApp::Model::LineChartMdl . . . . .	39
4.1.9.2.4	Norris::NorrisApp::Model::MapChartMdl . . . . .	39
4.1.9.2.5	Norris::NorrisApp::Model::TableMdl . . . . .	40
4.1.10	Norris::NorrisApp::Services . . . . .	40
4.1.10.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	40
4.1.10.1.1	Descrizione . . . . .	40
4.1.10.2	Classi . . . . .	41
4.1.10.2.1	Norris::NorrisApp::Services::BarLineSvc . . . . .	41
4.1.10.2.2	Norris::NorrisApp::Services::ColorsSvc . . . . .	41
4.1.10.2.3	Norris::NorrisApp::Services::FirstConnectSvc . . . . .	42
4.1.10.2.4	Norris::NorrisApp::Services::FrontSvc . . . . .	42
4.1.10.2.5	Norris::NorrisApp::Services::MapSvc . . . . .	42
4.1.10.2.6	Norris::NorrisApp::Services::SocketsSvc . . . . .	43
4.1.10.2.7	Norris::NorrisApp::Services::TableSvc . . . . .	43
4.1.11	Norris::NorrisApp::Views . . . . .	44
4.1.11.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	44
4.1.11.1.1	Descrizione . . . . .	44
4.1.11.2	Classi . . . . .	44
4.1.11.2.1	Norris::NorrisApp::Views::BarChartView . . . . .	44
4.1.11.2.2	Norris::NorrisApp::Views::Index . . . . .	44
4.1.11.2.3	Norris::NorrisApp::Views::LineChartView . . . . .	45
4.1.11.2.4	Norris::NorrisApp::Views::MapChartView . . . . .	45
4.1.11.2.5	Norris::NorrisApp::Views::TableView . . . . .	45
4.2	Componente applicazione <i>Android<sub>G</sub></i> . . . . .	46
4.2.1	AndroidApp . . . . .	46
4.2.1.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	46
4.2.1.1.1	Descrizione . . . . .	46
4.2.1.1.2	<i>Package<sub>G</sub></i> contenuti . . . . .	46
4.2.2	AndroidApp::Activities . . . . .	47
4.2.2.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	47
4.2.2.1.1	Descrizione . . . . .	47
4.2.2.2	Classi . . . . .	47
4.2.2.2.1	AndroidApp::Activities::ErrorActivity . . . . .	47
4.2.2.2.2	AndroidApp::Activities::GraphActivity . . . . .	47
4.2.2.2.3	AndroidApp::Activities::MainActivity . . . . .	48
4.2.2.2.4	AndroidApp::Activities::PageActivity . . . . .	48
4.2.2.2.5	AndroidApp::Activities::RecentActivity . . . . .	49
4.2.2.2.6	AndroidApp::Activities::RequestActivity . . . . .	49
4.2.2.2.7	AndroidApp::Activities::SettingsActivity . . . . .	50
4.2.3	AndroidApp::AppModel . . . . .	50
4.2.3.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	50
4.2.3.1.1	Descrizione . . . . .	50
4.2.3.2	Classi . . . . .	50
4.2.3.2.1	AndroidApp::AppModel::Adapter . . . . .	50
4.2.3.2.2	AndroidApp::AppModel::Cache . . . . .	51
4.2.3.2.3	AndroidApp::AppModel::MainObj . . . . .	51
4.2.3.2.4	AndroidApp::AppModel::Preferences . . . . .	51
4.2.4	AndroidApp::Layouts . . . . .	52
4.2.4.1	Informazioni sul <i>package<sub>G</sub></i> . . . . .	52



4.2.4.1.1	Descrizione . . . . .	52
4.2.4.2	Classi . . . . .	52
4.2.4.2.1	AndroidApp::Layouts::GraphLayout . . . . .	52
4.2.4.2.2	AndroidApp::Layouts::MainLayout . . . . .	53
4.2.4.2.3	AndroidApp::Layouts::PageLayout . . . . .	53
4.2.4.2.4	AndroidApp::Layouts::RecentLayout . . . . .	54
4.2.4.2.5	AndroidApp::Layouts::SettingsLayout . . . . .	54
<b>5</b>	<b>Comunicazione <i>client<sub>G</sub>-server<sub>G</sub></i></b>	<b>55</b>
5.1	Prima richiesta di pagina . . . . .	55
5.2	<i>Routing<sub>G</sub></i> . . . . .	56
5.3	Inoltro notifiche <i>push<sub>G</sub></i> . . . . .	56
<b>6</b>	<b>Diagrammi di attività</b>	<b>58</b>
6.1	Funzionalità Sviluppatore . . . . .	58
6.1.1	Attività principali . . . . .	59
6.1.2	Attività di creazione . . . . .	60
6.1.3	Attività di creazione <i>Bar Chart<sub>G</sub></i> . . . . .	61
6.1.4	Attività di creazione <i>Line Chart<sub>G</sub></i> . . . . .	63
6.1.5	Attività di creazione <i>Map Chart<sub>G</sub></i> . . . . .	65
6.1.6	Attività di creazione <i>Table<sub>G</sub></i> . . . . .	66
6.1.7	Attività di modifica . . . . .	68
6.1.8	Attività di update grafico <i>Bar Chart<sub>G</sub></i> . . . . .	69
6.1.9	Attività di update grafico <i>Line Chart<sub>G</sub></i> . . . . .	70
6.1.10	Attività di update grafico <i>Map Chart<sub>G</sub></i> . . . . .	70
6.1.11	Attività di update grafico <i>Table<sub>G</sub></i> . . . . .	71
6.2	Funzionalità Utente . . . . .	72
6.2.1	Funzionalità generali . . . . .	72
6.2.2	App <i>Android<sub>G</sub></i> . . . . .	73
6.2.3	Caso applicativo APS Holding . . . . .	75
<b>7</b>	<b><i>Design Pattern<sub>G</sub></i></b>	<b>76</b>
7.1	<i>Design pattern<sub>G</sub></i> architetturali . . . . .	76
7.1.1	<i>MVC<sub>G</sub></i> . . . . .	76
7.2	<i>Design pattern<sub>G</sub></i> creazionali . . . . .	77
7.2.1	<i>Singleton<sub>G</sub></i> . . . . .	77
<b>8</b>	<b>Stime di fattibilità e di bisogno di risorse</b>	<b>78</b>
<b>9</b>	<b>Tracciamento</b>	<b>79</b>
9.1	Tracciamento componenti - requisiti . . . . .	79
9.2	Tracciamento requisiti - componenti . . . . .	82
<b>A</b>	<b>Descrizione <i>Design pattern<sub>G</sub></i></b>	<b>85</b>
A.1	<i>Design pattern<sub>G</sub></i> architetturali . . . . .	85
A.1.1	<i>MVCS<sub>G</sub></i> . . . . .	85
A.1.2	<i>MVC<sub>G</sub></i> . . . . .	85
A.2	<i>Design pattern<sub>G</sub></i> creazionali . . . . .	85
A.2.1	<i>Singleton<sub>G</sub></i> . . . . .	85





## Elenco delle tabelle

2	Chiamate URI . . . . .	7
3	Tabella Tracciamento Componenti - Requisiti . . . . .	81
4	Tabella Tracciamento Requisiti - Componenti . . . . .	84

## Elenco delle figure

1	Diagramma dei <i>package<sub>G</sub></i> Norris . . . . .	8
2	Diagramma del <i>package<sub>G</sub></i> Lib . . . . .	9
3	Il <i>package<sub>G</sub></i> Presentation Layer della libreria Norris. . . . .	10
4	Il <i>package<sub>G</sub></i> Business Layer della libreria Norris. . . . .	10
5	Il <i>package<sub>G</sub></i> Data Layer della libreria Norris. . . . .	11
6	Il <i>package<sub>G</sub></i> Utils della libreria Norris. . . . .	11
7	Diagramma dei <i>package<sub>G</sub></i> di NorrisApp . . . . .	12
8	Diagramma dei <i>package<sub>G</sub></i> applicazione <i>Android<sub>G</sub></i> . . . . .	13
9	Diagramma del <i>package<sub>G</sub></i> Norris . . . . .	14
10	Diagramma delle classi Lib . . . . .	15
11	Diagramma delle classi BusinessLayer . . . . .	16
12	Diagramma delle classi DataLayer . . . . .	24
13	Diagramma delle classi PresentationLayer . . . . .	27
14	Diagramma delle classi Utils . . . . .	31
15	Diagramma delle classi NorrisApp . . . . .	34
16	Diagramma delle classi Controllers . . . . .	35
17	Diagramma delle classi Model . . . . .	38
18	Diagramma delle classi Services . . . . .	40
19	Diagramma delle classi Views . . . . .	44
20	Diagramma delle classi AndroidApp . . . . .	46
21	Componente Activities . . . . .	47
22	Componente AppModel . . . . .	50
23	Componente Layouts . . . . .	52
24	Diagramma di sequenza - Prima richiesta di pagina . . . . .	55
25	Diagramma di sequenza - <i>Routing<sub>G</sub></i> di una richiesta . . . . .	56
26	Diagramma di sequenza - Aggiornamento di un grafico <i>Bar Chart<sub>G</sub></i> . . . . .	56
27	Diagramma di attività - Attività principali sviluppatore . . . . .	59
28	Diagramma di attività - Funzionalità di creazione . . . . .	60
29	Diagramma di attività - Funzionalità di creazione <i>Bar Chart<sub>G</sub></i> . . . . .	61
30	Diagramma di attività - Funzionalità di creazione <i>Line Chart<sub>G</sub></i> . . . . .	63
31	Diagramma di attività - Funzionalità di creazione <i>Map Chart<sub>G</sub></i> . . . . .	65
32	Diagramma di attività - Funzionalità di creazione <i>Table<sub>G</sub></i> . . . . .	66
33	Diagramma di attività - Attività di modifica . . . . .	68
34	Diagramma di attività - Attività di update . . . . .	68
35	Diagramma di attività - Update grafico <i>Bar Chart<sub>G</sub></i> . . . . .	69
36	Diagramma di attività - Update grafico <i>Line Chart<sub>G</sub></i> . . . . .	70
37	Diagramma di attività - Update grafico <i>Map Chart<sub>G</sub></i> . . . . .	70
38	Diagramma di attività - Update grafico <i>Table<sub>G</sub></i> . . . . .	71
39	Diagramma di attività - Funzionalità . . . . .	72
40	Diagramma di attività - App <i>Android<sub>G</sub></i> . . . . .	73
41	Diagramma di attività - Caso applicativo APS Holding . . . . .	75
42	Un immagine del <i>Design Pattern<sub>G</sub></i> architetturale <i>MVC<sub>G</sub></i> . . . . .	77
43	Le classi che implementano il <i>design pattern<sub>G</sub></i> creazionale <i>Singleton<sub>G</sub></i> . . . . .	77

## 1 Introduzione

### 1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello del progetto Norris.

All'interno del documento verrà presentata l'architettura generale secondo la quale saranno organizzate le varie componenti software e i *design pattern*<sub>G</sub> adoperati per la creazione del prodotto. Sarà, inoltre, dettagliato il tracciamento tra le componenti software individuate ed i requisiti.

### 1.2 Scopo del prodotto

Lo scopo del prodotto è la realizzazione di un *framework*<sub>G</sub> per *Node.js*<sub>G</sub>, compatibile con l'utilizzo standard dei *middleware*<sub>G</sub> di *Express*<sub>G</sub> 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*<sub>G</sub>).

### 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>;
- **Verbale d'incontro con il Proponente** in data 2015/03/31.

#### 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);
- Node.js - Marc Wandschneider - 1a edizione (2013).

## 2 Tecnologie utilizzate

In questa sezione vengono descritte le tecnologie su cui si basa lo sviluppo del progetto. Per ognuna di esse vengono indicati la ragione del loro utilizzo, l'ambito di utilizzo delle tecnologia ed i vantaggi/svantaggi che ne derivano.

All'interno del *Glossario\_ver5.0.0* saranno presenti ulteriori dettagli descrittivi sulle tecnologie.

### 2.1 *Node.js<sub>G</sub>*

L'utilizzo di *Node.js<sub>G</sub>* è vincolato dalla richiesta del Proponente (requisito: RAV1); questo *framework<sub>G</sub>* verrà utilizzato per lo sviluppo della componente *back-end<sub>G</sub>*.

#### Vantaggi:

- *Node.js<sub>G</sub>* è basato su un modello *event-driven<sub>G</sub>* con I/O non bloccante. Tale approccio risulta molto efficiente in situazioni critiche di elevato traffico di rete e in applicazioni real-time;
- Supporta l'organizzazione modulare dell'architettura tramite *npm<sub>G</sub>*. Quest'ultimo infatti permette di importare facilmente i moduli necessari e di combinarli tra loro.

#### Svantaggi:

- Il principale svantaggio di questa tecnologia è il suo essere ancora in versione alpha, questo comporta documentazione facilmente deprecata e difficoltà ulteriori per il suo utilizzo.

#### Contesto d'utilizzo:

- Il *framework<sub>G</sub>* viene utilizzato per lo sviluppo del *back-end<sub>G</sub>*.

### 2.2 *Express<sub>G</sub>*

L'utilizzo di *Express<sub>G</sub>* è vincolato dalla richiesta del Proponente (requisito: RAV2); questo *framework<sub>G</sub>* verrà utilizzato per la realizzazione dell'infrastruttura per la componente *back-end<sub>G</sub>*.

#### Vantaggi:

- *Express<sub>G</sub>* semplifica l'uso di *Node.js<sub>G</sub>* e offre una migliore implementazione di alcuni aspetti chiave importanti per lo sviluppo del progetto;
- La creazione di *API<sub>G</sub>* è resa più facile e veloce grazie ai numerosi moduli e metodi di utilità messi a disposizione da *Express<sub>G</sub>*.

#### Svantaggi:

- L'unico svantaggio che presenta questa tecnologia è l'impossibilità di inviare più oggetti in seguito ad una richiesta Get. Tuttavia questo non andrà ad incidere particolarmente con il suo utilizzo.

#### Contesto d'utilizzo:

- Il *framework<sub>G</sub>* viene utilizzato per la creazione di un sub-routing interno per gestire le richieste GET.

### 2.3 *AngularJS<sub>G</sub>*

Si è deciso di utilizzare *AngularJS<sub>G</sub>* come *framework<sub>G</sub>* per lo sviluppo della componente *front-end<sub>G</sub>*.

#### Vantaggi:

- Incorpora il *MVC<sub>G</sub>* lato *client<sub>G</sub>*; questo rende più facile l'organizzazione della logica dell'architettura complessiva;
- **Approccio dichiarativo:** l'integrazione di questo *framework<sub>G</sub>* avviene direttamente nel codice *HTML<sub>G</sub>* in modo chiaro e conciso. In particolare, i due aspetti più importanti di questo approccio sono:
  - Creazione di viste dinamiche che effettuano l'aggiornamento automatico tramite *two-way data binding<sub>G</sub>*, rimuovendo allo sviluppatore parte della complessità della gestione di questo aspetto;
  - Gestione delle dipendenze effettuata in maniera dichiarativa e quindi rende facilmente isolabili i comportamenti e le responsabilità dei singoli componenti, semplificando le procedure di test.

#### Svantaggi:

- L'utilizzo risulta particolarmente complesso.

#### Contesto d'utilizzo:

- Il *framework<sub>G</sub>* viene utilizzato per lo sviluppo del *front-end<sub>G</sub>*.

### 2.4 *Google Charts<sub>G</sub>*

Si è deciso di utilizzare *Google Charts<sub>G</sub>* come libreria grafica per la creazione di tutte le tipologie di grafico, esclusa la tipologia *Map Chart<sub>G</sub>*, nella componente *front-end<sub>G</sub>*.

#### Vantaggi:

- Permette di implementare tutte le funzioni richieste dal capitolato;
- È presente una direttiva per *AngularJS<sub>G</sub>*;
- È disponibile una buona documentazione.

#### Svantaggi:

- A cause delle sue numerose opzioni risulta complesso l'utilizzo.

#### Contesto d'utilizzo:

- Il *framework<sub>G</sub>* viene utilizzato per la creazione dei grafici di tipo *Bar Chart<sub>G</sub>*, *Line Chart<sub>G</sub>* e *Table<sub>G</sub>* dal lato *front-end<sub>G</sub>*.

## 2.5 *Google Maps API<sub>G</sub>*

Si è deciso di utilizzare *Google Maps API<sub>G</sub>* come libreria grafica per la creazione di grafici di tipo *Map Chart<sub>G</sub>* nella componente *front-end<sub>G</sub>* e nell'applicativo *Android<sub>G</sub>*.

### Vantaggi:

- Diffusione ampia, largo supporto e documentazione ricca; questo rende preferibile l'utilizzo di questa libreria rispetto ad eventuali alternative.
- Presenta sia una versione *JavaScript<sub>G</sub>* da utilizzare per la componente *front-end<sub>G</sub>*, sia una versione *Java<sub>G</sub>* per l'applicativo *Android<sub>G</sub>* utilizzabile ancora più semplicemente in quanto praticamente nativa.

### Svantaggio:

- L'utilizzo è gratuito esclusivamente con un volume di utenza ridotto, in particolare la richiesta giornaliera di caricamento delle mappe deve essere inferiore a 25000.

### Contesto d'utilizzo:

- Il *framework<sub>G</sub>* viene utilizzato per la creazione dei grafici di tipo *Map Chart<sub>G</sub>* dal lato *front-end<sub>G</sub>*.

Tuttavia, data la natura principalmente dimostrativa del progetto, è stato stimato che un tale limite al volume di richieste è accettabile in quanto sarà difficilmente raggiungibile.

Nel caso si verifichi un futuro utilizzo di Norris che risulti particolarmente scalabile, fino al superamento del limite di richieste, sarà compito dello sviluppatore interessato occuparsi di questo aspetto.

## 2.6 *MPAndroidChart<sub>G</sub>*

Si è deciso di utilizzare *MPAndroidChart<sub>G</sub>* come libreria grafica per la creazione di grafici di tipo *Bar Chart<sub>G</sub>* e *Line Chart<sub>G</sub>* nell'applicativo *Android<sub>G</sub>*.

### Vantaggi:

- Fornisce numerose funzionalità riguardanti la veste grafica e l'interazione con i grafici creati.
- Compatibile con le versioni di *Android<sub>G</sub>* richieste e semplice integrazione.

## 2.7 *Socket.io<sub>G</sub>*

L'utilizzo di *Socket.io<sub>G</sub>* è vincolato dalla richiesta del Proponente (requisito: RAV3); questa libreria verrà utilizzata per la realizzazione della componente che gestisce le notifiche *push<sub>G</sub>*.

### Vantaggi:

- Semplifica le conoscenze necessarie per la gestione del protocollo *WebSocket<sub>G</sub>*;
- Fornisce una libreria *JavaScript<sub>G</sub>* per la componente *back-end<sub>G</sub>* e una per la parte *front-end<sub>G</sub>*, il che lo rende facilmente integrabile nel progetto.

- Sono disponibili diverse librerie *Java<sub>G</sub>* per l'utilizzo in applicazioni *Android<sub>G</sub>*, il che rimuove eventuali problemi di integrazione di librerie scritte in linguaggi diversi.

**Svantaggi:**

- Documentazione ridotta e per gran parte deprecata.

**Contesto d'utilizzo:**

- Il *framework<sub>G</sub>* viene utilizzato per l'attuazione delle notifiche *push<sub>G</sub>* da parte del *back-end<sub>G</sub>* verso il *front-end<sub>G</sub>*.

## 2.8 *JSON<sub>G</sub>*

Si è deciso di utilizzare *JSON<sub>G</sub>* come formato per lo scambio dei dati tra *back-end<sub>G</sub>* e *front-end<sub>G</sub>* o applicativo *Android<sub>G</sub>*.

**Vantaggi:**

- *JavaScript<sub>G</sub>* rende l'utilizzo di *JSON<sub>G</sub>* semplice ed immediato, praticamente escludendo ogni altra scelta da questa decisione.

**Contesto d'utilizzo:**

- Il formato viene adoperato per l'invio dei dati all'interno di Norris.

## 2.9 *HTML5<sub>G</sub>*

Si è deciso di utilizzare *HTML5<sub>G</sub>* come linguaggio per la creazione della struttura delle pagine web.

**Vantaggi:**

- Consente di integrare facilmente le funzionalità di *AngularJS<sub>G</sub>*;
- Strutturato per offrire codice più pulito rispetto alle versioni precedenti;
- Compatibile con le versioni dei browser richiesti (requisiti: RAV4 e RAV5).

**Contesto d'utilizzo:**

- Il linguaggio è stato adoperato allo scopo di creare le view del *front-end<sub>G</sub>*.

## 3 Descrizione architettura

### 3.1 Metodo e formalismo di specifica

Nell'esposizione dell'architettura di Norris si procederà con un approccio *top-down<sub>G</sub>*, descrivendo l'architettura iniziando dal generale ed andando al particolare. Si procederà quindi alla descrizione dei *package<sub>G</sub>* 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. Successivamente si illustreranno degli esempi di uso dei *design pattern<sub>G</sub>* nell'architettura del sistema, rimandando la spiegazione generale degli stessi all'appendice A - *Descrizione Design pattern<sub>G</sub>*

Per i diagrammi di *package<sub>G</sub>*, di classe e di attività si utilizzerà il formalismo *UML<sub>G</sub>* 2.x. Nel riportare i diagrammi di *package<sub>G</sub>* e di classe si farà uso, dove appropriato, dei colori per aiutare la distinzione tra componenti diversi. Si noti in particolare che le classi di colore verde appartengono a librerie e componenti esterni e sono quindi da considerarsi fuori dai *package<sub>G</sub>*, anche se riportate all'interno in alcuni diagrammi per maggior chiarezza.

Nel trattare i componenti, si chiarisce che sono da intendersi come *package<sub>G</sub>* 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<sub>G</sub>* web, costituita dal browser degli utenti che visualizzeranno le pagine *front-end<sub>G</sub>*, una componente *AndroidApp* che costituisce l'applicazione per *smartphone<sub>G</sub>* *Android<sub>G</sub>* e una componente web *server<sub>G</sub>* costituita dalla sezione *back-end<sub>G</sub>* di Norris.

#### 3.2.1 Interfaccia REST-like

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

- Semplicità di utilizzo;
- Facile integrazione con i *framework<sub>G</sub>* esistenti (*AngularJS<sub>G</sub>* e *Express<sub>G</sub>*);
- Indipendenza dal linguaggio di programmazione utilizzato.

*REST<sub>G</sub>* utilizza il concetto di risorsa, ovvero un aggregato di dati con un nome (*URI<sub>G</sub>*) e una rappresentazione, su cui è possibile invocare le operazioni *CRUD<sub>G</sub>* 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

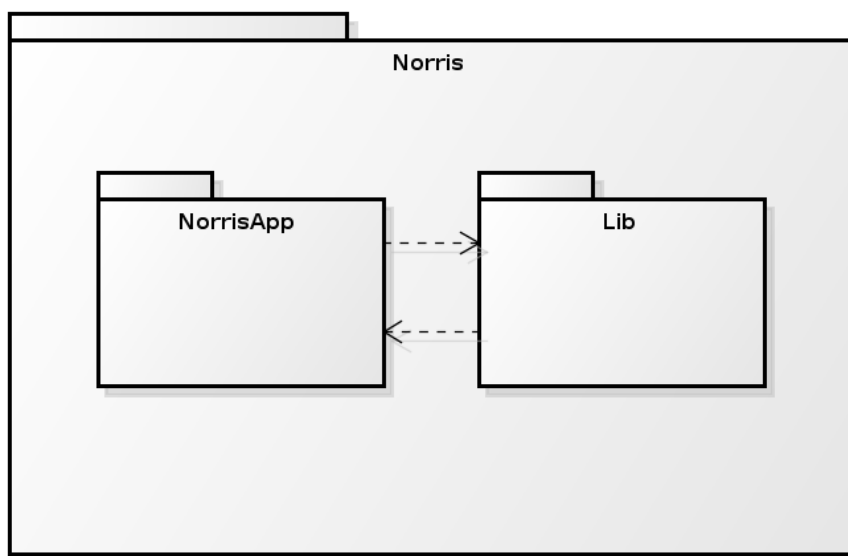


Figura 1: Diagramma dei *package<sub>G</sub>* Norris

L'architettura del sistema è divisa in due *package<sub>G</sub>* principali, a loro volta divisi in sotto *package<sub>G</sub>*:

- **Lib:** si occuperà della gestione logica del *framework<sub>G</sub>*;
- **NorriApp:** si occuperà di fornire un'interfaccia visualizzabile ai *client<sub>G</sub>* 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<sub>G</sub>* gli aggiornamenti ai singoli grafici.

### 3.2.2.1 Lib

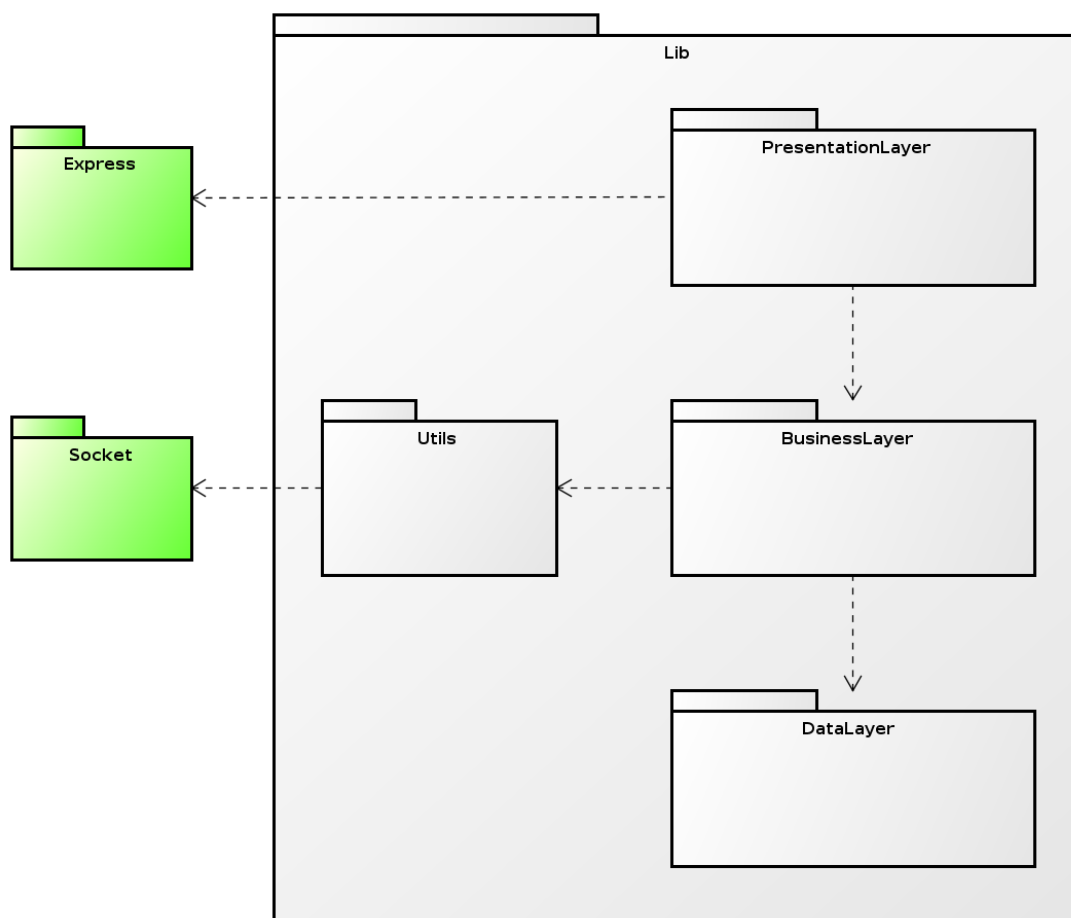


Figura 2: Diagramma del *package<sub>G</sub>* Lib

Il *package<sub>G</sub>* Lib contiene le componenti necessarie al *framework<sub>G</sub>* 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<sub>G</sub>*, quali browser web e applicativo *Android<sub>G</sub>*. È strutturato secondo lo stile architetturale *Three Tier Architecture<sub>G</sub>* 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<sub>G</sub>* **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<sub>G</sub>*, permetterà l'accesso ai loro metodi specifici. Si occuperà inoltre di istanziare autonomamente un *subrouting<sub>G</sub>* nel punto di *mount<sub>G</sub>* designato dallo sviluppatore che gestirà le richieste dai *client<sub>G</sub>* e l'invio degli aggiornamenti.

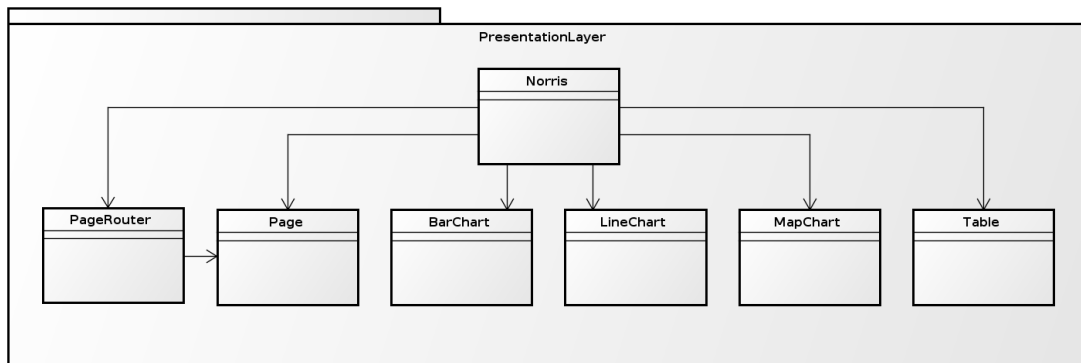


Figura 3: Il *package<sub>G</sub>* Presentation Layer della libreria Norris.

- **Business Layer:** all'interno di questo *package<sub>G</sub>* 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<sub>G</sub>*. 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**

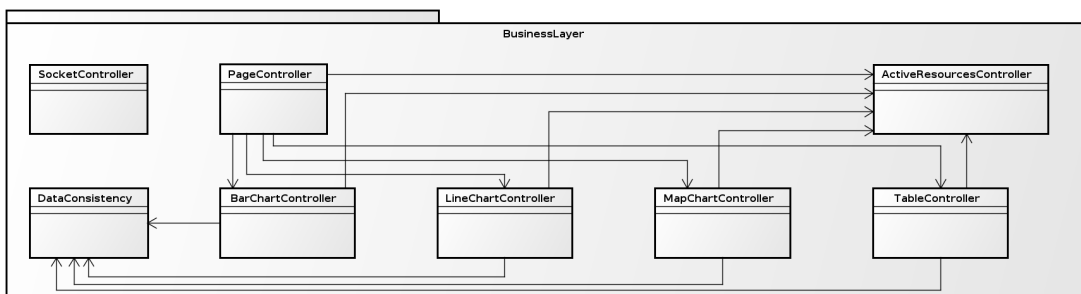


Figura 4: Il *package<sub>G</sub>* Business Layer della libreria Norris.

- **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.

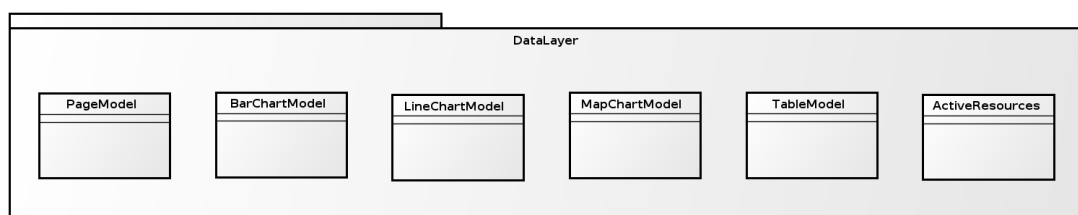


Figura 5: Il *package<sub>G</sub>* Data Layer della libreria Norris.

- Utils:** questo *package<sub>G</sub>* non appartiene allo stile architetturale *Three Tier Architecture<sub>G</sub>* e stato inserito all'interno del *package<sub>G</sub>* **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<sub>G</sub>* 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.

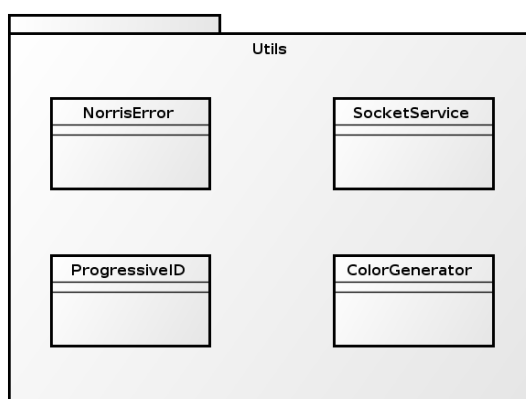


Figura 6: Il *package<sub>G</sub>* Utils della libreria Norris.

### 3.2.2.2 NorrisApp

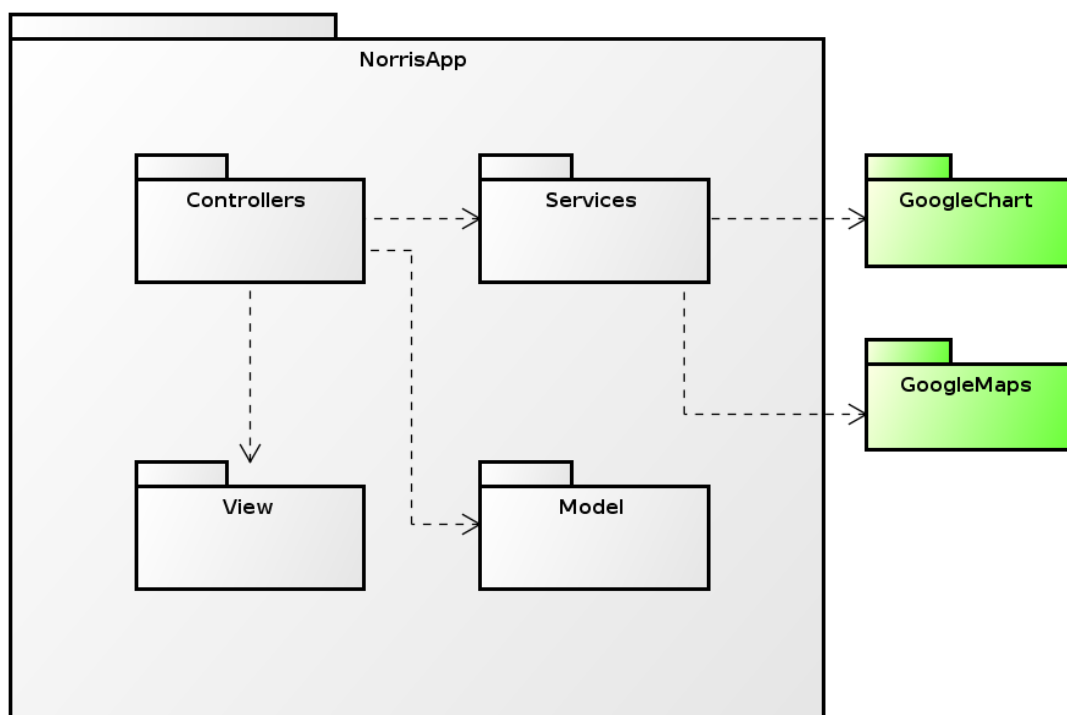


Figura 7: Diagramma dei *package<sub>G</sub>* di NorrisApp

Questa parte dell'architettura si occupa di contenere la sezione *front-end<sub>G</sub>*. 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<sub>G</sub>* principali:

- **View:** *package<sub>G</sub>* comprendente le classi che costituiscono la view relativa al *design pattern<sub>G</sub> MVC<sub>G</sub>* del componente *Front-end<sub>G</sub>*. Ogni view rappresenta un tipo di grafico, che verrà popolato con i dati richiesti. Le componenti di questo *package<sub>G</sub>* non possiedono né campi dati né metodi associati, poichè la logica di business è gestita interamente dalle classi del *package<sub>G</sub>* Controllers, in particolare nella componente scope. Le classi di questo *package<sub>G</sub>* sono implementate tramite file *HTML<sub>G</sub>* e costituiscono i *template<sub>G</sub>* di pagina *AngularJS<sub>G</sub>*.
- **Controller:** *package<sub>G</sub>* comprendente le classi che costituiscono i controller relativi al *design pattern<sub>G</sub> MVC<sub>G</sub>* del componente *Front-end<sub>G</sub>*. Ogni controller gestisce la logica applicativa riguardante un determinato tipo di grafico mediante le funzioni esposte dalle classi del *package<sub>G</sub>* 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<sub>G</sub>* che comprende le classi dei modelli relativi al *design pattern<sub>G</sub> MVC<sub>G</sub>*, dei dati utilizzati dal *front-end<sub>G</sub>*. Servono a fornire al Controller e al Service i dati ricevuti dal *back-end<sub>G</sub>*.

- **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.

### 3.2.3 Applicativo $Android_G$

Questa parte dell'architettura è stata progettata, ma non sarà implementata e non sarà consegnata alla revisione di Accettazione in base alle decisioni prese dal **FlameTech Inc.** come descritto nei verbali del **FlameTech Inc.** .

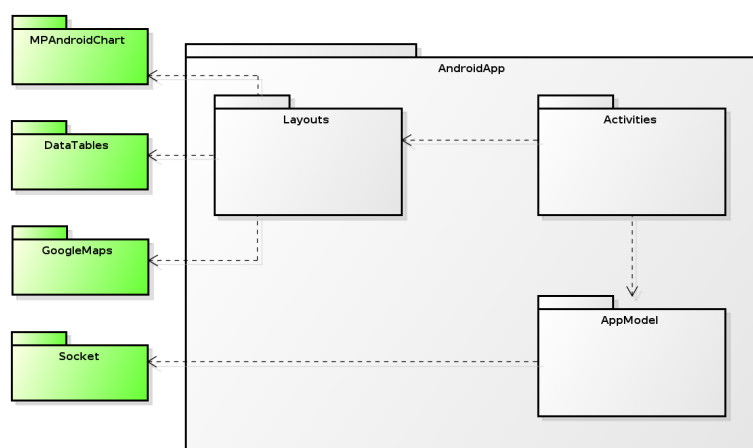


Figura 8: Diagramma dei  $package_G$  applicazione  $Android_G$

Questa parte contiene l'applicazione  $Android_G$  ed è suddivisa in tre  $package_G$  rispecchiando il  $design\ pattern_G\ MVC_G$ . In particolare il  $package_G$  Layouts corrisponde alla view, AppModel corrisponde alla parte model ed il  $package_G$  Activities corrisponde al controller.

- **Layouts:**  $package_G$  comprendente le classi che costituiscono le singole view dell'applicazione. Ogni view rappresenta una sezione come ad esempio: la schermata principale, il menu, le impostazioni.
- **Activities:**  $package_G$  comprendente le classi che gestiscono le operazioni e la logica applicativa. Ogni classe rappresenta una specifica attività che può essere eseguita dall'utilizzatore finale.
- **AppModel:**  $package_G$  che comprende le classi dei modelli dei dati. Servono a fornire alle activities i dati necessari al funzionamento dell'applicazione e i dati ricevuti dal  $back-end_G$  per la costruzione dei grafici.

## 4 Componenti e classi

### 4.1 Componente Norris *framework<sub>G</sub>*

#### 4.1.1 Norris

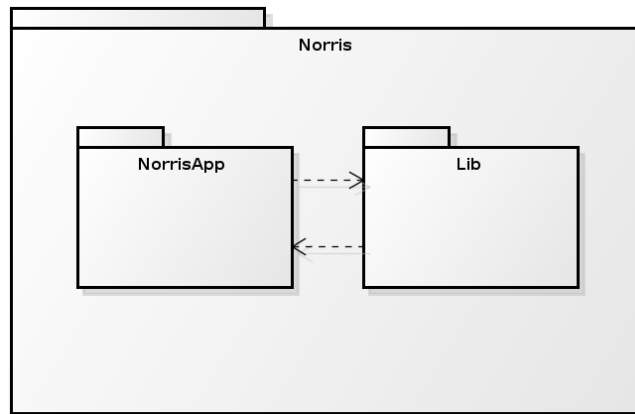


Figura 9: Diagramma del *package<sub>G</sub>* Norris

#### 4.1.1.1 Informazioni sul *package<sub>G</sub>*

##### 4.1.1.1.1 Descrizione

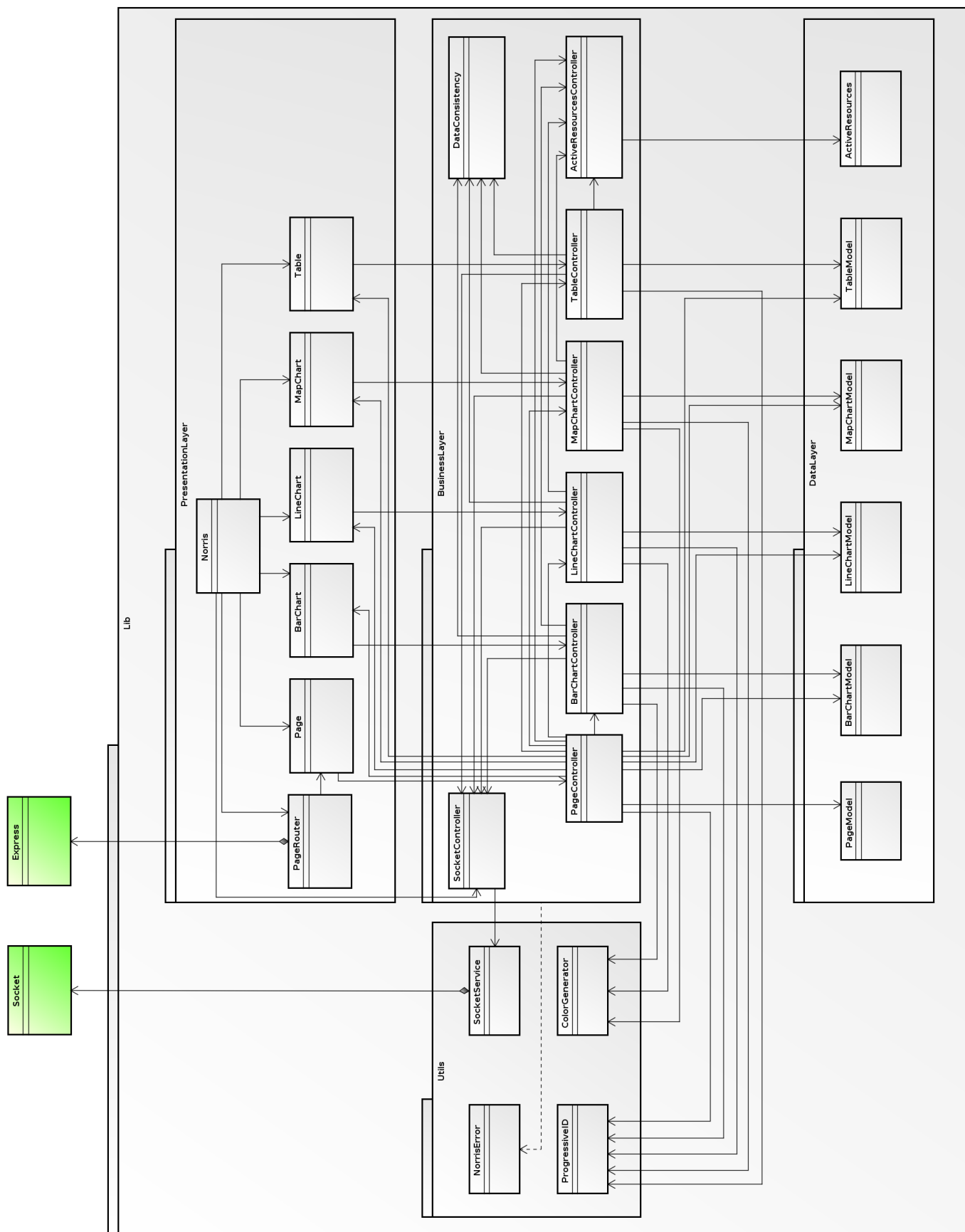
Questo *package<sub>G</sub>* contiene tutte le componenti del *framework<sub>G</sub>* Norris. NorrisApp eseguirà richieste get sulla Lib per ottenere i primi dati della pagina e dei grafici ad essa correlata, successivamente la Lib invierà tramite notifiche *push<sub>G</sub>* gli aggiornamenti ai singoli grafici..

##### 4.1.1.1.2 *Package<sub>G</sub>* contenuti

- Norris::Lib;
- Norris::NorrisApp.



### 4.1.2 Norris::Lib



#### 4.1.2.1 Informazioni sul *package<sub>G</sub>*

##### 4.1.2.1.1 Descrizione

Questo *package<sub>G</sub>* contiene tutte le componenti che permettono allo sviluppatore di creare pagine e grafici, e di aggiornarli.

##### 4.1.2.1.2 *Package<sub>G</sub>* contenuti

- Norris::Lib::BusinessLayer;
- Norris::Lib::DataLayer;
- Norris::Lib::PresentationLayer;
- Norris::Lib::Utils.

#### 4.1.3 Norris::Lib::BusinessLayer

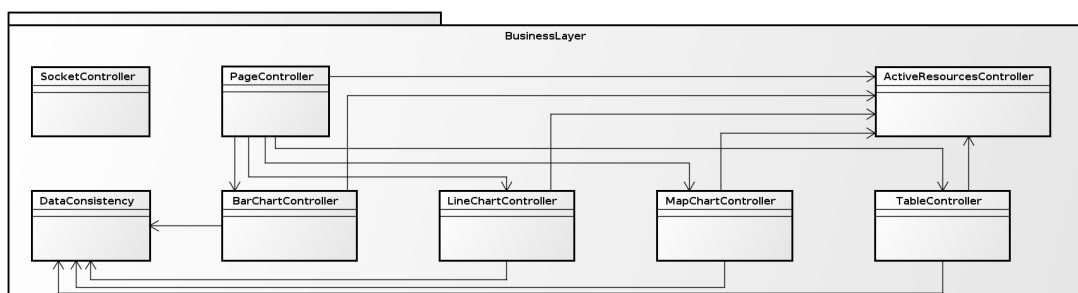


Figura 11: Diagramma delle classi BusinessLayer

#### 4.1.3.1 Informazioni sul *package<sub>G</sub>*

##### 4.1.3.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi che permettono al *framework<sub>G</sub>* di gestire la comunicazione tra il DataLayer e il PresentationLayer.

##### 4.1.3.2 Classi

##### 4.1.3.2.1 Norris::Lib::BusinessLayer::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.

###### Relazioni con altre classi

- Norris::Lib::BusinessLayer::BarChartController

Relazione entrante. La classe `BarChartController` utilizza le funzionalità della classe `ActiveResourcesController`.

– **Norris::Lib::BusinessLayer::LineChartController**

Relazione entrante. La classe `LineChartController` utilizza le funzionalità della classe `ActiveResourcesController`.

– **Norris::Lib::BusinessLayer::MapChartController**

Relazione entrante. La classe `MapChartController` utilizza le funzionalità della classe `ActiveResourcesController`.

– **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe `PageController` utilizza le funzionalità della classe `ActiveResourcesController` per memorizzare il riferimento ad un oggetto di tipo `PageModel` dopo averlo creato.

– **Norris::Lib::BusinessLayer::TableController**

Relazione entrante. La classe `TableController` utilizza le funzionalità della classe `ActiveResourcesController`.

– **Norris::Lib::DataLayer::ActiveResources**

Relazione uscente. La classe `ActiveResourcesController` utilizza questa classe per indicizzare e accedere alle risorse attive.

– **Norris::Lib::Utils::NorrisError**

Relazione uscente. La classe `ActiveResourcesController` utilizza le funzionalità della classe `NorrisError` per la gestione degli errori.

#### 4.1.3.2.2 Norris::Lib::BusinessLayer::BarChartController

##### Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Bar Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da `PresentationLayer::BarChart` per l'utilizzo delle *API<sub>G</sub>* per la creazione e l'aggiornamento dei grafici di tipo *Bar Chart<sub>G</sub>*.

##### Relazioni con altre classi

– **Norris::Lib::PresentationLayer::BarChart**

Relazione entrante. La classe `BarChart` utilizza le funzionalità della classe `BarChartController`.

– **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe `PageController` utilizza le funzionalità della classe `BarChartController` per recuperare le informazioni di un oggetto di tipo `BarChartModel`.

– **Norris::Lib::BusinessLayer::ActiveResourcesController**

Relazione uscente. La classe `BarChartController` utilizza le funzionalità della classe `ActiveResourcesController`.

– **Norris::Lib::DataLayer::BarChartModel**

Relazione uscente. La classe `BarChartController` utilizza questa classe per la creazione di oggetti di tipo `BarChartModel`.

– **Norris::Lib::Utils::ColorManager**

Relazione uscente. La classe `BarChartController` utilizza le funzionalità della classe `ColorGenerator`.

– **Norris::Lib::BusinessLayer::DataConsistency**

Relazione uscente. La classe `BarChartController` utilizza le funzionalità della classe `DataConsistency`.

– **Norris::Lib::Utils::NorrisError**

Relazione uscente. La classe `BarChartController` utilizza le funzionalità della classe `NorrisError` per la gestione degli errori.

– **Norris::Lib::Utils::ProgressiveID**

Relazione uscente. La classe `BarChartController` utilizza le funzionalità della classe `ProgressiveID`.

– **Norris::Lib::BusinessLayer::SocketController**

Relazione uscente. La classe `BarChartController` utilizza le funzionalità della classe `SocketController`.

#### 4.1.3.2.3 **Norris::Lib::BusinessLayer::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<sub>G</sub>* 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<sub>G</sub>* delle opzioni di un tipo di grafico conterrà tutte le proprietà possibili per quel tipo di grafico, mentre un *template<sub>G</sub>* di una proprietà interna conterrà solo le proprietà che interessano quest'ultima.

##### Relazioni con altre classi

– **Norris::Lib::DataLayer::ActiveResources**

Relazione entrante. La classe `DataConsistency` utilizza le funzionalità della classe `NorrisError` per la gestione degli errori.

– **Norris::Lib::BusinessLayer::BarChartController**

Relazione entrante. La classe `BarChartController` utilizza le funzionalità della classe `DataConsistency`.

– **Norris::Lib::BusinessLayer::LineChartController**

Relazione entrante. La classe `LineChartController` utilizza le funzionalità della classe `DataConsistency`.

- **Norris::Lib::BusinessLayer::MapChartController**  
Relazione entrante. La classe MapChartController utilizza le funzionalità della classe DataConsistency.
- **Norris::Lib::BusinessLayer::TableController**  
Relazione entrante. La classe TableController utilizza le funzionalità della classe DataConsistency.
- **Norris::Lib::Utils::NorrisError**  
Relazione uscente. La classe DataConsistency utilizza le funzionalità della classe NorrisError per la gestione degli errori.

#### 4.1.3.2.4 Norris::Lib::BusinessLayer::LineChartController

##### Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Line Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da PresentationLayer::LineChart per l'utilizzo delle *API<sub>G</sub>* per la creazione e l'aggiornamento dei grafici di tipo *Line Chart<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::Lib::PresentationLayer::LineChart**  
Relazione entrante. La classe LineChart utilizza le funzionalità della classe LineChartController.
- **Norris::Lib::BusinessLayer::PageController**  
Relazione entrante. La classe PageController utilizza le funzionalità della classe LineChartController per recuperare le informazioni di un oggetto di tipo LineChartModel.
- **Norris::Lib::BusinessLayer::ActiveResourcesController**  
Relazione uscente. La classe LineChartController utilizza le funzionalità della classe ActiveResourcesController.
- **Norris::Lib::Utils::ColorManager**  
Relazione uscente. La classe LineChartController utilizza le funzionalità della classe ColorGenerator.
- **Norris::Lib::BusinessLayer::DataConsistency**  
Relazione uscente. La classe LineChartController utilizza le funzionalità della classe DataConsistency.
- **Norris::Lib::DataLayer::LineChartModel**  
Relazione uscente. La classe LineChartController utilizza questa classe per la creazione di oggetti di tipo LineChartModel.
- **Norris::Lib::Utils::NorrisError**  
Relazione uscente. La classe LineChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

- **Norris::Lib::Utils::ProgressiveID**

Relazione uscente. La classe LineChartController utilizza le funzionalità della classe ProgressiveID.

- **Norris::Lib::BusinessLayer::SocketController**

Relazione uscente. La classe LineChartController utilizza le funzionalità della classe SocketController.

#### 4.1.3.2.5 Norris::Lib::BusinessLayer::MapChartController

##### Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Map Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da PresentationLayer::MapChart per l'utilizzo delle *API<sub>G</sub>* per la creazione e l'aggiornamento dei grafici di tipo *Map Chart<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::Lib::PresentationLayer::MapChart**

Relazione entrante. La classe MapChart utilizza le funzionalità della classe MapChartController.

- **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe PageController utilizza le funzionalità della classe MapChartController per recuperare le informazioni di un oggetto di tipo MapChartModel.

- **Norris::Lib::BusinessLayer::ActiveResourcesController**

Relazione uscente. La classe MapChartController utilizza le funzionalità della classe ActiveResourcesController.

- **Norris::Lib::Utils::ColorManager**

Relazione uscente. La classe MapChartController utilizza le funzionalità della classe ColorGenerator.

- **Norris::Lib::BusinessLayer::DataConsistency**

Relazione uscente. La classe MapChartController utilizza le funzionalità della classe DataConsistency.

- **Norris::Lib::DataLayer::MapChartModel**

Relazione uscente. La classe MapChartController utilizza questa classe per la creazione di oggetti di tipo MapChartModel.

- **Norris::Lib::Utils::NorrisError**

Relazione uscente. La classe MapChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

- **Norris::Lib::Utils::ProgressiveID**

Relazione uscente. La classe MapChartController utilizza le funzionalità della classe ProgressiveID.

- **Norris::Lib::BusinessLayer::SocketController**

Relazione uscente. La classe MapChartController utilizza le funzionalità della classe SocketController.

#### 4.1.3.2.6 Norris::Lib::BusinessLayer::PageController

##### Descrizione

Questa classe comprende i metodi per la creazione delle pagine.

##### Utilizzo

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

##### Relazioni con altre classi

- **Norris::Lib::PresentationLayer::Page**  
Relazione entrante. La classe Page utilizza le funzionalità della classe PageController.
- **Norris::Lib::BusinessLayer::ActiveResourcesController**  
Relazione uscente. La classe PageController utilizza le funzionalità della classe ActiveResourcesController per memorizzare il riferimento ad un oggetto di tipo PageModel dopo averlo creato.
- **Norris::Lib::PresentationLayer::BarChart**  
Relazione uscente. La classe PageController utilizza le funzionalità della classe BarChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.
- **Norris::Lib::BusinessLayer::BarChartController**  
Relazione uscente. La classe PageController utilizza le funzionalità della classe BarChartController per recuperare le informazioni di un oggetto di tipo BarChartModel.
- **Norris::Lib::DataLayer::BarChartModel**  
Relazione uscente. La classe PageController utilizza le funzionalità della classe BarChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe BarChartController.
- **Norris::Lib::PresentationLayer::LineChart**  
Relazione uscente. La classe PageController utilizza le funzionalità della classe LineChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.
- **Norris::Lib::BusinessLayer::LineChartController**  
Relazione uscente. La classe PageController utilizza le funzionalità della classe LineChartController per recuperare le informazioni di un oggetto di tipo LineChartModel.
- **Norris::Lib::DataLayer::LineChartModel**  
Relazione uscente. La classe PageController utilizza le funzionalità della classe LineChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe LineChartController.
- **Norris::Lib::PresentationLayer::MapChart**

Relazione uscente. La classe PageController utilizza le funzionalità della classe MapChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

– **Norris::Lib::BusinessLayer::MapChartController**

Relazione uscente. La classe PageController utilizza le funzionalità della classe MapChartController per recuperare le informazioni di un oggetto di tipo MapChartModel.

– **Norris::Lib::DataLayer::MapChartModel**

Relazione uscente. La classe PageController utilizza le funzionalità della classe MapChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe MapChartController.

– **Norris::Lib::Utils::NorrisError**

Relazione uscente. La classe PageController utilizza le funzionalità della classe NorrisError per la gestione degli errori .

– **Norris::Lib::DataLayer::PageModel**

Relazione uscente. La classe PageController utilizza questa classe per la creazione di oggetti di tipo PageModel.

– **Norris::Lib::Utils::ProgressiveID**

Relazione uscente. La classe PageController utilizza le funzionalità della classe ProgressiveID per assegnare il codice ID univoco durante la creazione di un oggetto di tipo PageModel.

– **Norris::Lib::PresentationLayer::Table**

Relazione uscente. La classe PageController utilizza le funzionalità della classe Table per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

– **Norris::Lib::BusinessLayer::TableController**

Relazione uscente. La classe PageController utilizza le funzionalità della classe TableController per recuperare le informazioni di un oggetto di tipo TableModel.

– **Norris::Lib::DataLayer::TableModel**

Relazione uscente. La classe PageController utilizza le funzionalità della classe TableModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe TableController.

#### 4.1.3.2.7 Norris::Lib::BusinessLayer::SocketController

##### Descrizione

Questa classe si occupa di gestire le interazioni con la componente *socket.ioG* utilizzata all'interno della libreria Norris nella classe SocketService. .

##### Utilizzo

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



### Relazioni con altre classi

- **Norris::Lib::DataLayer::ActiveResources**  
Relazione entrante. La classe SocketController utilizza le funzionalità della classe NorrisError per la gestione degli errori.
- **Norris::Lib::BusinessLayer::BarChartController**  
Relazione entrante. La classe BarChartController utilizza le funzionalità della classe SocketController.
- **Norris::Lib::BusinessLayer::LineChartController**  
Relazione entrante. La classe LineChartController utilizza le funzionalità della classe SocketController.
- **Norris::Lib::BusinessLayer::MapChartController**  
Relazione entrante. La classe MapChartController utilizza le funzionalità della classe SocketController.
- **Norris::Lib::PresentationLayer::Norris**  
Relazione entrante. La classe Norris utilizza le funzionalità della classe SocketController per configurare il *namespace<sub>G</sub>*.
- **Norris::Lib::PresentationLayer::PageRouter**  
Relazione entrante. La classe PageRouter utilizza le funzionalità della classe SocketController.
- **Norris::Lib::Utils::NorrisError**  
Relazione uscente. La classe SocketController utilizza le funzionalità della classe NorrisError per la gestione degli errori.
- **Norris::Lib::Utils::SocketService**  
Relazione uscente. La classe SocketController utilizza le funzionalità della classe SocketService.

#### 4.1.3.2.8 Norris::Lib::BusinessLayer::TableController

##### Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Table<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da PresentationLayer::Table per l'utilizzo delle *API<sub>G</sub>* per la creazione e l'aggiornamento dei grafici di tipo *Table<sub>G</sub>*.

### Relazioni con altre classi

- **Norris::Lib::BusinessLayer::PageController**  
Relazione entrante. La classe PageController utilizza le funzionalità della classe TableController per recuperare le informazioni di un oggetto di tipo TableModel.
- **Norris::Lib::PresentationLayer::Table**

Relazione entrante. La classe Table utilizza le funzionalità della classe TableController.

– **Norris::Lib::BusinessLayer::ActiveResourcesController**

Relazione uscente. La classe TableController utilizza le funzionalità della classe ActiveResourcesController.

– **Norris::Lib::BusinessLayer::DataConsistency**

Relazione uscente. La classe TableController utilizza le funzionalità della classe DataConsistency.

– **Norris::Lib::Utils::NorrisError**

Relazione uscente. La classe TableController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

– **Norris::Lib::Utils::ProgressiveID**

Relazione uscente. La classe TableController utilizza le funzionalità della classe ProgressiveID.

– **Norris::Lib::DataLayer::TableModel**

Relazione uscente. La classe TableController utilizza questa classe per la creazione di oggetti di tipo TableModel.

#### 4.1.4 Norris::Lib::DataLayer

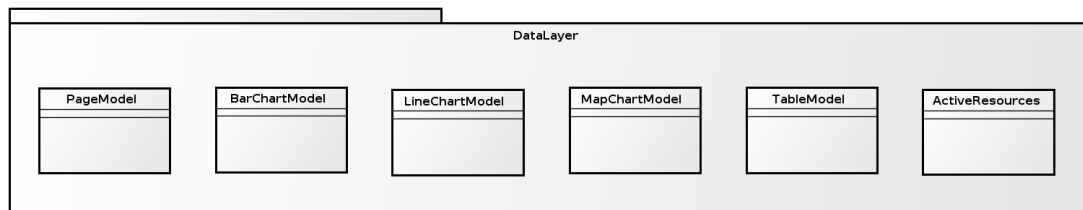


Figura 12: Diagramma delle classi DataLayer

##### 4.1.4.1 Informazioni sul *package<sub>G</sub>*

###### 4.1.4.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi per la gestione del modello dei dati.

###### 4.1.4.2 Classi

###### 4.1.4.2.1 Norris::Lib::DataLayer::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.

###### Relazioni con altre classi

- **Norris::Lib::BusinessLayer::ActiveResourcesController**  
Relazione entrante. La classe `ActiveResourcesController` utilizza questa classe per indicizzare e accedere alle risorse attive.
- **Norris::Lib::BusinessLayer::DataConsistency**  
Relazione uscente. La classe `DataConsistency` utilizza le funzionalità della classe `NorrisError` per la gestione degli errori.
- **Norris::Lib::BusinessLayer::SocketController**  
Relazione uscente. La classe `SocketController` utilizza le funzionalità della classe `NorrisError` per la gestione degli errori.

#### 4.1.4.2.2 Norris::Lib::DataLayer::BarChartModel

##### Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Bar Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da `BusinessLayer::BarChartController` per la creazione di grafici di tipo *Bar Chart<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::Lib::BusinessLayer::BarChartController**  
Relazione entrante. La classe `BarChartController` utilizza questa classe per la creazione di oggetti di tipo `BarChartModel`.
- **Norris::Lib::BusinessLayer::PageController**  
Relazione entrante. La classe `PageController` utilizza le funzionalità della classe `BarChartModel` per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe `BarChartController`.

#### 4.1.4.2.3 Norris::Lib::DataLayer::LineChartModel

##### Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Line Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da `BusinessLayer::LineChartController` per la creazione e la modifica dei grafici di tipo *Line Chart<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::Lib::BusinessLayer::LineChartController**  
Relazione entrante. La classe `LineChartController` utilizza questa classe per la creazione di oggetti di tipo `LineChartModel`.
- **Norris::Lib::BusinessLayer::PageController**  
Relazione entrante. La classe `PageController` utilizza le funzionalità della classe `LineChartModel` per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe `LineChartController`.

#### 4.1.4.2.4 Norris::Lib::DataLayer::MapChartModel

##### Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Map Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da *BusinessLayer::MapChartController* per la creazione e la modifica dei grafici di tipo *Map Chart<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::Lib::BusinessLayer::MapChartController**

Relazione entrante. La classe *MapChartController* utilizza questa classe per la creazione di oggetti di tipo *MapChartModel*.

- **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe *PageController* utilizza le funzionalità della classe *MapChartModel* per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe *MapChartController*.

#### 4.1.4.2.5 Norris::Lib::DataLayer::PageModel

##### Descrizione

Questa classe rappresenta il modello dei dati di una pagina.

##### Utilizzo

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

##### Relazioni con altre classi

- **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe *PageController* utilizza questa classe per la creazione di oggetti di tipo *PageModel*.

#### 4.1.4.2.6 Norris::Lib::DataLayer::TableModel

##### Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Table<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da *BusinessLayer::TableController* per la creazione e la modifica dei grafici di tipo *Table<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe PageController utilizza le funzionalità della classe TableModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe TableController.

– **Norris::Lib::BusinessLayer::TableController**

Relazione entrante. La classe TableContoller utilizza questa classe per la creazione di oggetti di tipo TableModel.

#### 4.1.5 Norris::Lib::PresentationLayer

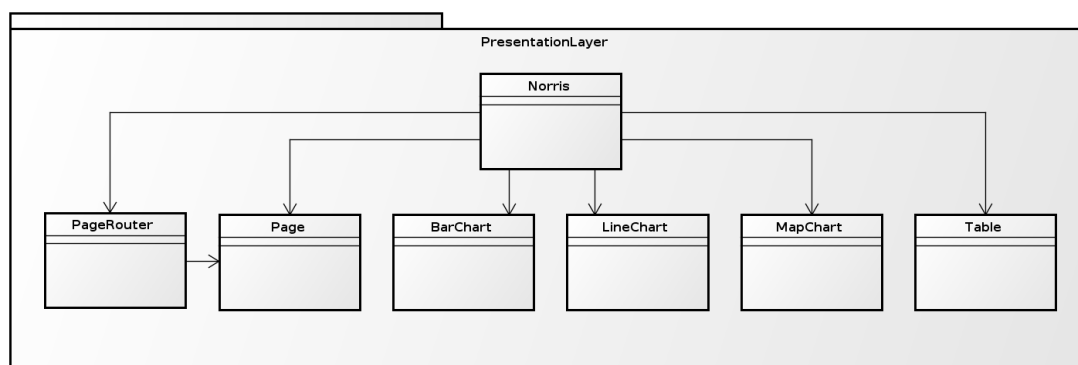


Figura 13: Diagramma delle classi PresentationLayer

##### 4.1.5.1 Informazioni sul *package<sub>G</sub>*

###### 4.1.5.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi che permettono la comunicazione con l'esterno.

##### 4.1.5.2 Classi

###### 4.1.5.2.1 Norris::Lib::PresentationLayer::BarChart

###### Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Bar Chart<sub>G</sub>*.

###### Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Bar Chart<sub>G</sub>*.

###### Relazioni con altre classi

– **Norris::Lib::PresentationLayer::Norris**

Relazione entrante. La classe NorrisIndex importa le funzioni di BarChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto BarChart.

– **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe PageController utilizza le funzionalità della classe BarChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

– **Norris::Lib::BusinessLayer::BarChartController**

Relazione uscente. La classe BarChart utilizza le funzionalità della classe BarChartController.

#### 4.1.5.2.2 Norris::Lib::PresentationLayer::LineChart

##### Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Line Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Line Chart<sub>G</sub>*.

##### Relazioni con altre classi

– **Norris::Lib::PresentationLayer::Norris**

Relazione entrante. La classe NorrisIndex importa le funzioni di LineChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto LineChart.

– **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe PageController utilizza le funzionalità della classe LineChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

– **Norris::Lib::BusinessLayer::LineChartController**

Relazione uscente. La classe LineChart utilizza le funzionalità della classe LineChartController.

#### 4.1.5.2.3 Norris::Lib::PresentationLayer::MapChart

##### Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Map Chart<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Map Chart<sub>G</sub>*.

##### Relazioni con altre classi

– **Norris::Lib::PresentationLayer::Norris**

Relazione entrante. La classe NorrisIndex importa le funzioni di MapChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto MapChart.

– **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe PageController utilizza le funzionalità della classe MapChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

– **Norris::Lib::BusinessLayer::MapChartController**

Relazione uscente. La classe MapChart utilizza le funzionalità della classe MapChartController.

#### 4.1.5.2.4 Norris::Lib::PresentationLayer::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<sub>G</sub>* sul punto di *mount<sub>G</sub>* designato dallo sviluppatore..

##### Relazioni con altre classi

– **Norris::Lib::PresentationLayer::BarChart**

Relazione uscente. La classe NorrisIndex importa le funzioni di BarChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto BarChart.

– **Norris::Lib::PresentationLayer::LineChart**

Relazione uscente. La classe NorrisIndex importa le funzioni di LineChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto LineChart.

– **Norris::Lib::PresentationLayer::MapChart**

Relazione uscente. La classe NorrisIndex importa le funzioni di MapChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto MapChart.

– **Norris::Lib::PresentationLayer::Page**

Relazione uscente. La classe NorrisIndex importa le funzioni di Page per dare accesso allo sviluppatore alle funzionalità dell'oggetto Page.

– **Norris::Lib::PresentationLayer::PageRouter**

Relazione uscente. La classe NorrisIndex utilizza le funzioni di PageRouter per la creazione di un *subrouting<sub>G</sub>* interno a Norris.

– **Norris::Lib::BusinessLayer::SocketController**

Relazione uscente. La classe Norris utilizza le funzionalità della classe SocketController per configurare il *namespace<sub>G</sub>*.

– **Norris::Lib::PresentationLayer::Table**

Relazione uscente. La classe NorrisIndex importa le funzioni di Table per dare accesso allo sviluppatore alle funzionalità dell'oggetto Table.

#### 4.1.5.2.5 Norris::Lib::PresentationLayer::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.

##### Relazioni con altre classi

- **Norris::Lib::PresentationLayer::Norris**  
Relazione entrante. La classe NorrisIndex importa le funzioni di Page per dare accesso allo sviluppatore alle funzionalità dell'oggetto Page.
- **Norris::Lib::BusinessLayer::PageController**  
Relazione uscente. La classe Page utilizza le funzionalità della classe PageController.

#### 4.1.5.2.6 Norris::Lib::PresentationLayer::PageRouter

##### Descrizione

Questa classe gestisce la creazione e l'utilizzo del *subrouting<sub>G</sub>*.

##### Utilizzo

La classe verrà utilizzata per la creazione di un *subrouting<sub>G</sub>*, sul punto di *mount<sub>G</sub>* designato dallo sviluppatore, che si occuperà di gestire le richieste GET dei *client<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::Lib::PresentationLayer::Norris**  
Relazione entrante. La classe NorrisIndex utilizza le funzioni di PageRouter per la creazione di un *subrouting<sub>G</sub>* interno a Norris.
- **Norris::Lib::BusinessLayer::SocketController**  
Relazione uscente. La classe PageRouter utilizza le funzionalità della classe SocketController.

#### 4.1.5.2.7 Norris::Lib::PresentationLayer::Table

##### Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Table<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Table<sub>G</sub>*.

##### Relazioni con altre classi



– **Norris::Lib::PresentationLayer::Norris**

Relazione entrante. La classe NorrisIndex importa le funzioni di Table per dare accesso allo sviluppatore alle funzionalità dell'oggetto Table.

– **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe PageController utilizza le funzionalità della classe Table per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

– **Norris::Lib::BusinessLayer::TableController**

Relazione uscente. La classe Table utilizza le funzionalità della classe TableController.

#### 4.1.6 Norris::Lib::Utils

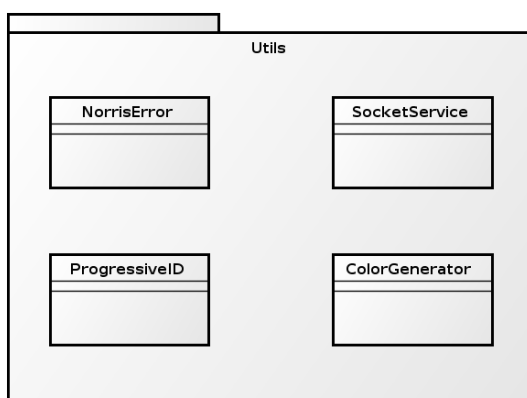


Figura 14: Diagramma delle classi Utils

##### 4.1.6.1 Informazioni sul *package<sub>G</sub>*

###### 4.1.6.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi di supporto alle classi del *package<sub>G</sub>* BusinessLayer.

##### 4.1.6.2 Classi

###### 4.1.6.2.1 Norris::Lib::Utils::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.

###### Relazioni con altre classi

– **Norris::Lib::BusinessLayer::BarChartController**

Relazione entrante. La classe BarChartController utilizza le funzionalità della classe ColorGenerator.

– **Norris::Lib::BusinessLayer::LineChartController**

Relazione entrante. La classe LineChartController utilizza le funzionalità della classe ColorGenerator.

– **Norris::Lib::BusinessLayer::MapChartController**

Relazione entrante. La classe MapChartController utilizza le funzionalità della classe ColorGenerator.

#### 4.1.6.2.2 Norris::Lib::Utils::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.

##### Relazioni con altre classi

– **Norris::Lib::BusinessLayer::ActiveResourcesController**

Relazione entrante. La classe ActiveResourcesController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

– **Norris::Lib::BusinessLayer::BarChartController**

Relazione entrante. La classe BarChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

– **Norris::Lib::BusinessLayer::DataConsistency**

Relazione entrante. La classe DataConsistency utilizza le funzionalità della classe NorrisError per la gestione degli errori.

– **Norris::Lib::BusinessLayer::LineChartController**

Relazione entrante. La classe LineChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

– **Norris::Lib::BusinessLayer::MapChartController**

Relazione entrante. La classe MapChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

– **Norris::Lib::BusinessLayer::PageController**

Relazione entrante. La classe PageController utilizza le funzionalità della classe NorrisError per la gestione degli errori .

– **Norris::Lib::BusinessLayer::SocketController**

Relazione entrante. La classe SocketController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

– **Norris::Lib::BusinessLayer::TableController**

Relazione entrante. La classe TableController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

#### 4.1.6.2.3 Norris::Lib::Utils::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.

##### Relazioni con altre classi

###### – Norris::Lib::BusinessLayer::BarChartController

Relazione entrante. La classe BarChartController utilizza le funzionalità della classe ProgressiveID.

###### – Norris::Lib::BusinessLayer::LineChartController

Relazione entrante. La classe LineChartController utilizza le funzionalità della classe ProgressiveID.

###### – Norris::Lib::BusinessLayer::MapChartController

Relazione entrante. La classe MapChartController utilizza le funzionalità della classe ProgressiveID.

###### – Norris::Lib::BusinessLayer::PageController

Relazione entrante. La classe PageController utilizza le funzionalità della classe ProgressiveID per assegnare il codice ID univoco durante la creazione di un oggetto di tipo PageModel.

###### – Norris::Lib::BusinessLayer::TableController

Relazione entrante. La classe TableController utilizza le funzionalità della classe ProgressiveID.

#### 4.1.6.2.4 Norris::Lib::Utils::SocketService

##### Descrizione

Questa classe contiene i metodi riguardanti *Socket.io* che verranno utilizzati dalla libreria Norris..

##### Utilizzo

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

##### Relazioni con altre classi

###### – Norris::Lib::BusinessLayer::SocketController

Relazione entrante. La classe SocketController utilizza le funzionalità della classe SocketService.

#### 4.1.7 Norris::NorrisApp

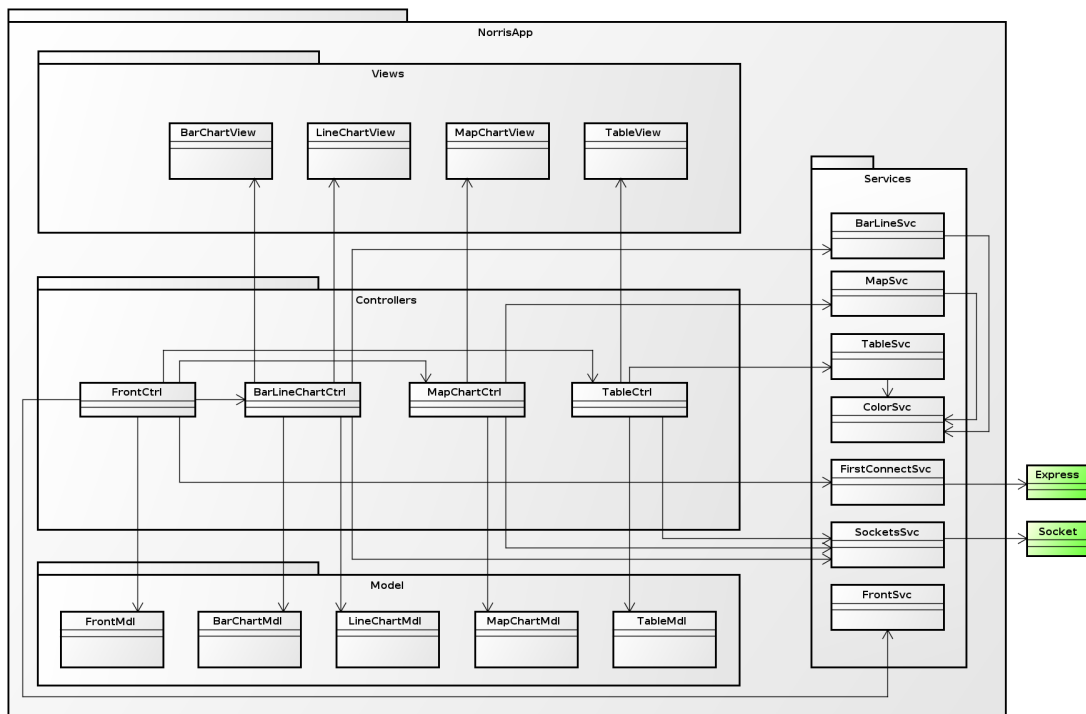


Figura 15: Diagramma delle classi NorrisApp

##### 4.1.7.1 Informazioni sul *package<sub>G</sub>*

###### 4.1.7.1.1 Descrizione

Questo *package<sub>G</sub>* contiene tutte le componenti che permettono alla libreria Norris di visualizzare ed aggiornare pagine e grafici via browser..

###### 4.1.7.1.2 *Package<sub>G</sub>* contenuti

- Norris::NorrisApp::Controllers;
- Norris::NorrisApp::Model;
- Norris::NorrisApp::Services;
- Norris::NorrisApp::Views.

#### 4.1.8 Norris::NorrisApp::Controllers

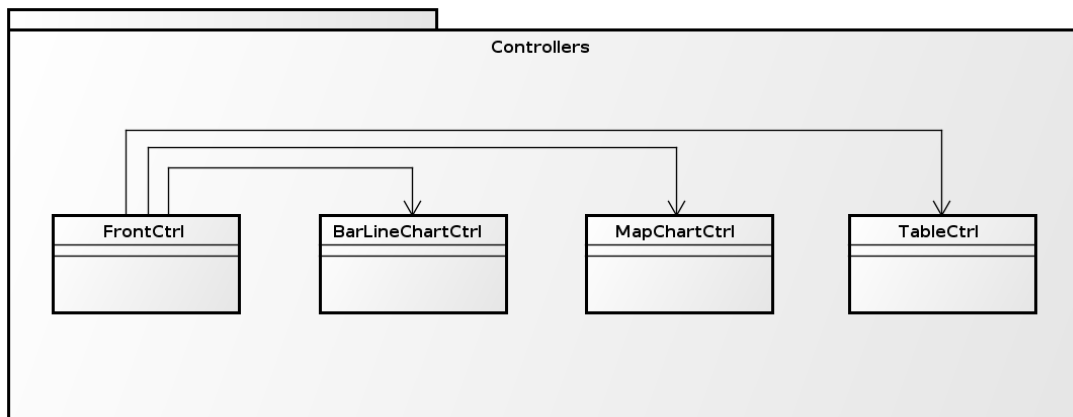


Figura 16: Diagramma delle classi Controllers

##### 4.1.8.1 Informazioni sul *package<sub>G</sub>*

###### 4.1.8.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi che permettono al *framework<sub>G</sub>* di gestire la comunicazione tra Model e Views.

##### 4.1.8.2 Classi

###### 4.1.8.2.1 Norris::NorrisApp::Controllers::BarLineChartCtrl

###### Descrizione

Questa classe contiene i metodi necessari per gestire i grafici di tipo *Bar Chart<sub>G</sub>* e *Line Chart<sub>G</sub>* lato *front-end<sub>G</sub>*.

###### Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end<sub>G</sub>* che riguardano i grafici di tipo *Bar Chart<sub>G</sub>* e *Line Chart<sub>G</sub>*.

###### Relazioni con altre classi

###### – Norris::NorrisApp::Controllers::FrontCtrl

Relazione entrante. La classe *FrontCtrl* utilizzerà le funzionalità della classe *BarLineChartCtrl*.

###### – Norris::NorrisApp::Model::BarChartMdl

Relazione uscente. La classe *BarLineChartCtrl* utilizzerà le funzionalità della classe *BarChartMdl*.

###### – Norris::NorrisApp::Views::BarChartView

Relazione uscente. La classe *BarLineChartCtrl* utilizzerà le funzionalità della classe *BarChartView*.

###### – Norris::NorrisApp::Services::BarLineSvc

Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarLineSvc.

– **Norris::NorrisApp::Model::LineChartMdl**

Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe LineChartMdl.

– **Norris::NorrisApp::Views::LineChartView**

Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe LineChartView.

– **Norris::NorrisApp::Services::SocketsSvc**

Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe SocketsSvc.

#### 4.1.8.2.2 Norris::NorrisApp::Controllers::FrontCtrl

##### Descrizione

Questa classe si occupa di gestire il *front-end<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata allo scopo di eseguire le prime chiamate necessarie al *front-end<sub>G</sub>*.

##### Relazioni con altre classi

– **Norris::NorrisApp::Controllers::BarLineChartCtrl**

Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe BarLineChartCtrl.

– **Norris::NorrisApp::Services::FirstConnectSvc**

Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe FirstConnectSvc.

– **Norris::NorrisApp::Model::FrontMdl**

Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe FrontMdl.

– **Norris::NorrisApp::Services::FrontSvc**

Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe FrontSvc.

– **Norris::NorrisApp::Controllers::MapChartCtrl**

Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe MapChartCtrl.

– **Norris::NorrisApp::Controllers::TableCtrl**

Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe TableCtrl.

#### 4.1.8.2.3 Norris::NorrisApp::Controllers::MapChartCtrl

##### Descrizione

Questa classe contiene i metodi necessari per gestire il grafico di tipo *Map Chart<sub>G</sub>* lato *front-end<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end<sub>G</sub>* che riguardano il grafico di tipo *Map Chart<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::NorrisApp::Controllers::FrontCtrl**  
Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe MapChartCtrl.
- **Norris::NorrisApp::Model::MapChartMdl**  
Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartMdl.
- **Norris::NorrisApp::Views::MapChartView**  
Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartView.
- **Norris::NorrisApp::Services::MapSvc**  
Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe MapSvc.
- **Norris::NorrisApp::Services::SocketsSvc**  
Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe SocketSvc.

#### 4.1.8.2.4 Norris::NorrisApp::Controllers::TableCtrl

##### Descrizione

Questa classe contiene i metodi necessari per gestire il grafico di tipo *Table<sub>G</sub>* lato *front-end<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end<sub>G</sub>* che riguardano il grafico di tipo *Table<sub>G</sub>*.

##### Relazioni con altre classi

- **Norris::NorrisApp::Controllers::FrontCtrl**  
Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe TableCtrl.
- **Norris::NorrisApp::Services::SocketsSvc**  
Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe SocketsSvc.

– **Norris::NorrisApp::Model::TableMdl**

Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe TableMdl.

– **Norris::NorrisApp::Services::TableSvc**

Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe TableSvc.

– **Norris::NorrisApp::Views::TableView**

Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe TableView.

#### 4.1.9 Norris::NorrisApp::Model

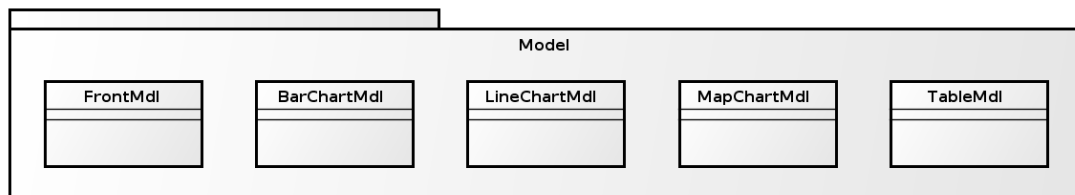


Figura 17: Diagramma delle classi Model

##### 4.1.9.1 Informazioni sul *package<sub>G</sub>*

###### 4.1.9.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi per la gestione del modello dei dati .

##### 4.1.9.2 Classi

###### 4.1.9.2.1 Norris::NorrisApp::Model::BarChartMdl

###### Descrizione

Questa classe conterrà i valori relativi al grafico *Bar Chart<sub>G</sub>* dal lato *front-end<sub>G</sub>*.

###### Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS<sub>G</sub>* prevede l'inserimento dei valori all'interno delle classi del *package<sub>G</sub>* Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end<sub>G</sub>* di Norris.

###### Relazioni con altre classi

– **Norris::NorrisApp::Controllers::BarLineChartCtrl**

Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarChartMdl.



#### 4.1.9.2.2 Norris::NorrisApp::Model::FrontMdl

##### Descrizione

Questa classe conterrà i valori relativi ai grafici di una pagina dal lato *front-end<sub>G</sub>*.

##### Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS<sub>G</sub>* prevede l'inserimento dei valori all'interno delle classi del *package<sub>G</sub>* *Controllers*. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end<sub>G</sub>* di Norris.

##### Relazioni con altre classi

###### – Norris::NorrisApp::Controllers::FrontCtrl

Relazione entrante. La classe *FrontCtrl* utilizzerà le funzionalità della classe *FrontMdl*.

#### 4.1.9.2.3 Norris::NorrisApp::Model::LineChartMdl

##### Descrizione

Questa classe conterrà i valori relativi al grafico *Line Chart<sub>G</sub>* dal lato *front-end<sub>G</sub>*.

##### Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS<sub>G</sub>* prevede l'inserimento dei valori all'interno delle classi del *package<sub>G</sub>* *Controllers*. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end<sub>G</sub>* di Norris.

##### Relazioni con altre classi

###### – Norris::NorrisApp::Controllers::BarLineChartCtrl

Relazione entrante. La classe *BarLineChartCtrl* utilizzerà le funzionalità della classe *LineChartMdl*.

#### 4.1.9.2.4 Norris::NorrisApp::Model::MapChartMdl

##### Descrizione

Questa classe conterrà i valori relativi al grafico *Map Chart<sub>G</sub>* dal lato *front-end<sub>G</sub>*.

##### Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS<sub>G</sub>* prevede l'inserimento dei valori all'interno delle classi del *package<sub>G</sub>* *Controllers*. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end<sub>G</sub>* di Norris.

##### Relazioni con altre classi

– Norris::NorrisApp::Controllers::MapChartCtrl

Relazione entrante. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartMdl.

#### 4.1.9.2.5 Norris::NorrisApp::Model::TableMdl

##### Descrizione

Questa classe conterrà i valori relativi al grafico *TableG* dal lato *front-endG*.

##### Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS<sub>G</sub>* prevede l'inserimento dei valori all'interno delle classi del *package<sub>G</sub>* Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end<sub>G</sub>* di Norris.

##### Relazioni con altre classi

– Norris::NorrisApp::Controllers::TableCtrl

Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe TableMdl.

#### 4.1.10 Norris::NorrisApp::Services

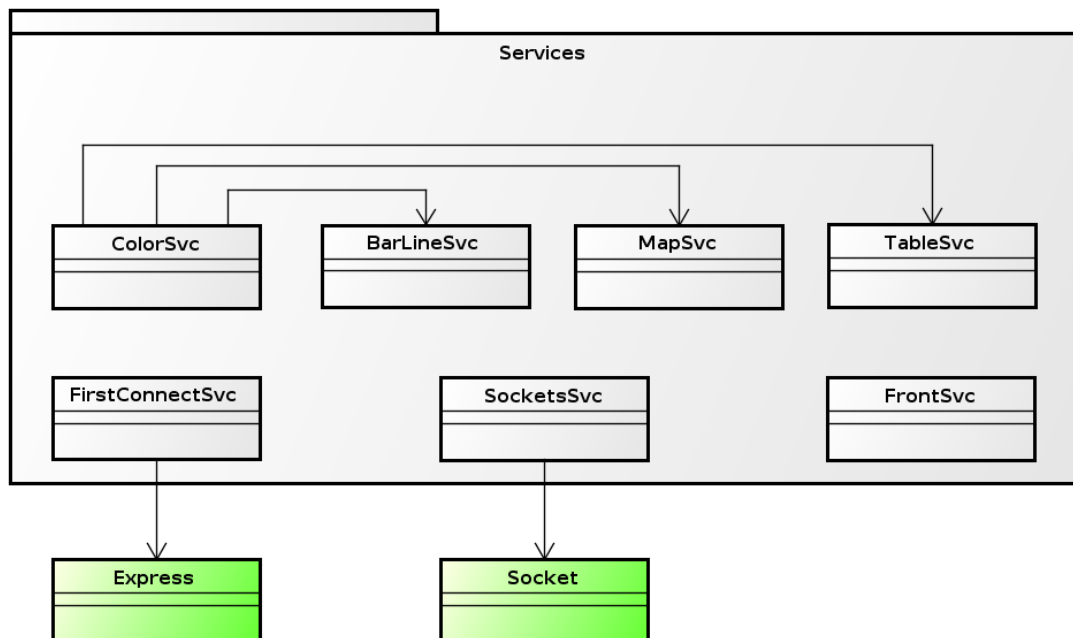


Figura 18: Diagramma delle classi Services

#### 4.1.10.1 Informazioni sul *package<sub>G</sub>*

##### 4.1.10.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi di supporto alle classi del *package<sub>G</sub>* Controllers

#### 4.1.10.2 Classi

##### 4.1.10.2.1 Norris::NorrisApp::Services::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.

###### Relazioni con altre classi

- **Norris::NorrisApp::Controllers::BarLineChartCtrl**  
Relazione entrante. La classe `BarLineChartCtrl` utilizzerà le funzionalità della classe `BarLineSvc`.
- **Norris::NorrisApp::Services::ColorsSvc**  
Relazione uscente. La classe `BarLineSvc` utilizza le funzionalità della classe `ColorError` per la gestione degli errori.

##### 4.1.10.2.2 Norris::NorrisApp::Services::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<sub>G</sub>* a quello conforme alle librerie utilizzate lato *front-end<sub>G</sub>*.

###### Relazioni con altre classi

- **Norris::NorrisApp::Services::BarLineSvc**  
Relazione entrante. La classe `BarLineSvc` utilizza le funzionalità della classe `ColorError` per la gestione degli errori.
- **Norris::NorrisApp::Services::MapSvc**  
Relazione entrante. La classe `MapSvc` utilizza le funzionalità della classe `ColorError` per la gestione degli errori.
- **Norris::NorrisApp::Services::TableSvc**  
Relazione entrante. La classe `TableSvc` utilizza le funzionalità della classe `ColorsSvc` per la gestione delle funzionalità relative ai colori.

#### 4.1.10.2.3 Norris::NorrisApp::Services::FirstConnectSvc

##### Descrizione

Questa classe contiene il metodo necessario al recupero dell' *URL<sub>G</sub>* del *server<sub>G</sub>*.

##### Utilizzo

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

##### Relazioni con altre classi

###### – Norris::NorrisApp::Controllers::FrontCtrl

Relazione entrante. La classe *FrontCtrl* utilizzerà le funzionalità della classe *FirstConnectSvc*.

#### 4.1.10.2.4 Norris::NorrisApp::Services::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.

##### Relazioni con altre classi

###### – Norris::NorrisApp::Controllers::FrontCtrl

Relazione entrante. La classe *FrontCtrl* utilizzerà le funzionalità della classe *FrontSvc*.

#### 4.1.10.2.5 Norris::NorrisApp::Services::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.

##### Relazioni con altre classi

###### – Norris::NorrisApp::Controllers::MapChartCtrl

Relazione entrante. La classe *MapChartCtrl* utilizzerà le funzionalità della classe *MapSvc*.

###### – Norris::NorrisApp::Services::ColorsSvc

Relazione uscente. La classe *MapSvc* utilizza le funzionalità della classe *ColorError* per la gestione degli errori.

#### 4.1.10.2.6 Norris::NorrisApp::Services::SocketsSvc

##### Descrizione

Questa classe contiene i metodi necessari alla gestione dei servizi di *Socket.io*<sub>G</sub> dal lato del *front-end*<sub>G</sub>.

##### Utilizzo

Sarà utilizzata allo scopo di gestire le connessioni con il *socket*<sub>G</sub> lato *server*<sub>G</sub> dalle classi del *package*<sub>G</sub> Controllers.

##### Relazioni con altre classi

- **Norris::NorrisApp::Controllers::BarLineChartCtrl**  
Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe SocketsSvc.
- **Norris::NorrisApp::Controllers::MapChartCtrl**  
Relazione entrante. La classe MapChartCtrl utilizzerà le funzionalità della classe SocketsSvc.
- **Norris::NorrisApp::Controllers::TableCtrl**  
Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe SocketsSvc.

#### 4.1.10.2.7 Norris::NorrisApp::Services::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.

##### Relazioni con altre classi

- **Norris::NorrisApp::Controllers::TableCtrl**  
Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe TableSvc.
- **Norris::NorrisApp::Services::ColorsSvc**  
Relazione uscente. La classe TableSvc utilizza le funzionalità della classe ColorsSvc per la gestione delle funzionalità relative ai colori.

#### 4.1.11 Norris::NorrisApp::Views

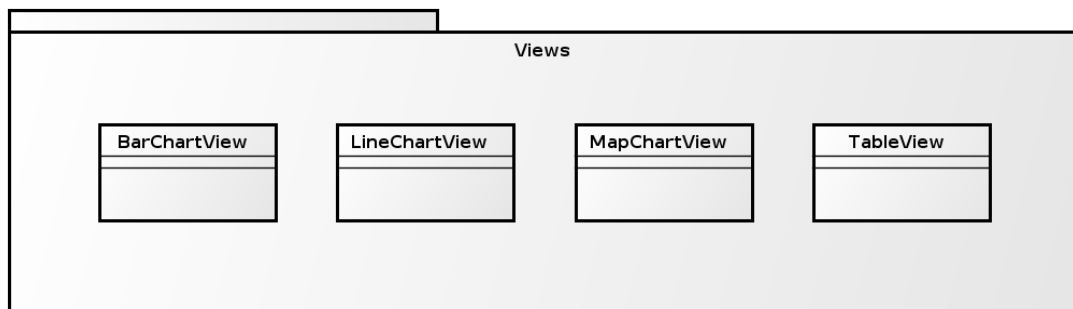


Figura 19: Diagramma delle classi Views

##### 4.1.11.1 Informazioni sul *package<sub>G</sub>*

###### 4.1.11.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi necessarie all’inserimento finale dei grafici nella pagina che verrà visualizzata dall’utente finale tramite browser..

##### 4.1.11.2 Classi

###### 4.1.11.2.1 Norris::NorrisApp::Views::BarChartView

###### Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Bar Chart<sub>G</sub>* nel *front-end<sub>G</sub>*.

###### Utilizzo

Sarà utilizzata da *AngularJS<sub>G</sub>* per visualizzare un grafico di tipo *Bar Chart<sub>G</sub>* all’interno della pagina finale.

###### Relazioni con altre classi

###### – Norris::NorrisApp::Controllers::BarLineChartCtrl

Relazione entrante. La classe *BarLineChartCtrl* utilizzerà le funzionalità della classe *BarChartView*.

###### 4.1.11.2.2 Norris::NorrisApp::Views::Index

###### Descrizione

Questa classe importa tutti gli script e i file necessari al *front-end<sub>G</sub>*. .

###### Utilizzo

Sarà inviata ai browser in seguito alla loro prima richiesta e si occuperà di importare gli script necessari al funzionamento del *front-end<sub>G</sub>*..

#### 4.1.11.2.3 Norris::NorrisApp::Views::LineChartView

##### Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Line Chart<sub>G</sub>* nel *front-end<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da *AngularJS<sub>G</sub>* per visualizzare un grafico di tipo *Line Chart<sub>G</sub>* all'interno della pagina finale.

##### Relazioni con altre classi

- Norris::NorrisApp::Controllers::BarLineChartCtrl

Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe LineChartView.

#### 4.1.11.2.4 Norris::NorrisApp::Views::MapChartView

##### Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Map Chart<sub>G</sub>* dal lato *front-end<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da *AngularJS<sub>G</sub>* visualizzare un grafico di tipo *Map Chart<sub>G</sub>* all'interno della pagina finale.

##### Relazioni con altre classi

- Norris::NorrisApp::Controllers::MapChartCtrl

Relazione entrante. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartView.

#### 4.1.11.2.5 Norris::NorrisApp::Views::TableView

##### Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Table<sub>G</sub>* dal lato *front-end<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da *AngularJS<sub>G</sub>* per visualizzare un grafico di tipo *Table<sub>G</sub>* all'interno della pagina finale.

##### Relazioni con altre classi

- Norris::NorrisApp::Controllers::TableCtrl

Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe TableView.

## 4.2 Componente applicazione *Android<sub>G</sub>*

### 4.2.1 AndroidApp

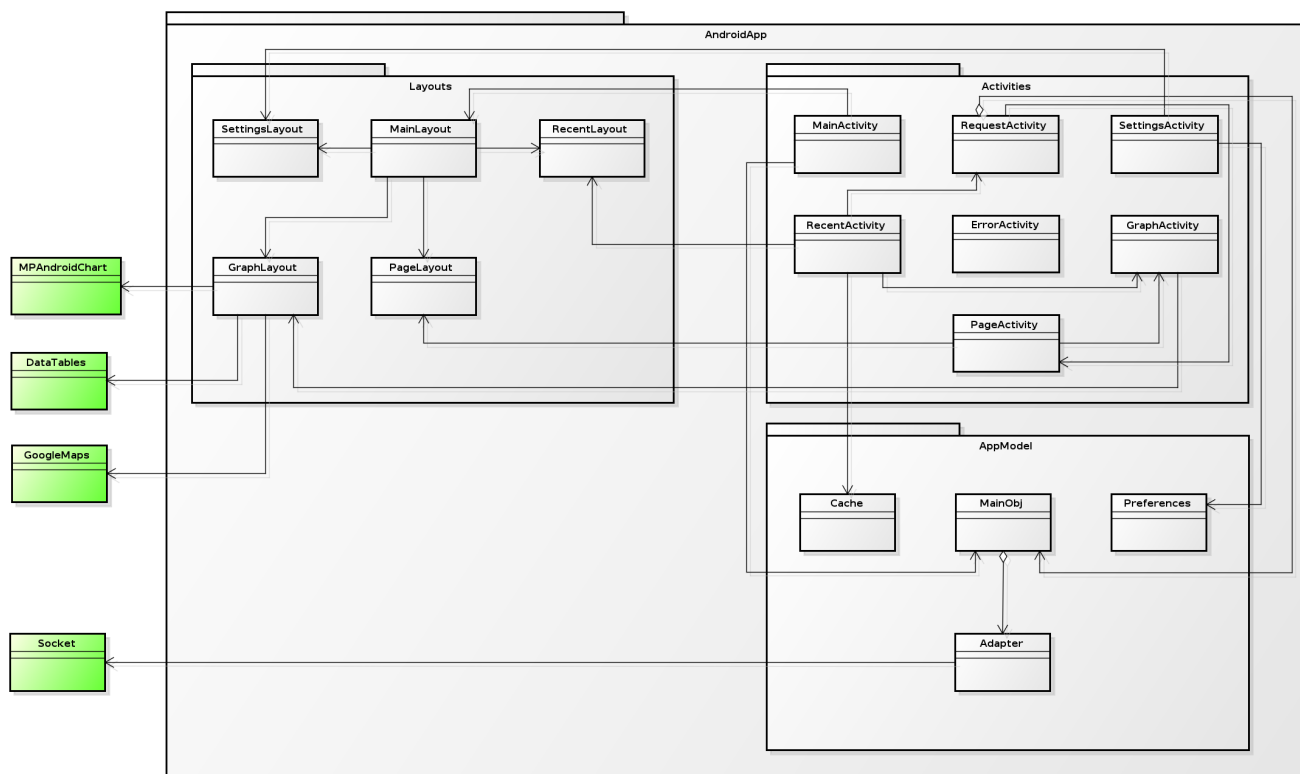


Figura 20: Diagramma delle classi *AndroidApp*

#### 4.2.1.1 Informazioni sul *package<sub>G</sub>*

##### 4.2.1.1.1 Descrizione

Questo *package<sub>G</sub>* contiene tutte le classi che si occupano di gestire la visualizzazione dei grafici Norris nell'applicazione per *smartphone<sub>G</sub> Android<sub>G</sub>*.

##### 4.2.1.1.2 *Package<sub>G</sub>* contenuti

- *AndroidApp::Activities*;
- *AndroidApp::AppModel*;
- *AndroidApp::Layouts*.



## 4.2.2 AndroidApp::Activities

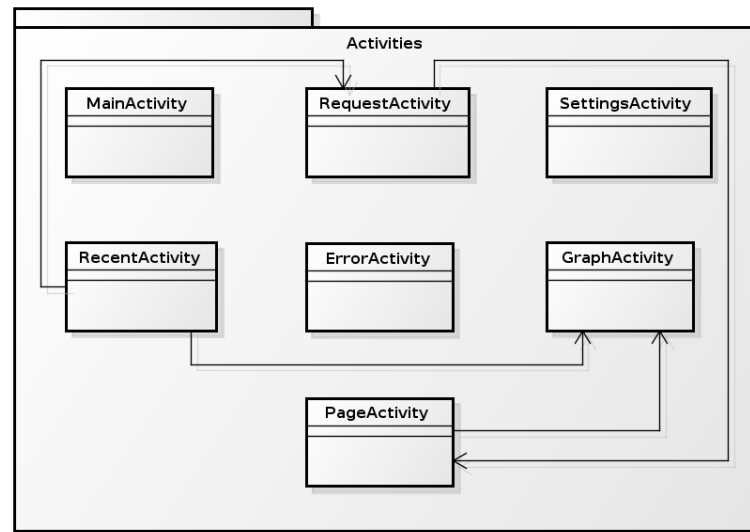


Figura 21: Componente Activities

### 4.2.2.1 Informazioni sul *package<sub>G</sub>*

#### 4.2.2.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi per la gestione dell'applicazione.

### 4.2.2.2 Classi

#### 4.2.2.2.1 AndroidApp::Activities::ErrorActivity

##### Descrizione

Questa classe contiene le activities per la gestione degli errori nell'applicazione.

##### Utilizzo

Questa classe si occupa degli errori nell'applicazione.

#### 4.2.2.2.2 AndroidApp::Activities::GraphActivity

##### Descrizione

Questa classe contiene le activities per la gestione dei grafici visualizzati nell'applicazione.

##### Utilizzo

Questa classe si occupa della gestione dei grafici nell'applicazione.

##### Relazioni con altre classi

- AndroidApp::Activities::PageActivity

Relazione entrante. Classe che si occupa della gestione dei grafici nell'applicazione.

– **AndroidApp::Activities::RecentActivity**

Relazione entrante. Classe che si occupa della gestione dei grafici nell'applicazione.

– **AndroidApp::Layouts::GraphLayout**

Relazione uscente. Classe che contiene i layout dei grafici.

#### 4.2.2.2.3 **AndroidApp::Activities::MainActivity**

##### **Descrizione**

Questa classe contiene le activities per la gestione dell'applicazione.

##### **Utilizzo**

Questa classe si occupa dell'avvio dell'applicazione.

##### **Relazioni con altre classi**

– **AndroidApp::Layouts::MainLayout**

Relazione uscente. Classe che contiene il layout dell'applicazione.

– **AndroidApp::AppModel::MainObj**

Relazione uscente. Classe che contiene i dati visualizzati dall'applicazione .

#### 4.2.2.2.4 **AndroidApp::Activities::PageActivity**

##### **Descrizione**

Questa classe contiene le activities per la gestione delle pagine contenenti i grafici nell'applicazione.

##### **Utilizzo**

Questa classe si occupa della gestione delle pagine nell'applicazione.

##### **Relazioni con altre classi**

– **AndroidApp::Activities::RequestActivity**

Relazione entrante. Classe che si occupa della gestione delle pagine contenenti i grafici nell'applicazione.

– **AndroidApp::Activities::GraphActivity**

Relazione uscente. Classe che si occupa della gestione dei grafici nell'applicazione.

– **AndroidApp::Layouts::PageLayout**

Relazione uscente. Classe che contiene il layout per la pagina contenente la lista dei grafici.

#### 4.2.2.2.5 AndroidApp::Activities::RecentActivity

##### Descrizione

Questa classe contiene le activities per la gestione dei contenuti visualizzati recentemente tramite l'applicazione.

##### Utilizzo

Questa classe si occupa delle risorse recenti visualizzate nell'applicazione.

##### Relazioni con altre classi

- **AndroidApp::AppModel::Cache**  
Relazione uscente. Classe per la memorizzazione dei dati sui grafici recenti.
- **AndroidApp::Activities::GraphActivity**  
Relazione uscente. Classe che si occupa della gestione dei grafici nell'applicazione.
- **AndroidApp::Layouts::RecentLayout**  
Relazione uscente. Classe che contiene il layout con i grafici visti di recente.
- **AndroidApp::Activities::RequestActivity**  
Relazione uscente. Questa classe gestisce le richieste dei dati nell'applicazione.

#### 4.2.2.2.6 AndroidApp::Activities::RequestActivity

##### Descrizione

Questa classe contiene le activities per la gestione delle richieste dei contenuti ad AppModel::MainObj.

##### Utilizzo

Questa classe si occupa di gestire le richieste dei dati nell'applicazione.

##### Relazioni con altre classi

- **AndroidApp::Activities::RecentActivity**  
Relazione entrante. Questa classe gestisce le richieste dei dati nell'applicazione.
- **AndroidApp::AppModel::MainObj**  
Relazione uscente. Classe che contiene i dati visualizzati dall'applicazione .
- **AndroidApp::Activities::PageActivity**  
Relazione uscente. Classe che si occupa della gestione delle pagine contenenti i grafici nell'applicazione.

#### 4.2.2.2.7 AndroidApp::Activities::SettingsActivity

##### Descrizione

Questa classe contiene le activities per la gestione delle impostazioni dell'applicazione.

##### Utilizzo

Questa classe si occupa delle impostazioni dell'applicazione.

##### Relazioni con altre classi

- **AndroidApp::AppModel::Preferences**  
Relazione uscente. Classe che contiene le impostazioni scelte dall'utente.
- **AndroidApp::Layouts::SettingsLayout**  
Relazione uscente. Classe che contiene il layout del menu impostazioni.

#### 4.2.3 AndroidApp::AppModel

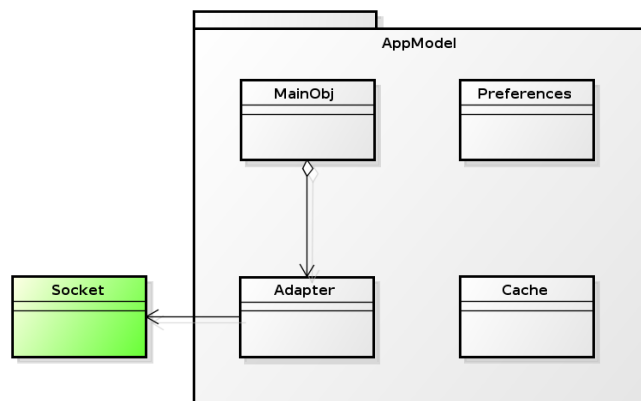


Figura 22: Componente AppModel

#### 4.2.3.1 Informazioni sul *package<sub>G</sub>*

##### 4.2.3.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi per mantenere e gestire i dati necessari all'applicazione.

##### 4.2.3.2 Classi

#### 4.2.3.2.1 AndroidApp::AppModel::Adapter

##### Descrizione

Questa classe si occupa dell'interazione dell'applicazione con il *server<sub>G</sub>* tramite *Socket.io<sub>G</sub>*.

##### Utilizzo

Sarà utilizzata da MainObject per memorizzare i dati recenti.

**Relazioni con altre classi**

- **AndroidApp::AppModel::MainObj**

Relazione entrante. Classe per l'interfacciamento a *Socket.ioG*.

**4.2.3.2.2 AndroidApp::AppModel::Cache****Descrizione**

Questa classe contiene i dati già richiesti e visualizzati dall'utente.

**Utilizzo**

Sarà utilizzata da *Activities::RecentActivity* per memorizzare i dati recenti.

**Relazioni con altre classi**

- **AndroidApp::Activities::RecentActivity**

Relazione entrante. Classe per la memorizzazione dei dati sui grafici recenti.

**4.2.3.2.3 AndroidApp::AppModel::MainObj****Descrizione**

Questa classe si occupa della trasmissione dei dati dell'applicazione.

**Utilizzo**

Questa classe si occupa di gestire i dati dell'applicazione.

**Relazioni con altre classi**

- **AndroidApp::Activities::MainActivity**

Relazione entrante. Classe che contiene i dati visualizzati dall'applicazione .

- **AndroidApp::Activities::RequestActivity**

Relazione entrante. Classe che contiene i dati visualizzati dall'applicazione .

- **AndroidApp::AppModel::Adapter**

Relazione uscente. Classe per l'interfacciamento a *Socket.ioG*.

**4.2.3.2.4 AndroidApp::AppModel::Preferences****Descrizione**

Questa classe contiene le impostazioni dell'applicazione selezionate dall'utente.

**Utilizzo**

Sarà utilizzata da *Activities::SettingsActivity* per memorizzare le preferenze.

**Relazioni con altre classi**

- **AndroidApp::Activities::SettingsActivity**

Relazione entrante. Classe che contiene le impostazioni scelte dall'utente.

#### 4.2.4 AndroidApp::Layouts

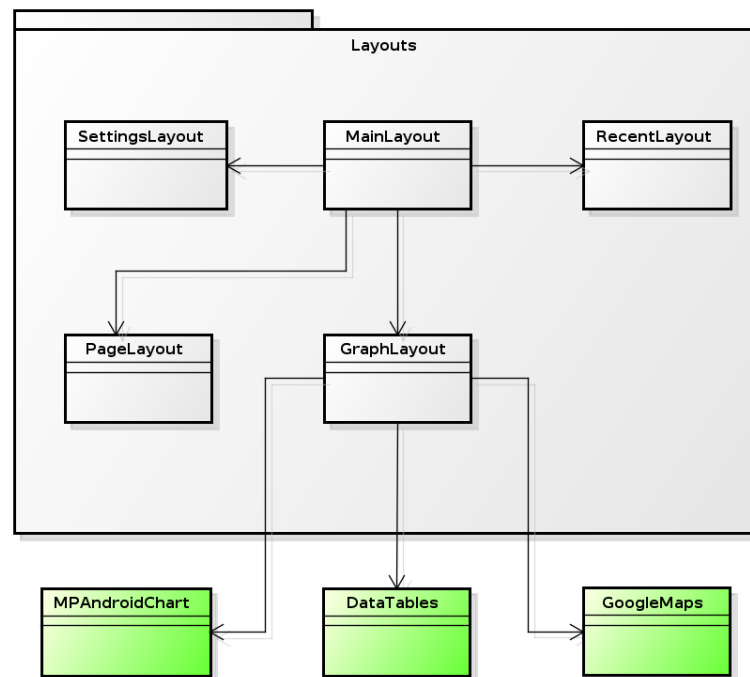


Figura 23: Componente Layouts

##### 4.2.4.1 Informazioni sul *package<sub>G</sub>*

###### 4.2.4.1.1 Descrizione

Questo *package<sub>G</sub>* contiene le classi con le view usate dall'applicazione.

###### 4.2.4.2 Classi

###### 4.2.4.2.1 AndroidApp::Layouts::GraphLayout

###### Descrizione

Questa classe contiene il layout di visualizzazione del grafico selezionato dall'utente.

###### Utilizzo

Sarà utilizzata dalla classe Activities::GraphActivity per la gestione del layout di visualizzazione del grafico.

###### Relazioni con altre classi

- **AndroidApp::Activities::GraphActivity**  
Relazione entrante. Classe che contiene i layout dei grafici.
- **AndroidApp::Layouts::MainLayout**  
Relazione entrante. Classe che contiene i layout dei grafici.

#### 4.2.4.2.2 **AndroidApp::Layouts::MainLayout**

##### **Descrizione**

Questa classe contiene il layout di base dell'applicazione.

##### **Utilizzo**

Sarà utilizzata dalla classe Activities::MainActivity per la gestione del layout.

##### **Relazioni con altre classi**

- **AndroidApp::Activities::MainActivity**  
Relazione entrante. Classe che contiene il layout dell'applicazione.
- **AndroidApp::Layouts::GraphLayout**  
Relazione uscente. Classe che contiene i layout dei grafici.
- **AndroidApp::Layouts::PageLayout**  
Relazione uscente. Classe che contiene il layout per la pagina contenente la lista dei grafici.
- **AndroidApp::Layouts::RecentLayout**  
Relazione uscente. Classe che contiene il layout con i grafici visti di recente.
- **AndroidApp::Layouts::SettingsLayout**  
Relazione uscente. Classe che contiene il layout del menu impostazioni.

#### 4.2.4.2.3 **AndroidApp::Layouts::PageLayout**

##### **Descrizione**

Questa classe contiene il layout del menu che visualizzerà i dati della pagina scelta dall'utente.

##### **Utilizzo**

Sarà utilizzata dalla classe Activities::PageActivity per la gestione del layout di visualizzazione della pagina.

##### **Relazioni con altre classi**

- **AndroidApp::Layouts::MainLayout**  
Relazione entrante. Classe che contiene il layout per la pagina contenente la lista dei grafici.
- **AndroidApp::Activities::PageActivity**  
Relazione entrante. Classe che contiene il layout per la pagina contenente la lista dei grafici.

#### 4.2.4.2.4 **AndroidApp::Layouts::RecentLayout**

##### **Descrizione**

Questa classe include il layout del menu contenente i grafici recenti visualizzati tramite l'applicazione.

##### **Utilizzo**

Sarà utilizzata dalla classe `Activities::RecentActivity` per la gestione del layout del menu recenti.

##### **Relazioni con altre classi**

- **AndroidApp::Layouts::MainLayout**

Relazione entrante. Classe che contiene il layout con i grafici visti di recente.

- **AndroidApp::Activities::RecentActivity**

Relazione entrante. Classe che contiene il layout con i grafici visti di recente.

#### 4.2.4.2.5 **AndroidApp::Layouts::SettingsLayout**

##### **Descrizione**

Questa classe include il layout del menu contenente le impostazioni dell'applicazione.

##### **Utilizzo**

Sarà utilizzata dalla classe `Activities::SettingsActivity` per la gestione del layout del menu impostazioni.

##### **Relazioni con altre classi**

- **AndroidApp::Layouts::MainLayout**

Relazione entrante. Classe che contiene il layout del menu impostazioni.

- **AndroidApp::Activities::SettingsActivity**

Relazione entrante. Classe che contiene il layout del menu impostazioni.



## 5 Comunicazione $client_G$ - $server_G$

Vengono di seguito presentati i diagrammi di sequenza prodotti durante la fase di progettazione, i quali descrivono la gestione delle comunicazioni tra  $server_G$  e  $client_G$ . Per illustrare le comunicazioni sono stati creati tre diagrammi: uno che rappresenta la prima richiesta di pagina, uno per la gestione della richiesta all'interno del  $server_G$  ed infine un diagramma che rappresenta l'inoltro degli aggiornamenti dei dati.

### 5.1 Prima richiesta di pagina

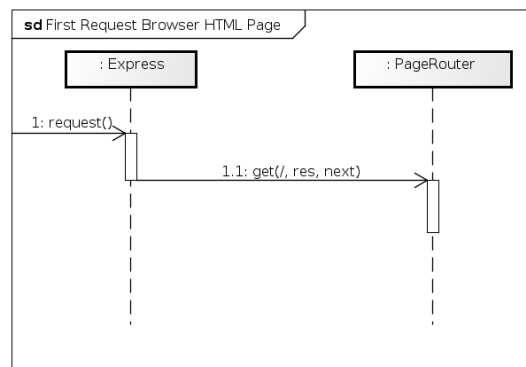


Figura 24: Diagramma di sequenza - Prima richiesta di pagina

La prima richiesta di pagina del  $client_G$ , sia esso un browser o l'applicazione  $Android_G$ , avviene tramite una chiamata  $HTTP_G$  al  $server_G$ , il quale tramite l'utilizzo di  $Express_G$  riceve la richiesta.

Una volta ricevuta, quest'ultima sarà analizzata e gestita tramite la classe `PageRouter` che si occuperà di completare, per quanto concerne il browser, il  $template_G$  html con l'url del  $server_G$  e di restituirlo tramite l'ausilio di  $Express_G$  al  $client_G$  che si occuperà di eseguire una seconda richiesta per i dati della pagina mentre, per quanto riguarda l'applicativo  $Android_G$  verrà automaticamente restituito il  $JSON_G$  contenente i dati grezzi della pagina.

## 5.2 *Routing<sub>G</sub>*

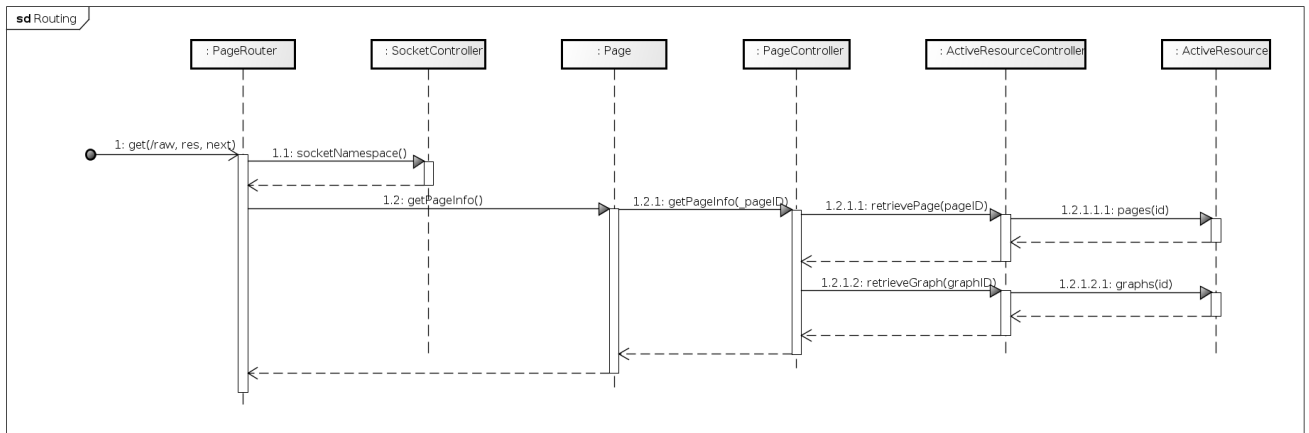


Figura 25: Diagramma di sequenza - *Routing<sub>G</sub>* di una richiesta

La gestione delle richieste di pagina valide partirà dalla classe *PageRouter* che invierà un *template<sub>G</sub>* contenente l'*URL<sub>G</sub>* del *server<sub>G</sub>* di Norris al *client<sub>G</sub>*. Questa pagina, tramite l'ausilio di *AngularJS<sub>G</sub>*, si occuperà di inviare un'ulteriore richiesta di tipo GET al *server<sub>G</sub>* che restituirà un pacchetto *JSON<sub>G</sub>* contenente i dati grezzi della pagina richiesta dall'utente. Questi dati verranno poi processati interamente dal lato *client<sub>G</sub>* che si occuperà di strutturare la pagina in maniera adeguata alla visualizzazione dei grafici tramite elementi *HTML<sub>G</sub>*. All'interno di ognuno di essi saranno inoltre presenti alcuni script che permetteranno la connessione *WebSocket<sub>G</sub>* necessaria allo scopo di aggiornare i grafici stessi.

## 5.3 Inoltro notifiche *push<sub>G</sub>*

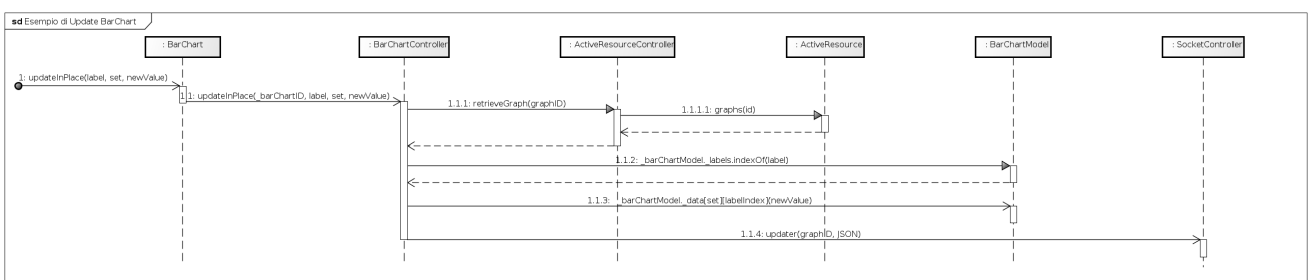


Figura 26: Diagramma di sequenza - Aggiornamento di un grafico *Bar Chart<sub>G</sub>*

Lo sviluppatore avrà a disposizione dei metodi di update specifici per ogni tipologia di grafico che potrà chiamare una volta creato l'oggetto che vorrà aggiornare. Ad ogni chiamata di aggiornamento la classe Controller dell'oggetto su cui viene chiamato il metodo si occuperà di andare a cercare quest'ultimo all'interno dell'istanza di Norris tramite l'ausilio della classe *ActiveResourceController*. A questo punto verrà aggiornata la copia in locale, dopodichè verrà richiamata la funzione di *SocketController* che invierà, sulla stanza identificata con l'id dell'oggetto da aggiornare, un pacchetto *JSON<sub>G</sub>* specifico per la tipologia di aggiornamento richiesta. Tale pacchetto verrà ricevuto dal



$client_G$  che si occuperà, una volta identificato come aggiornamento, di modificare il grafico in maniera congrua.

## 6 Diagrammi di attività

Vengono di seguito illustrati i diagrammi di attività prodotti durante la fase di progettazione, i quali descrivono le iterazioni dell'utente finale e dello sviluppatore con il sistema Norris.

È stato ritenuto opportuno suddividere i diagrammi in tre categorie principali, in modo analogo a quanto fatto nella descrizione dei casi d'uso del documento *AnalisiRequisiti\_ver5.0.0*:

- **Funzionalità Sviluppatore:** verranno illustrate le differenti possibilità di interazione fra lo sviluppatore e il *framework<sub>G</sub>* Norris;
- **Funzionalità Utente:** verranno illustrate le interazioni possibili operate da parte delle utente tramite il *client<sub>G</sub>* web;
- **Applicazione *Android<sub>G</sub>*:** verranno mostrate le varie operazioni possibili da parte degli utenti tramite l'applicativo *Android<sub>G</sub>*.

Inizialmente per ogni categoria verrà fornito uno schema ad alto livello, per poi scendere nel dettaglio tramite sotto-diagrammi più specifici. Le attività che verranno esplose sono marcate dal simbolo apposito.

Al fine di rendere il diagramma più facilmente leggibile, è stata considerata implicita la possibilità per l'utente di chiudere in qualsiasi momento la connessione al *server<sub>G</sub>*, per esempio chiudendo la finestra del browser o terminando l'applicazione *Android<sub>G</sub>*.

### 6.1 Funzionalità Sviluppatore

Vengono mostrate e descritte di seguito le operazioni che possono essere eseguite da uno sviluppatore mediante l'uso del *framework<sub>G</sub>* Norris:

### 6.1.1 Attività principali

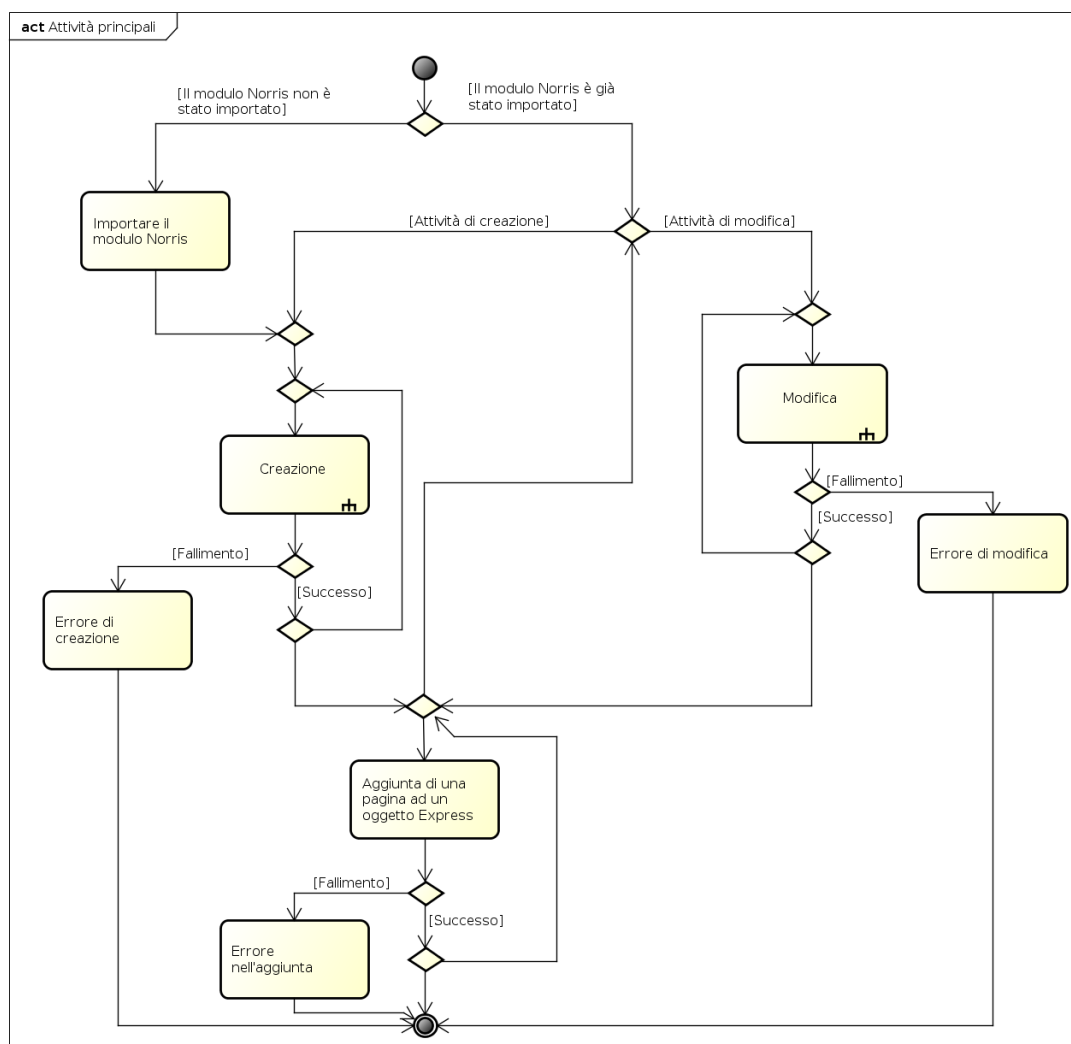


Figura 27: Diagramma di attività - Attività principali sviluppatore

Norris fornisce delle funzionalità a uno sviluppatore che volesse servirsi di grafici aggiornabili in tempo reale. Per fare uso delle funzionalità del *framework<sub>G</sub>*, è necessario inizialmente importare il modulo nel proprio script. Successivamente sarà possibile accedere alle funzionalità di creazione e modifica di pagine e grafici, descritte in modo più approfondito nelle sezioni 6.1.2 - *Attività di creazione* e 6.1.7 - *Attività di modifica*.

### 6.1.2 Attività di creazione

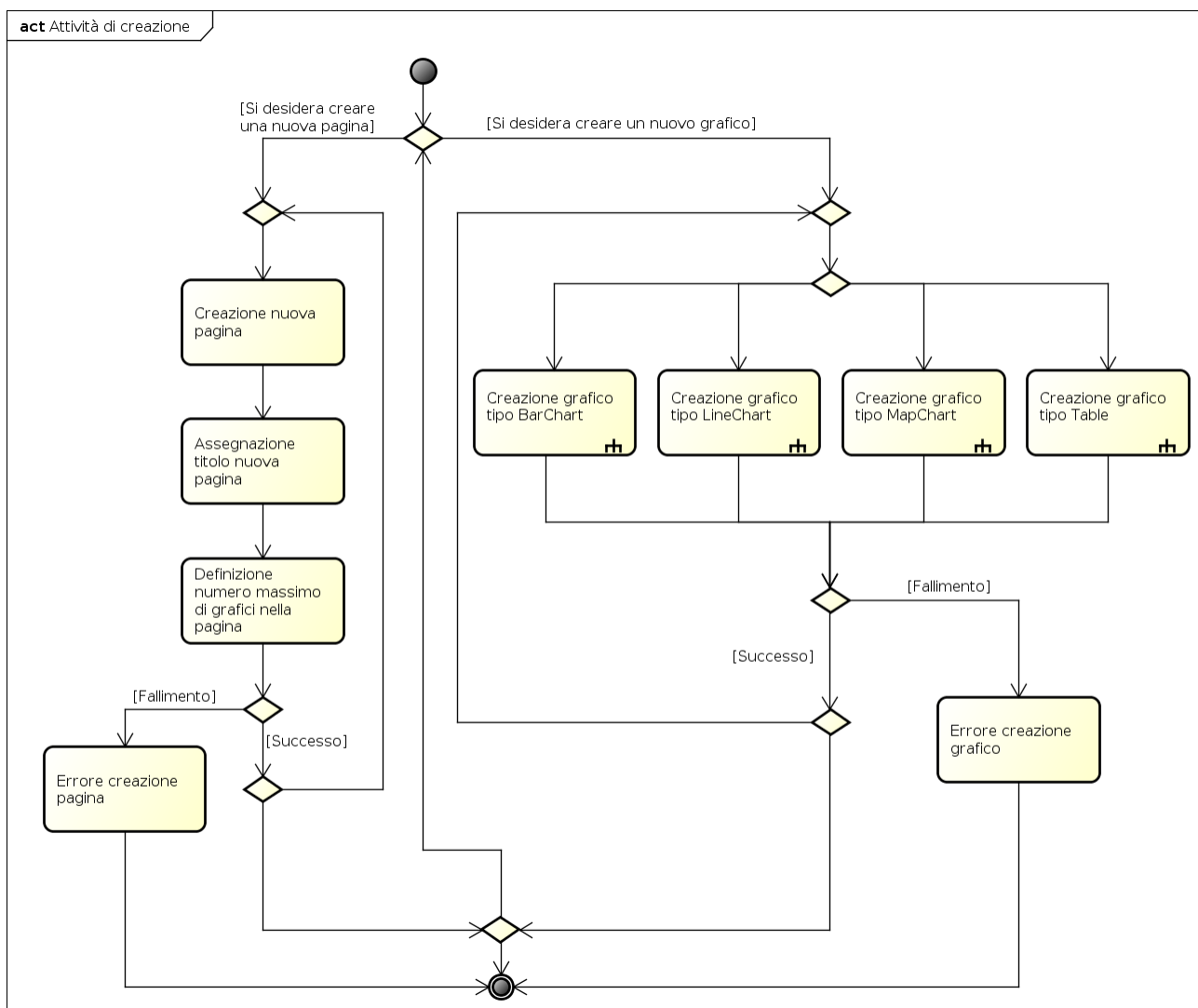


Figura 28: Diagramma di attività - Funzionalità di creazione

Dopo aver richiesto le funzionalità di Norris, lo sviluppatore può creare pagine e grafici mediante le apposite *APIG*. L'attività di creazione si distingue nelle due sotto attività di creazione pagina e grafico.

Per creare una pagina, lo sviluppatore deve istanziare un oggetto di tipo *page* e assegnarli un titolo. Per creare un grafico invece, lo sviluppatore potrà scegliere il tipo di grafico da creare, invocando l'apposita funzione di creazione del tipo desiderato, analizzate in dettaglio nei diagrammi successivi.

Ogni grafico avrà una serie di opzioni di istanziamento che sarà possibile configurare manualmente in fase di creazione, altrimenti verranno utilizzati valori di default. In caso di fallimento, si genererà un errore, altrimenti si potrà procedere con un' altra operazione, o ripetere una creazione.

### 6.1.3 Attività di creazione *Bar Chart<sub>G</sub>*

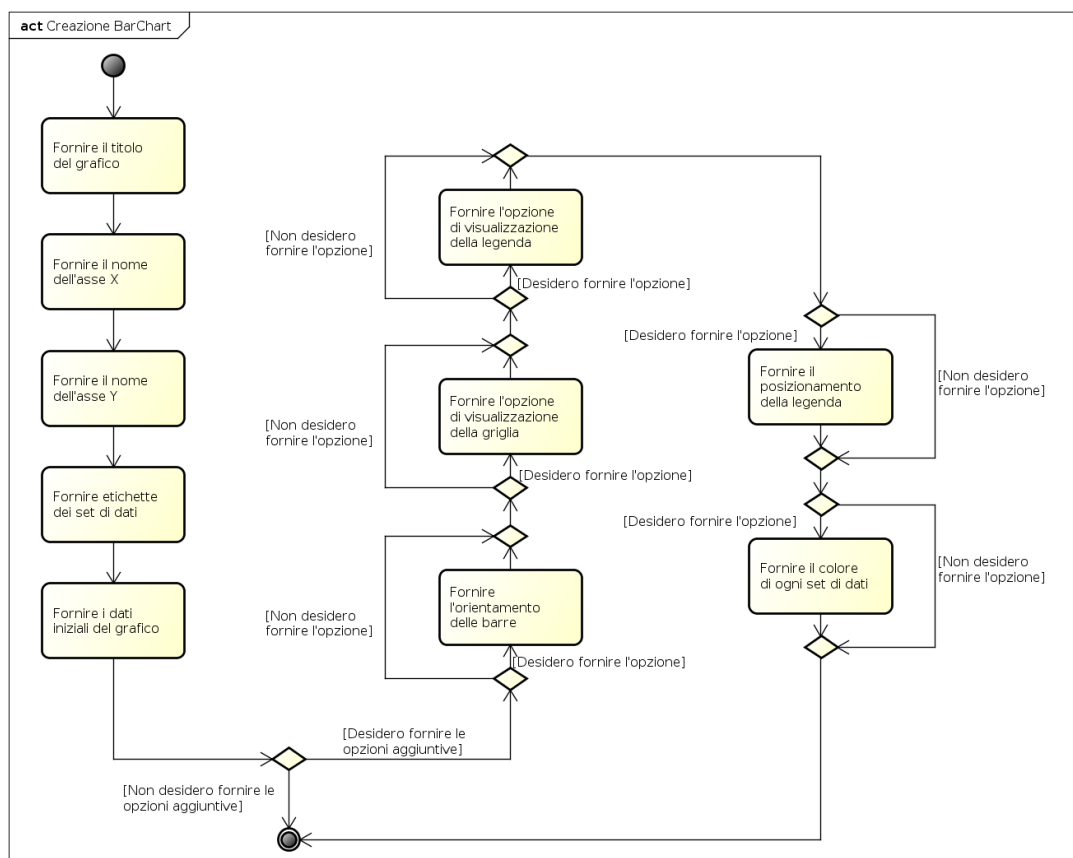


Figura 29: Diagramma di attività - Funzionalità di creazione *Bar Chart<sub>G</sub>*

Lo sviluppatore che volesse istanziare un grafico di tipo *Bar Chart<sub>G</sub>* ha alcune proprietà che dovrà inserire obbligatoriamente come:

- Titolo;
- Nome dell'asse X;
- Nome dell'asse Y;
- Etichette dei set dei dati;
- I primi dati del grafico.

Oltre a queste lo sviluppatore avrà la possibilità di impostare alcuni parametri opzionali come:

- L'orientamento, orrizzontale o verticale;
- La visualizzazione della griglia;
- La visualizzazione della legenda;
- Il posizionamento della legenda;

- Il colore di ogni set di dati.

Nel caso in cui essi non vengano impostati manualmente Norris si occuperà di settare automaticamente alcuni valori di default.



### 6.1.4 Attività di creazione *Line Chart<sub>G</sub>*

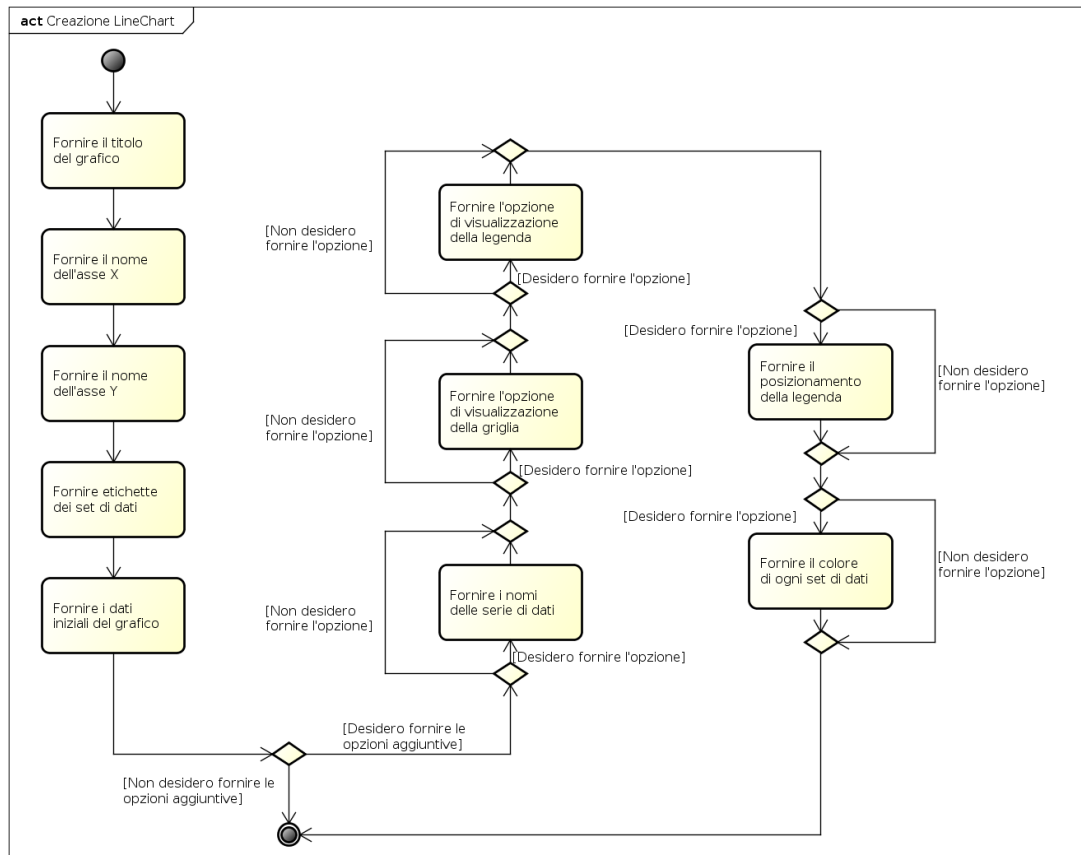


Figura 30: Diagramma di attività - Funzionalità di creazione *Line Chart<sub>G</sub>*

Lo sviluppatore che volesse istanziare un grafico di tipo *Line Chart<sub>G</sub>* ha alcune proprietà che dovrà inserire obbligatoriamente come:

- Titolo;
- Nome dell'asse X;
- Nome dell'asse Y;
- Etichette dei set dei dati;
- I primi dati del grafico.

Oltre a queste lo sviluppatore avrà la possibilità di impostare alcuni parametri opzionali come:

- I nomi delle serie di dati;
- La visualizzazione della griglia;
- La visualizzazione della legenda;
- Il posizionamento della legenda;

- Il colore di ogni set di dati.

Nel caso in cui essi non vengano impostati manualmente Norris si occuperà di settare automaticamente alcuni valori di default.

### 6.1.5 Attività di creazione *Map Chart<sub>G</sub>*

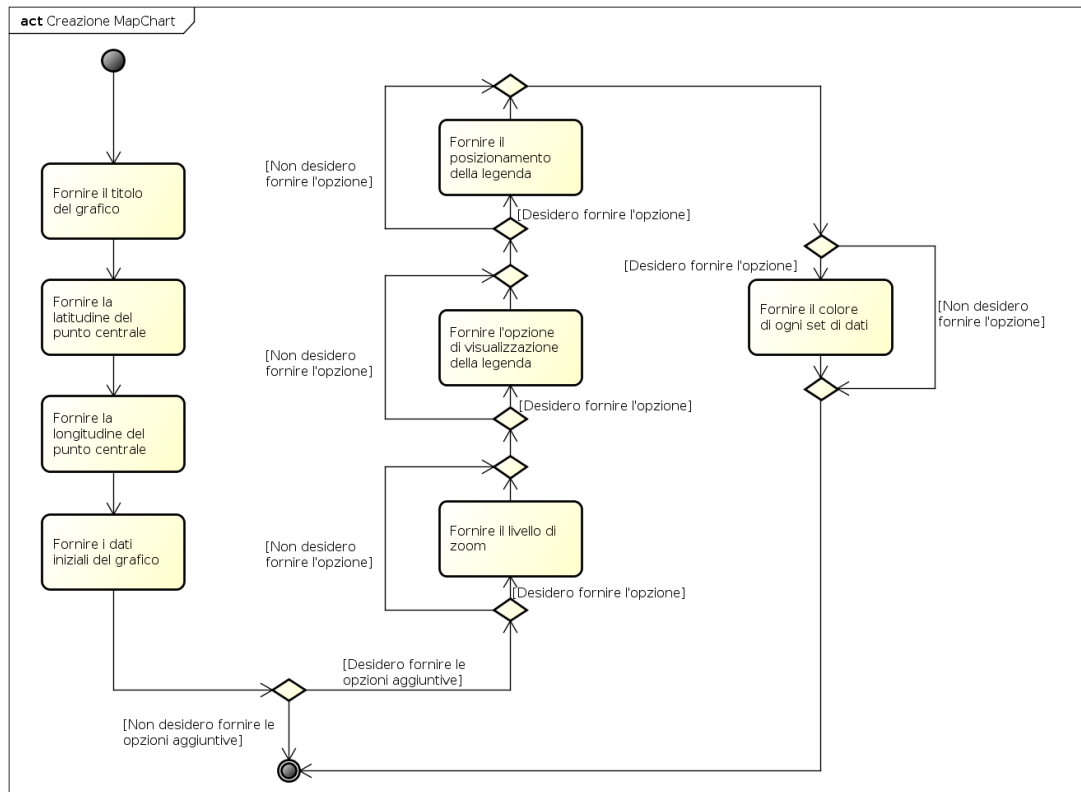


Figura 31: Diagramma di attività - Funzionalità di creazione *Map Chart<sub>G</sub>*

Lo sviluppatore che volesse istanziare un grafico di tipo *Map Chart<sub>G</sub>* ha alcune proprietà che dovrà inserire obbligatoriamente come:

- Titolo;
- Latitudine del punto centrale;
- Longitudine del punto centrale;
- I primi dati del grafico.

Oltre a queste lo sviluppatore avrà la possibilità di impostare alcuni parametri opzionali come:

- Il livello di zoom;
- La visualizzazione della legenda;
- Il posizionamento della legenda;
- Il colore di ogni set di dati.

Nel caso in cui essi non vengano impostati manualmente Norris si occuperà di settare automaticamente alcuni valori di default.

### 6.1.6 Attività di creazione *Table<sub>G</sub>*

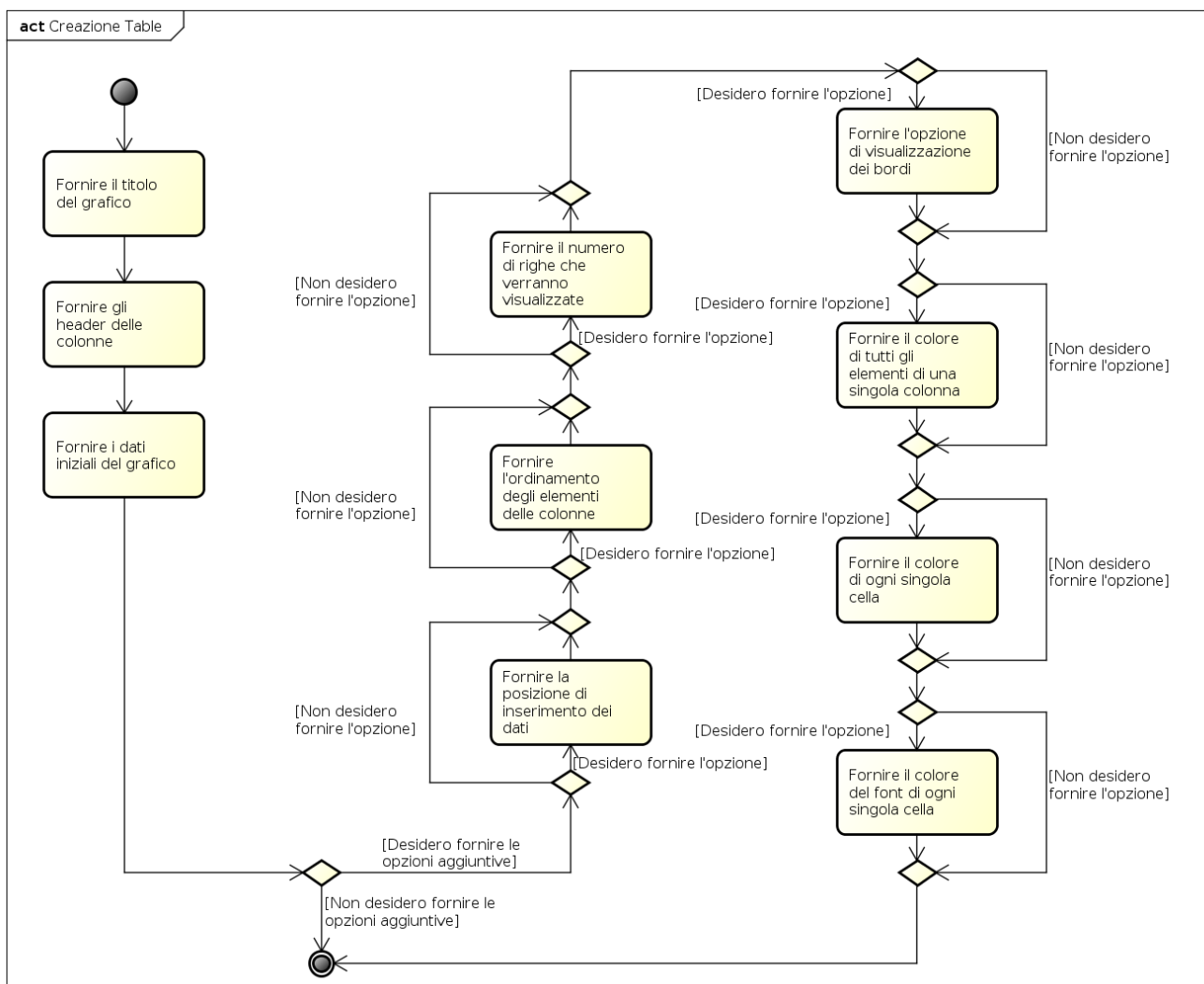


Figura 32: Diagramma di attività - Funzionalità di creazione *Table<sub>G</sub>*

Lo sviluppatore che volesse istanziare un grafico di tipo *Table<sub>G</sub>* ha alcune proprietà che dovrà inserire obbligatoriamente come:

- Titolo;
- Gli headers delle colonne;
- I primi dati del grafico.

Oltre a queste lo sviluppatore avrà la possibilità di impostare alcuni parametri opzionali come:

- La posizione di inserimento dei dati, testa o coda;
- L'ordinamento degli elementi delle colonne;
- Il numero di righe che verranno visualizzate;

- La visualizzazione dei bordi;
- Il colore di tutti gli elementi di una singola colonna;
- Il colore di ogni singola cella;
- il colore del font di ogni singola cella.

Nel caso in cui essi non vengano impostati manualmente Norris si occuperà di settare automaticamente alcuni valori di default.

### 6.1.7 Attività di modifica

Trattandosi di una libreria Norris non offre molte possibilità di modifica post creazione in quanto sarà sufficiente modificare i parametri nei costruttori per ottenere le modifiche desiderate.

Tuttavia è stato ritenuto necessario offrire allo sviluppatore la possibilità di eseguire delle modifiche alla pagina, ossia l'aggiunta dei grafici, e le funzionalità di aggiornamento ad ogni grafico. Per quanto concerne la rimozione dei grafici non si è vista la necessità di aggiungere nessun metodo in quanto lo sviluppatore potrà semplicemente cancellare l'aggiunta del grafico che desidera rimuovere.

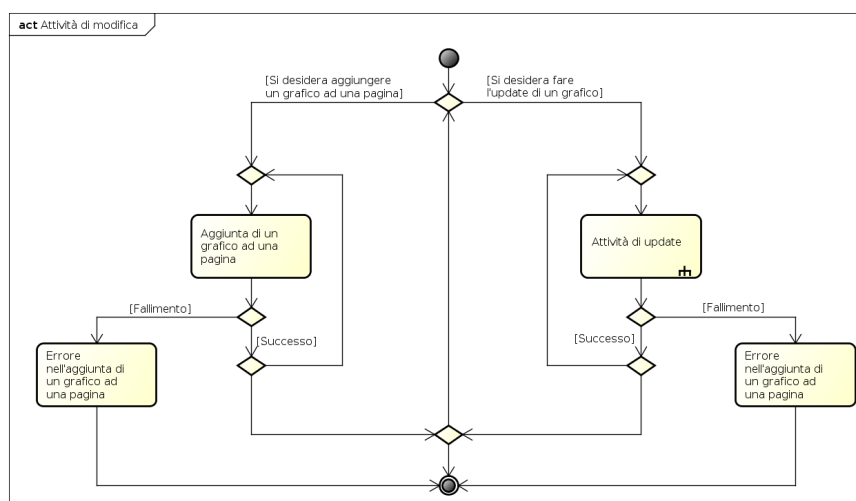


Figura 33: Diagramma di attività - Attività di modifica

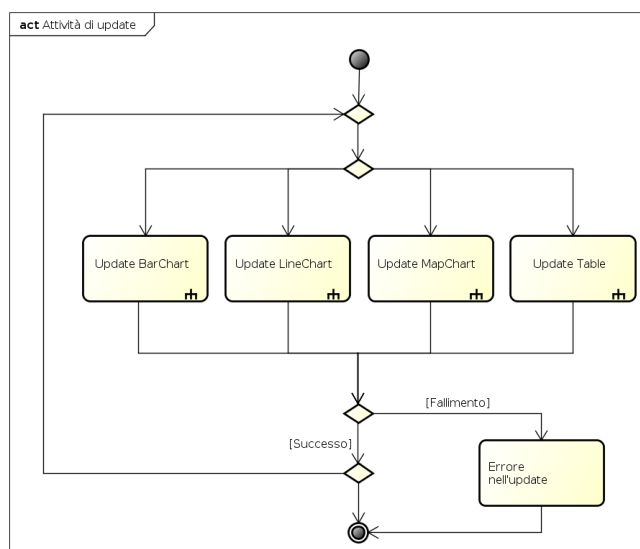


Figura 34: Diagramma di attività - Attività di update

### 6.1.8 Attività di update grafico *Bar Chart<sub>G</sub>*

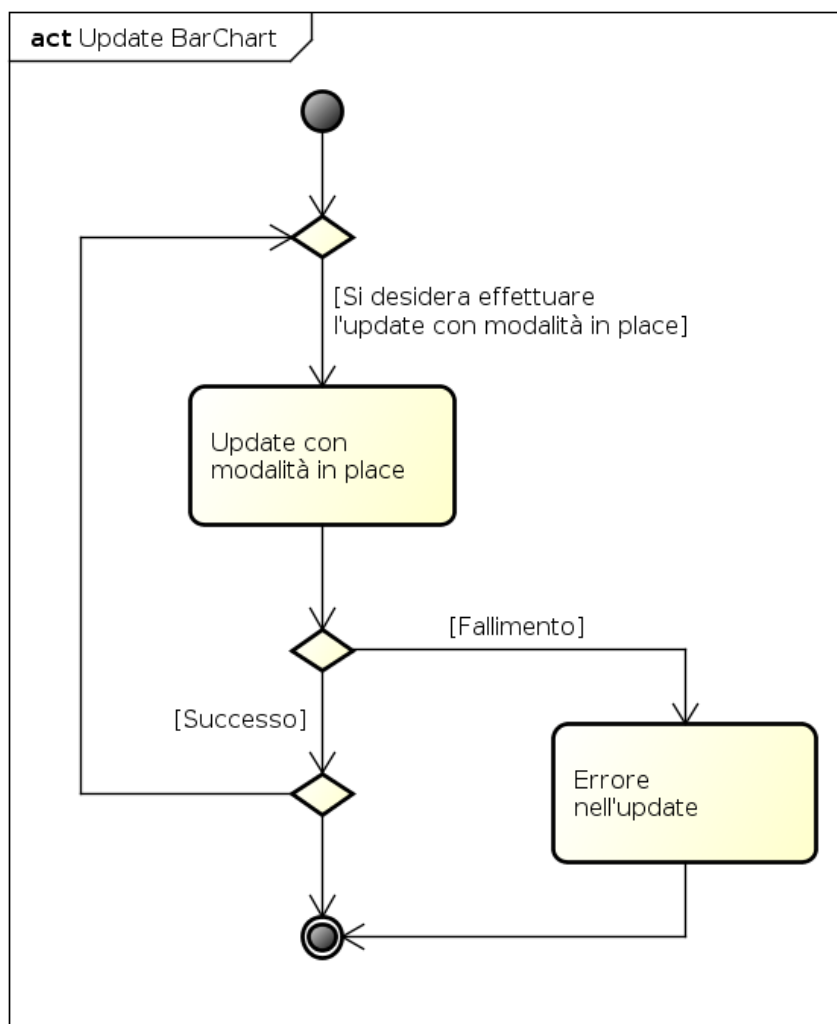


Figura 35: Diagramma di attività - Update grafico *Bar Chart<sub>G</sub>*

Lo sviluppatore va ad eseguire un update dei dati al grafico *Bar Chart<sub>G</sub>*.

### 6.1.9 Attività di update grafico *Line Chart<sub>G</sub>*

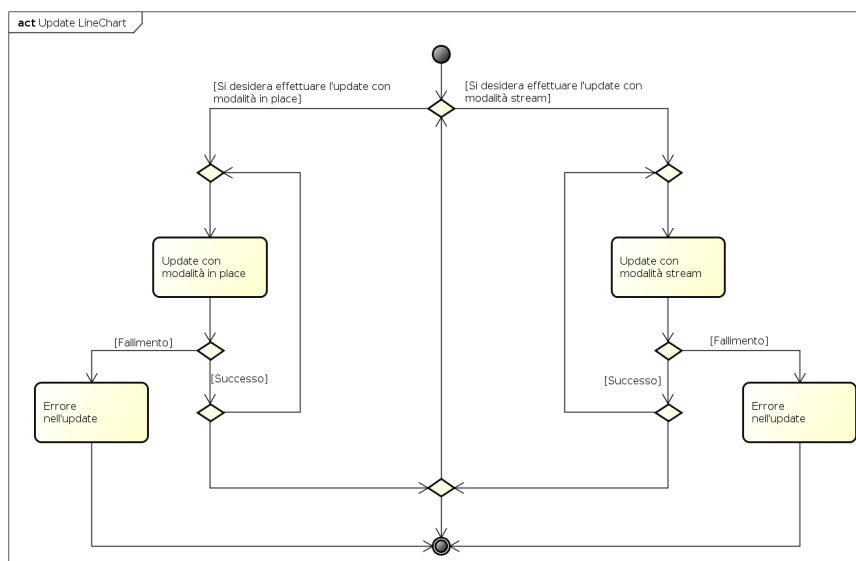


Figura 36: Diagramma di attività - Update grafico *Line Chart<sub>G</sub>*

Lo sviluppatore va ad eseguire un update dei dati al grafico *Line Chart<sub>G</sub>*.

### 6.1.10 Attività di update grafico *Map Chart<sub>G</sub>*

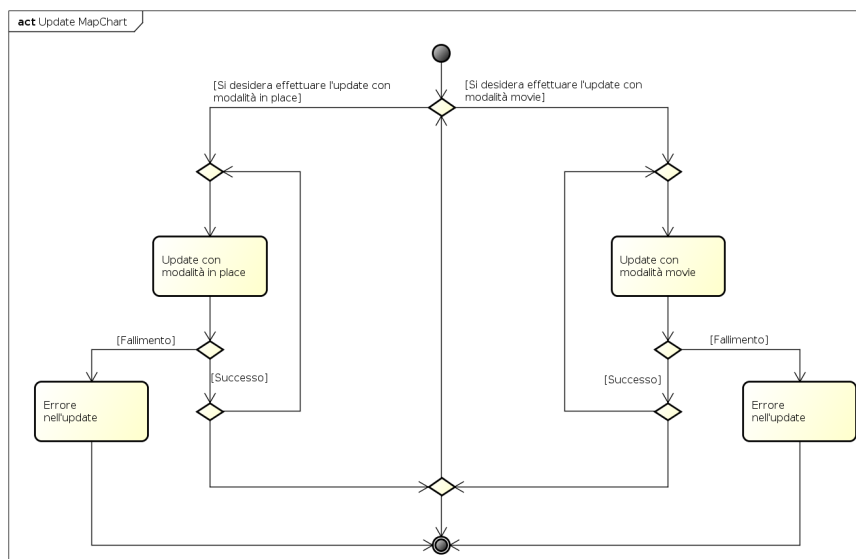


Figura 37: Diagramma di attività - Update grafico *Map Chart<sub>G</sub>*

Lo sviluppatore va ad eseguire un update dei dati al grafico *Map Chart<sub>G</sub>*.



### 6.1.11 Attività di update grafico *Table<sub>G</sub>*

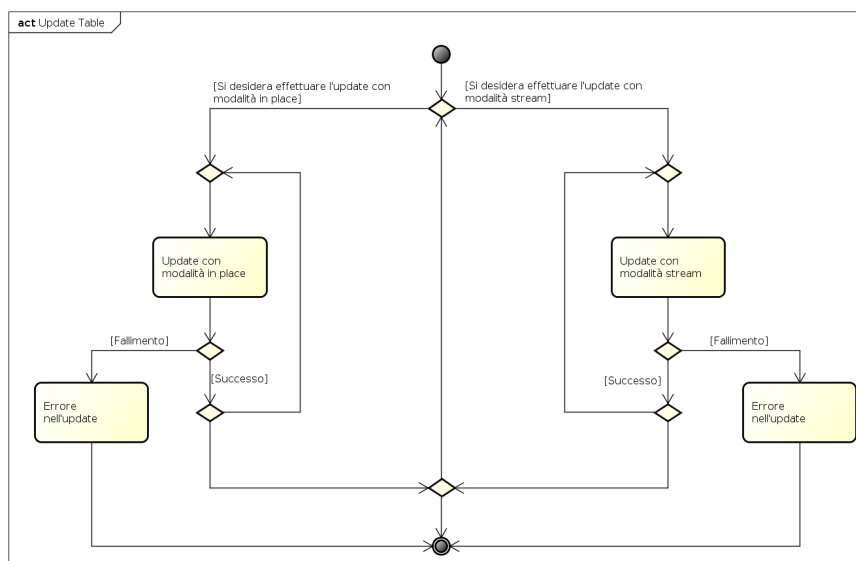


Figura 38: Diagramma di attività - Update grafico *Table<sub>G</sub>*

Lo sviluppatore va ad eseguire un update dei dati al grafico *Table<sub>G</sub>*.

## 6.2 Funzionalità Utente

Vengono mostrate e descritte di seguito le operazioni che possono essere eseguite da un utente finale che usufruisce dei grafici creati mediante l'uso del *framework<sub>G</sub>* Norris:

### 6.2.1 Funzionalità generali

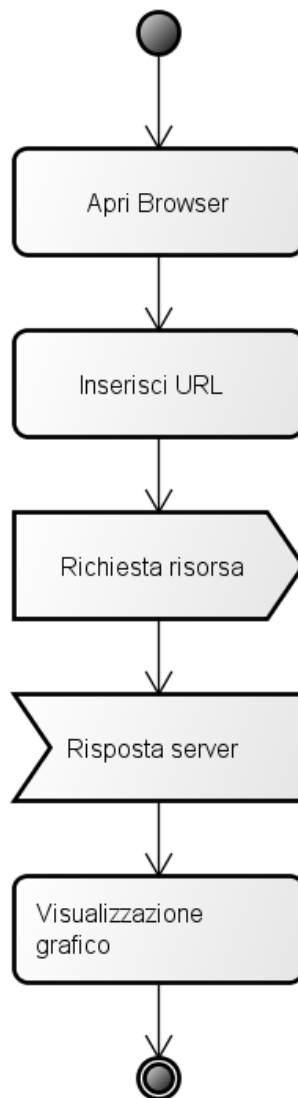


Figura 39: Diagramma di attività - Funzionalità

L'utente finale può visualizzare un grafico creato mediante il *framework<sub>G</sub>* Norris senza effettuare particolari operazioni. Il browser dell'utente effettuerà la chiamata *HTTP<sub>G</sub>* necessaria ad ottenere la risorsa specificata, aprendo il *socket<sub>G</sub>* necessario per ottenere gli aggiornamenti.

### 6.2.2 App *Android<sub>G</sub>*

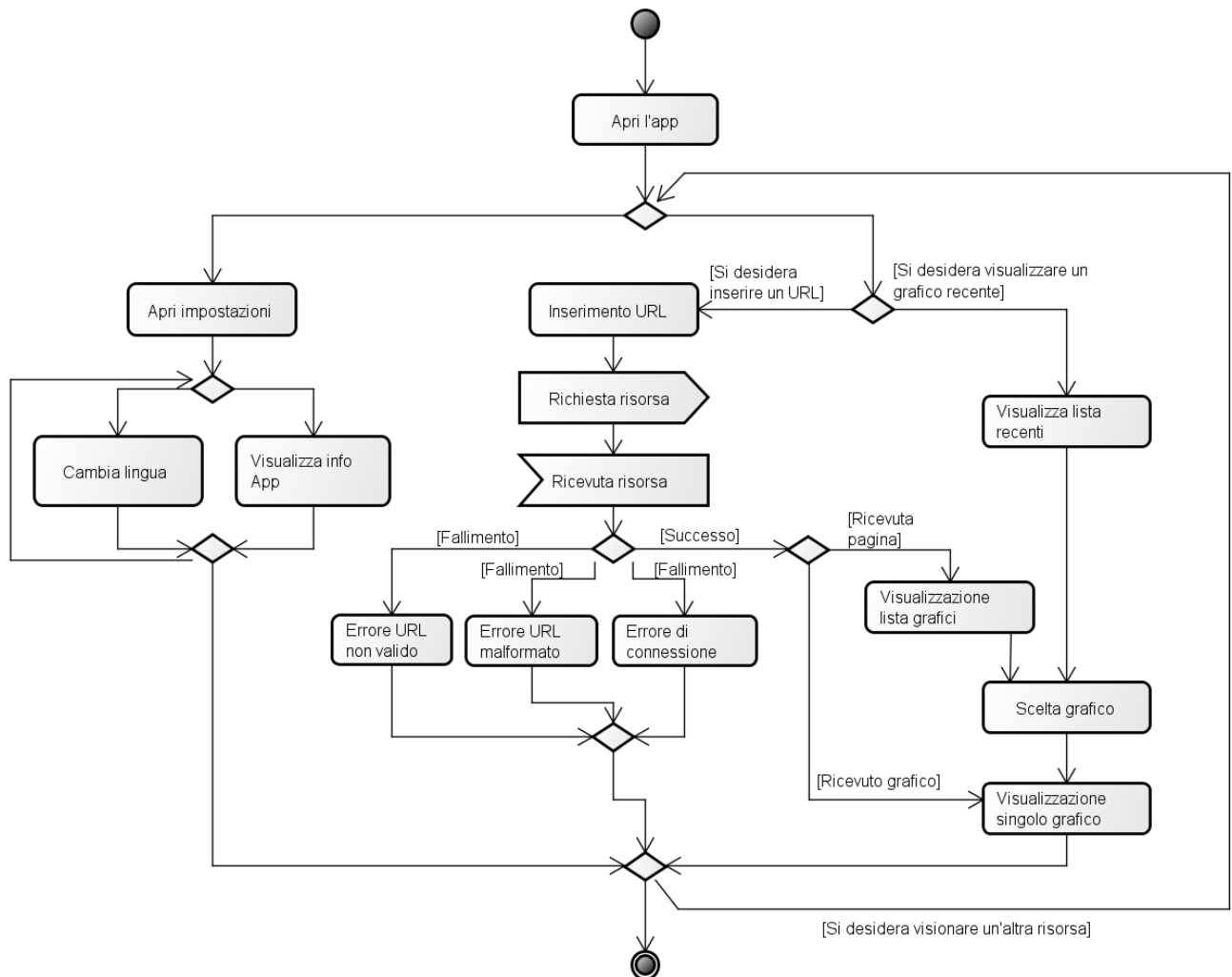


Figura 40: Diagramma di attività - App *Android<sub>G</sub>*

Dopo aver aperto l'app, l'utente avrà la possibilità di inserire l' $URL_G$  di una risorsa Norris per visualizzarla, o di selezionare una risorsa precedentemente visualizzata, se disponibile, dalla lista Recenti. Nel primo caso, l'app effettuerà una richiesta  $HTTP_G$  all'indirizzo specificato, che (se valido) ritornerà la risorsa richiesta. Se l' $URL_G$  inserita non fosse invece corretta, l'app restituirà errori specifici per varie cause:

- $URL_G$  non valido, nel caso non identificasse nessuna risorsa;
- $URL_G$  malformato, nel caso non corrispondesse alla forma di un  $URL_G HTTP_G$ ;
- Errore di connessione, nel caso la connessione fosse assente o fosse interrotta per motivi esterni.

Nel caso la risorsa ritornata invece fosse valida e fosse una pagina, l'app provvederà a mostrare una lista dei grafici in essa contenuti tra cui scegliere un singolo. Successivamente, o nel caso la risorsa ritornata fosse invece un grafico, l'app mostrerà direttamente quest'ultimo con l'orario di ultimo aggiornamento. Sarà in ogni momento

possibile tornare al passo precedente mediante il pulsante Back del sistema operativo *Android*. Dalla pagina iniziale sarà anche possibile accedere al menu delle impostazioni, in cui sarà possibile visualizzare delle informazioni sul gruppo **FlameTech Inc.** e le impostazioni sulla lingua. Sarà possibile scegliere tra inglese e italiano.

### 6.2.3 Caso applicativo APS Holding

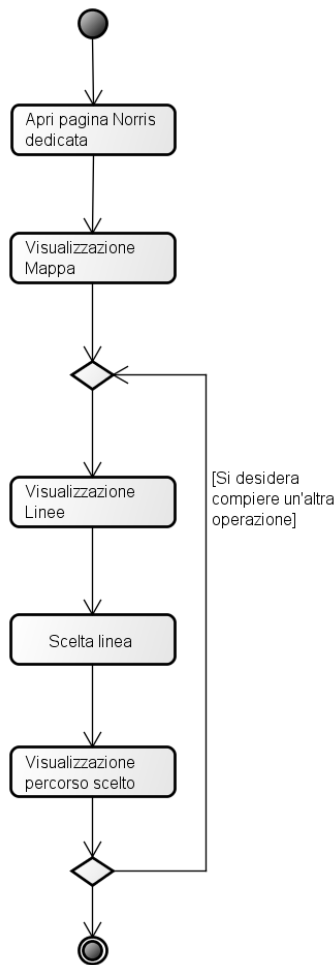


Figura 41: Diagramma di attività - Caso applicativo APS Holding

Per quanto concerne il caso applicativo, l'utente aprirà la pagina dedicata, e potrà visualizzare una mappa di Padova e un elenco di tutte le linee selezionabili. Sarà possibile selezionare le varie linee da una lista, e la mappa verrà di conseguenza aggiornata in tempo reale.

Sarà possibile ripetere l'operazione per ottenere dati di altre linee.

## 7 *Design Pattern<sub>G</sub>*

Un *design pattern<sub>G</sub>* descrive soluzioni eleganti ai molteplici problemi ricorrenti che si presentano durante l'attività di progettazione. È fondamentale per qualsiasi progettista conoscere a fondo i *design pattern<sub>G</sub>* in quanto facilita l'attività di progettazione, favorisce la riusabilità e dà benefici enormi in termini di manutenibilità. Fondamentalmente possiamo suddividere i *design pattern<sub>G</sub>* in quattro categorie:

- ***Design pattern<sub>G</sub> architetturali***: definiscono l'architettura dell'applicazione ad un alto livello di astrazione;
- ***Design pattern<sub>G</sub> creazionali***: consentono di nascondere i costruttori delle classi, permettendo di creare oggetti senza conoscere la loro implementazione;
- ***Design pattern<sub>G</sub> strutturali***: consentono di riutilizzare classi pre-esistenti, fornendo un'interfaccia adatta al riuso;
- ***Design pattern<sub>G</sub> comportamentali***: definiscono soluzioni per le interazioni tra gli oggetti.

Per una descrizione generale e più approfondita dei *design pattern<sub>G</sub>* utilizzati si veda l'appendice A - *Descrizione Design pattern<sub>G</sub>*.

Nella realizzazione del progetto Norris si è deciso di implementare i seguenti *design pattern<sub>G</sub>*.

### 7.1 *Design pattern<sub>G</sub> architetturali*

#### 7.1.1 *MVC<sub>G</sub>*

- **Scopo dell'utilizzo**: è stato scelto il pattern *MVC<sub>G</sub>* per progettare l'applicazione *Android<sub>G</sub>* permettendo così di separare la logica dalla sua rappresentazione grafica; È stato usato inoltre per progettare la parte *NorrisApp* poiché rispecchia la struttura di *AngularJS<sub>G</sub>*
- **Contesto dell'utilizzo**: il pattern *MVC<sub>G</sub>* è stato scelto per l'architettura generale dell'applicazione *Android<sub>G</sub>* e della parte *front-end<sub>G</sub>*, in quanto si adatta al meglio alle necessità progettuali di questa parte del prodotto.

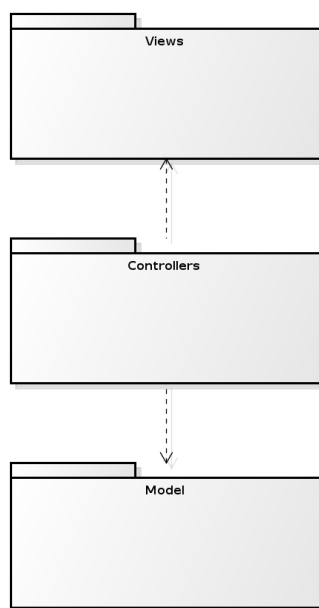


Figura 42: Un immagine del *Design Pattern<sub>G</sub>* architetturale *MVC<sub>G</sub>*

## 7.2 *Design pattern<sub>G</sub>* creazionali

### 7.2.1 *Singleton<sub>G</sub>*

- **Scopo dell'utilizzo:** assicurare che una classe abbia una sola istanza e fornire un punto d'accesso globale a tale istanza;
- **Contesto dell'utilizzo:** si è vista la necessità di utilizzarlo all'interno del *package<sub>G</sub>* *DataLayer* per quanto concerne la creazione della classe *ActiveResources* in modo tale da garantire l'unicità di istanziazione di quest'ultima. La classe si istanzierà autonomamente all'avvio della libreria e terrà traccia di tutti gli oggetti creati. *ProgressiveID*, classe del *package<sub>G</sub>* *Utils*, si comporterà in maniera analoga. Per quest'ultima si è ritenuto essenziale garantire la sua unicità in quanto essa si occuperà di generare gli ID univoci per ogni oggetto. Si rimanda alla lettura del documento *DefinizioneDiProdotto\_ver2.0.0* per una maggior comprensione dell'applicazione del *design pattern<sub>G</sub>* all'interno delle classi.

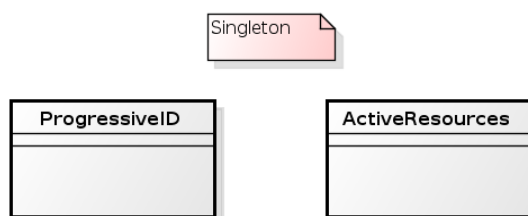


Figura 43: Le classi che implementano il *design pattern<sub>G</sub>* creazionale *Singleton<sub>G</sub>*

## 8 Stime di fattibilità e di bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente a fornire una stima di fattibilità e di bisogno di risorse, anche se è stato necessario ridefinire una parte della struttura.

L'analisi dell'architettura progettata ha permesso di constatare che le tecnologie che si è scelto di utilizzare risultano sufficientemente adeguate per la realizzazione del prodotto e per ricoprire le necessità progettuali. Sono state modificate le librerie esterne utilizzate per la creazione dei grafici di tipo *Bar Chart<sub>G</sub>* e di tipo *Line Chart<sub>G</sub>* in favore di una libreria con maggiore funzionalità, permettendo di ridurre la complessità dello sviluppo del prodotto.

In particolare, nonostante il linguaggio *JavaScript<sub>G</sub>* non presenti direttamente i costrutti utilizzati da altri linguaggi di programmazione orientati agli oggetti, sono presenti soluzioni alternative per ottenere il medesimo comportamento.

La maggior parte delle tecnologie e degli strumenti sono poco conosciute ai membri del gruppo; questi si impegneranno a colmare le proprie lacune sugli argomenti interessati tramite materiale fornito dall'amministratore ed eventuali ulteriori approfondimenti personali.

A causa di un errata stima nella progettazione per il **FlameTech Inc.** non è stato possibile implementare l'applicazione *Android<sub>G</sub>* che quindi non sarà consegnata alla revisione di accettazione come descritto nel documento *AnalisiRequisiti\_ver5.0.0*.



## 9 Tracciamento

Seguono le tabelle di tracciamento tra componenti-requisiti e tra requisiti-classi. Per semplicità di lettura, requisiti associati a figli di un componente non sono riportati anche nel padre.

### 9.1 Tracciamento componenti - requisiti

Componente	Requisiti
Norris	RAF3.5 RAV1
Norris::Lib	RAF1 RBV8.1
Norris::Lib::BusinessLayer	RAF1.2 RAF1.2.1 RAF1.2.2 RAF1.2.3 RAF2 RAF2.1 RAF3.1 RAF3.2 RAF3.3 RAF3.4 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer	RAF1.2 RAF1.2.1 RAF1.2.2 RAF1.2.3 RAF2 RAF2.1 RAF3.1 RAF3.2 RAF3.3 RAF3.4 RAF4 RAF4.1 RAF4.2

Componente	Requisiti
Norris::Lib::PresentationLayer	RAF1 RAF1.3 RAF3 RAF4.1.1 RAF4.1.10 RAF4.1.11 RAF4.1.12 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 RAF4.1.6 RAF4.1.7 RAF4.1.8 RAF4.1.9 RAF4.2.1
Norris::Lib::Utils	RAF1.1 RAV3
Norris::NorrisApp	RAF3.5 RAF3.5.1 RAV4 RAV5
Norris::NorrisApp::Controllers	RAF3.5 RAF3.5.1 RAV4 RAV5
Norris::NorrisApp::Model	RAF3.5 RAF3.5.1 RAV4 RAV5
Norris::NorrisApp::Services	RAF3.5 RAF3.5.1 RAV4 RAV5

Componente	Requisiti
Norris::NorrisApp::Views	RAF3.5 RAF3.5.1 RAV4 RAV5
AndroidApp	RAF3.5 RCF3.5.2 RAV9
AndroidApp::Activities	RCF3.5.2.1 RCF3.5.2.1.1 RCF3.5.2.1.2 RCF3.5.2.4.1 RCF3.5.2.4.2
AndroidApp::AppModel	RCF3.5.2.3
AndroidApp::Layouts	RCF3.5.2.2 RCF3.5.2.4

Tabella 3: Tabella Tracciamento Componenti - Requisiti

## 9.2 Tracciamento requisiti - componenti

Requisito	Componenti
RAF1	Norris::Lib Norris::Lib::PresentationLayer
RAF1.1	Norris::Lib::Utils
RAF1.2	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF1.2.1	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF1.2.2	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF1.2.3	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF1.3	Norris::Lib::PresentationLayer
RAF2	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF2.1	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF3	Norris::Lib::PresentationLayer
RAF3.1	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF3.2	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF3.3	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF3.4	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF3.5	AndroidApp Norris Norris::NorrisApp Norris::NorrisApp::Controllers Norris::NorrisApp::Model Norris::NorrisApp::Services Norris::NorrisApp::Views

Requisito	Componenti
RAF3.5.1	Norris::NorrisApp Norris::NorrisApp::Controllers Norris::NorrisApp::Model Norris::NorrisApp::Services Norris::NorrisApp::Views
RCF3.5.2	AndroidApp
RCF3.5.2.1	AndroidApp::Activities
RCF3.5.2.1.1	AndroidApp::Activities
RCF3.5.2.1.2	AndroidApp::Activities
RCF3.5.2.2	AndroidApp::Layouts
RCF3.5.2.3	AndroidApp::AppModel
RCF3.5.2.4	AndroidApp::Layouts
RCF3.5.2.4.1	AndroidApp::Activities
RCF3.5.2.4.2	AndroidApp::Activities
RAF4	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF4.1	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF4.1.1	Norris::Lib::PresentationLayer
RAF4.1.2	Norris::Lib::PresentationLayer
RAF4.1.3	Norris::Lib::PresentationLayer
RAF4.1.4	Norris::Lib::PresentationLayer
RAF4.1.5	Norris::Lib::PresentationLayer
RAF4.1.6	Norris::Lib::PresentationLayer
RAF4.1.7	Norris::Lib::PresentationLayer
RAF4.1.8	Norris::Lib::PresentationLayer
RAF4.1.9	Norris::Lib::PresentationLayer
RAF4.1.10	Norris::Lib::PresentationLayer

Requisito	Componenti
RAF4.1.11	Norris::Lib::PresentationLayer
RAF4.1.12	Norris::Lib::PresentationLayer
RAF4.1.13	Norris::Lib::PresentationLayer
RAF4.1.14	Norris::Lib::PresentationLayer
RAF4.1.15	Norris::Lib::PresentationLayer
RAF4.1.16	Norris::Lib::PresentationLayer
RAF4.1.17	Norris::Lib::PresentationLayer
RAF4.2	Norris::Lib::BusinessLayer Norris::Lib::DataLayer
RAF4.2.1	Norris::Lib::PresentationLayer
RAV1	Norris
RAV3	Norris::Lib::Utils
RAV4	Norris::NorrisApp Norris::NorrisApp::Controllers Norris::NorrisApp::Model Norris::NorrisApp::Services Norris::NorrisApp::Views
RAV5	Norris::NorrisApp Norris::NorrisApp::Controllers Norris::NorrisApp::Model Norris::NorrisApp::Services Norris::NorrisApp::Views
RBV8.1	Norris::Lib
RAV9	AndroidApp

Tabella 4: Tabella Tracciamento Requisiti - Componenti

## A Descrizione *Design pattern<sub>G</sub>*

### A.1 *Design pattern<sub>G</sub>* architetturali

#### A.1.1 *MVCS<sub>G</sub>*

- **Descrizione:** *MVCS<sub>G</sub>* non è un *design pattern<sub>G</sub>* ma si tratta di una variante al *MVC<sub>G</sub>* che espande quest'ultimo con l'aggiunta di un *package<sub>G</sub> Services*. *AngularJS<sub>G</sub>*, infatti, prevede la possibilità di espandere il concetto di *MVC<sub>G</sub>* con la logica del *MVW<sub>G</sub>* ossia: Model, View e Whatever works for you.
- **Motivazione:** è il *design pattern<sub>G</sub>* più adatto per lavorare con *AngularJS<sub>G</sub>* e permette di separare la logica dell'applicazione dalla sua rappresentazione grafica.
- **Ambito applicativo:** questa struttura viene spesso usato quando si lavora con *AngularJS<sub>G</sub>*.

#### A.1.2 *MVC<sub>G</sub>*

*Model-View-Controller* (MVC) è un pattern per l'implementazione di interfacce utente. Esso divide un'applicazione software in tre parti interconnesse, in modo da separare nettamente la rappresentazione interna dei dati dal modo in cui essa viene presentata all'utente. Il componente centrale, il modello, consiste di dati business, regole, logica e funzioni. Una *view* può essere qualsiasi output dell'informazione, come ad esempio un testo o un diagramma. Si possono avere molteplici *view* della stessa informazione. La terza parte, il *controller*, si occupa di accettare degli input e di convertirli in *comandi* per il *model* o per la *view*.

Oltre a dividere l'applicazione in queste tre componenti, *MVC<sub>G</sub>* si occupa anche di definire le interazioni tra esse:

- Un *controller* può inviare comandi al *model* per aggiornare il suo stato. Può inoltre inviare comandi alla sua *view* associata, in modo da cambiarne la presentazione;
- Un *model* quando cambia il suo stato interno notifica le sue *view* e i suoi *controller* associati. Questo permette alle *view* di cambiare la loro presentazione e ai *controller* di cambiare il loro insieme di comandi disponibili.
- Una *view* viene aggiornata dal *controller* sui dati che necessita per generare un output per l'utente.

Sebbene inizialmente sviluppata per applicazioni desktop, *MVC<sub>G</sub>* è stato usato moltissimo come architettura per le Web application in tutti i principali linguaggi di programmazione. Moltissimi *framework<sub>G</sub>* commerciali e non sono stati progettati utilizzando questo pattern.

### A.2 *Design pattern<sub>G</sub>* creazionali

#### A.2.1 *Singleton<sub>G</sub>*

- **Descrizione:** il *design pattern<sub>G</sub> Singleton<sub>G</sub>* assicura che una classe abbia un'unica istanza e fornisce un punto d'accesso globale a tale istanza;

- **Motivazione:** garantire un risparmio di risorse fisiche poiché la creazione di un'istanza dell'oggetto in questione è rimandata fino al momento del suo effettivo utilizzo;
- **Ambito applicativo:** tale *design pattern<sub>G</sub>* viene utilizzato nei seguenti casi:
  1. si desidera avere la garanzia che nel sistema vi sia una sola istanza di oggetti di un determinato tipo, e che tali oggetti siano accessibili da un punto di accesso ben preciso;
  2. si desidera che il controllo di unicità di un oggetto non venga delegato ad altre entità;
  3. si desidera che più oggetti condividano un unico pool di dati.