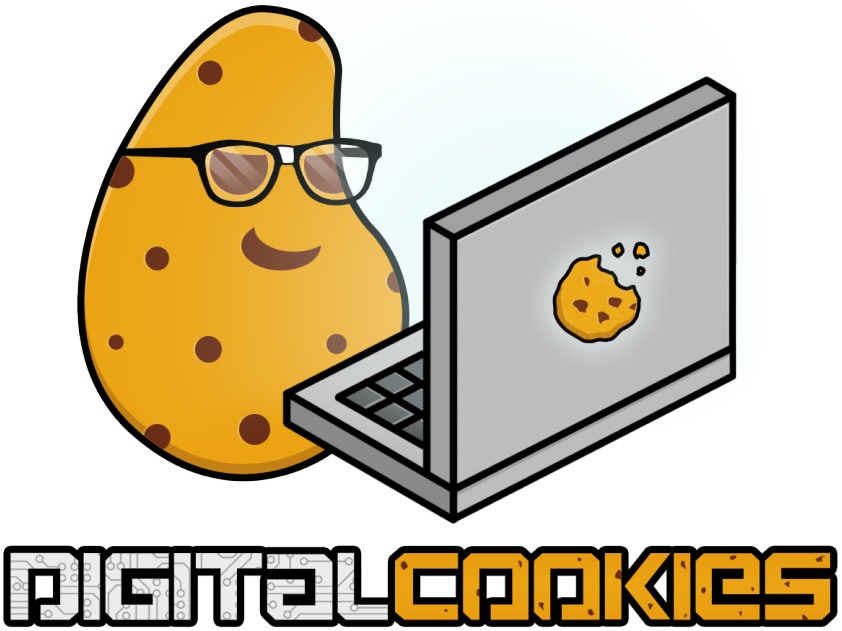

Definizione di Prodotto

Gruppo DigitalCookies — Progetto SWEDesigner

digitalcookies.group@gmail.com

Informazioni sul documento

Versione	2.0.0
Redazione	Alessia Bragagnolo, Christian Cabrera, Alberto Rossetti, Davide Albertini, Saverio Follador, Carlo Sindico
Verifica	Alessia Bragagnolo
Approvazione	Christian Cabrera
Uso	Interno
Distribuzione	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo DigitalCookies

Descrizione

Questo documento descrive la progettazione di dettaglio definita dal gruppo DigitalCookies relativa al progetto SWEDesigner.

Registro delle modifiche

Versione	Data	Collaboratori	Ruolo	Descrizione
2.0.0	03-07-2017	Christian Cabrera	Responsabile	Approvazione del documento
1.1.0	03-07-2017	Alessia Bragagnolo	Verificatore	Verifica del documento
1.0.4	30-06-2017	Carlo Sindico	Progettista	Informazioni aggiuntive sui metodi `retrieveClassesList` e `retrievePatternList` in 3.1.1.2.7
1.0.3	30-06-2017	Alberto Rossetti	Progettista	Specifica dettagliata di tutti i tipi JSON presenti nel documento
1.0.2	29-06-2017	Carlo Sindico	Progettista	Inserito diagramma delle classi in sezione 3.1
1.0.1	29-06-2017	Alberto Rossetti	Progettista	Modificata descrizione di `Function` nella sezione 4.1.2.2.2
1.0.0	16-06-2017	Davide Albertini	Responsabile	Approvazione del documento
0.4.0	16-06-2017	Alessia Bragagnolo	Verificatore	Verifica del documento
0.3.2	15-06-2017	Alberto Rossetti	Progettista	Incremento nella sezione 3.1.5.2.4 Relation
0.3.1	14-06-2017	Carlo Sindico	Progettista	Incremento nella sezione 4.1.2.2.1 MiddlewareLoader
0.3.0	26-05-2017	Alessia Bragagnolo	Verificatore	Verifica del documento

0.2.0	25-05-2017	Alberto Giudice	Verificatore	Verifica delle sezioni introduttive e Front-End
0.1.3	24-05-2017	Saverio Follador	Progettista	Stesura sezioni relative ai tracciamenti
0.1.2	22-05-2017	Davide Albertini	Progettista	Incremento nella sezione 4.1.7.2.5 RelationJavaGenerator
0.1.1	19-05-2017	Alessia Bragagnolo	Progettista	Correzioni diagrammi di sequenza
0.1.0	08-05-2017	Davide Albertini	Verificatore	Verifica del documento
0.0.9	05-05-2017	Carlo Sindico	Progettista	Stesura sezione relativa al Data Tier del Back-End
0.0.8	05-05-2017	Alberto Rossetti	Progettista	Stesura sezione relativa al ViewModel del Front-End
0.0.7	04-05-2017	Alessia Bragagnolo	Progettista	Stesura sezione relativa all'Application Tier del Back-End
0.0.6	04-05-2017	Christian Cabrera	Progettista	Stesura sezione relativa al Model del Front-End
0.0.5	03-05-2017	Carlo Sindico	Progettista	Stesura sezione relativa al Presentation Tier del Back-End
0.0.4	03-05-2017	Alberto Rossetti	Progettista	Stesura sezione relativa alla View del Front-End
0.0.3	02-05-2017	Alessia Bragagnolo	Progettista	Stesura sezione Standard di progetto

0.0.2	02-05-2017	Carlo Sindico	Progettista	Stesura sezione introduzione
0.0.1	01-05-2017	Alberto Rossetti	Progettista	Creazione del template

Indice

1 Introduzione **10**
1.1 Scopo del documento . . . 10
1.2 Scopo del prodotto . . . 10
1.3 Ambiguità . . . 10
1.4 Riferimenti . . . 10
1.4.1 Normativi . . . 10
1.4.2 Informativi . . . 11

2 Standard di progetto **12**
2.1 Standard di progettazione architetturale . . . 12
2.2 Standard di documentazione del codice . . . 12
2.3 Standard di denominazione di entità e relazioni . . . 12
2.4 Standard di programmazione . . . 12
2.5 Standard di lavoro . . . 12

3 Front-end **13**
3.1 Descrizione packages e classi . . . 13
3.1.1 Front-end::View . . . 13
3.1.1.1 Informazioni sul package . . . 13
3.1.1.2 Classi . . . 15
3.1.1.2.1 SinglePageApp . . . 15
3.1.1.2.2 CodeEditor . . . 17
3.1.1.2.3 DiagramEditor . . . 18
3.1.1.2.4 ClassDiagramEditor . . . 20
3.1.1.2.5 ActivityDiagramEditor . . . 22
3.1.1.2.6 DiagramPalette . . . 24
3.1.1.2.7 ClassDiagramPalette . . . 25
3.1.1.2.8 ActivityDiagramPalette . . . 27
3.1.1.2.9 DiagramFactory . . . 29
3.1.1.2.10 ClassFactory . . . 30
3.1.1.2.11 ActivityFactory . . . 32
3.1.2 Front-end::ViewModel . . . 33
3.1.2.1 Informazioni sul package . . . 33
3.1.2.2 Classi . . . 34
3.1.2.2.1 EditorObjController . . . 34
3.1.2.2.2 PaletteObjController . . . 36
3.1.2.2.3 CodeCommandController . . . 38
3.1.2.2.4 PaletteCommandController . . . 39

3.1.3 Front-end::Model . . . 40
3.1.3.1 Informazioni sul package . . . 40
3.1.4 Front-end::Model::Objects . . . 41
3.1.4.1 Informazioni sul package . . . 41
3.1.4.2 Classi . . . 42
3.1.4.2.1 BaseDiaObj . . . 42
3.1.5 Front-end::Model::Objects::ClassObjects . . . 43
3.1.5.1 Informazioni sul package . . . 43
3.1.5.2 Classi . . . 44
3.1.5.2.1 ClassDiaObj . . . 44
3.1.5.2.2 Class . . . 45
3.1.5.2.3 Comment . . . 50
3.1.5.2.4 Relation . . . 51
3.1.5.2.5 Dependency . . . 53
3.1.5.2.6 Association . . . 54
3.1.5.2.7 Aggregation . . . 56
3.1.5.2.8 Composition . . . 57
3.1.5.2.9 Generalization . . . 59
3.1.6 Front-end::Model::Objects::ActivityObjects . . . 60
3.1.6.1 Informazioni sul package . . . 60
3.1.6.2 Classi . . . 60
3.1.6.2.1 ActivityDiaObj . . . 60
3.1.6.2.2 VariableObj . . . 63
3.1.6.2.3 ExistingVariableObj . . . 64
3.1.6.2.4 MethodCallObj . . . 65
3.1.6.2.5 CycleObj . . . 67
3.1.6.2.6 IfElseObj . . . 68
3.1.6.2.7 OperatorObj . . . 69
3.1.6.2.8 StepObj . . . 71
3.1.6.2.9 JollyObj . . . 73
3.1.7 Front-end::Model::Commands . . . 74
3.1.7.1 Informazioni sul package . . . 74
3.1.7.2 Classi . . . 74
3.1.7.2.1 Command . . . 74
3.1.7.2.2 RequestCode . . . 76
3.1.7.2.3 GetTemplate . . . 77
3.1.8 Front-end::Model::Services . . . 79
3.1.8.1 Informazioni sul package . . . 79
3.1.8.2 Classi . . . 79
3.1.8.2.1 ServerConnector . . . 79

4 Back-end **81**
4.1 Descrizione packages e classi . . . 81

4.1.1 Back-end::PresentationTier . . . 81
4.1.1.1 Informazioni sul package . . . 81
4.1.2 Back-end::PresentationTier::Middleware . . . 82
4.1.2.1 Informazioni sul package . . . 82
4.1.2.2 Classi . . . 83
4.1.2.2.1 MiddlewareLoader . . . 83
4.1.2.2.2 ErrorHandler . . . 84
4.1.2.2.3 Router . . . 85
4.1.2.2.4 NotFoundHandler . . . 87
4.1.3 Back-end::PresentationTier::Controller . . . 88
4.1.3.1 Informazioni sul package . . . 88
4.1.3.2 Classi . . . 89
4.1.3.2.1 IndexGiver . . . 89
4.1.4 Back-end::ApplicationTier . . . 90
4.1.4.1 Informazioni sul package . . . 90
4.1.4.2 Classi . . . 90
4.1.4.2.1 ApplicationController . . . 90
4.1.5 Back-end::ApplicationTier::Generator . . . 92
4.1.5.1 Informazioni sul package . . . 92
4.1.5.2 Classi . . . 93
4.1.5.2.1 BaseGenerator . . . 93
4.1.6 Back-end::ApplicationTier::Generator::JavaGenerator . . . 94
4.1.6.1 Informazioni sul package . . . 94
4.1.6.2 Classi . . . 95
4.1.6.2.1 JavaGenerator . . . 95
4.1.7 Back-end::ApplicationTier::Generator::JavaGenerator ::ClassDiaJavaGenerator . . . 96
4.1.7.1 Informazioni sul package . . . 96
4.1.7.2 Classi . . . 97
4.1.7.2.1 ClassDiaJavaGenerator . . . 97
4.1.7.2.2 ClassJavaGenerator . . . 99
4.1.7.2.3 AttributeJavaGenerator . . . 101
4.1.7.2.4 MethodJavaGenerator . . . 102
4.1.7.2.5 RelationJavaGenerator . . . 104
4.1.7.2.6 CommentJavaGenerator . . . 106
4.1.8 Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator . . . 107
4.1.8.1 Informazioni sul package . . . 107
4.1.8.2 Classi . . . 108
4.1.8.2.1 ActivityDiaJavaGenerator . . . 108
4.1.8.2.2 InstructionJavaGenerator . . . 109
4.1.8.2.3 VariableJavaGenerator . . . 113
4.1.8.2.4 VariableConnectJavaGenerator . . . 114

4.1.8.2.5 MethodCallJavaGenerator . . . 115
4.1.8.2.6 JollyJavaGenerator . . . 117
4.1.8.2.7 StepJavaGenerator . . . 118
4.1.8.2.8 CycleJavaGenerator . . . 119
4.1.8.2.9 OperatorJavaGenerator . . . 121
4.1.8.2.10 IfElseJavaGenerator . . . 124
4.1.9 Back-end::ApplicationTier::Error . . . 126
4.1.9.1 Informazioni sul package . . . 126
4.1.9.2 Classi . . . 126
4.1.9.2.1 ErrorApplication . . . 126
4.1.10 Back-end::DataTier . . . 128
4.1.10.1 Informazioni sul package . . . 128
4.1.10.2 Classi . . . 129
4.1.10.2.1 Template . . . 129

5 Diagrammi di sequenza 131
5.1 Generazione Codice . . . 131
5.2 Generazione Codice Classe . . . 132
5.3 Generazione Codice Metodo . . . 133
5.4 Ottenimento template . . . 135
5.5 Richiesta Generazione Codice . . . 136

6 Tracciamento 137
6.1 Requisiti-Classi . . . 137
6.2 Classi-Requisiti . . . 143

Elenco delle figure

1 Diagramma dei package Front-end . . . 13
2 Front-end::View::SinglePageApp . . . 15
3 Front-end::View::CodeEditor . . . 17
4 Front-end::View::DiagramEditor . . . 18
5 Front-end::View::ClassDiagramEditor . . . 20
6 Front-end::View::ActivityDiagramEditor . . . 22
7 Front-end::View::DiagramPalette . . . 24
8 Front-end::View::ClassDiagramPalette . . . 25
9 Front-end::View::ActivityDiagramPalette . . . 27
10 Front-end::View::DiagramFactory . . . 29
11 Front-end::View::ClassFactory . . . 30
12 Front-end::View::ActivityFactory . . . 32
13 Front-end::ViewModel::EditorObjController . . . 34
14 Front-end::ViewModel::PaletteObjController . . . 36
15 Front-end::ViewModel::CodeCommandController . . . 38

16 Front-end::ViewModel::PaletteCommandController 39
17 Front-end::Model::Objects::BaseDiaObj 42
18 Front-end::Model::Objects::ClassObjects::ClassDiaObj 44
19 Front-end::Model::Objects::ClassObjects::Class 45
20 Front-end::Model::Objects::ClassObjects::Comment 50
21 Front-end::Model::Objects::ClassObjects::Relation 51
22 Front-end::Model::Objects::ClassObjects::Dependency 53
23 Front-end::Model::Objects::ClassObjects::Association 54
24 Front-end::Model::Objects::ClassObjects::Aggregation 56
25 Front-end::Model::Objects::ClassObjects::Composition 57
26 Front-end::Model::Objects::ClassObjects::Genralization 59
27 Front-end::Model::Objects::ActivityObjects::ActivityDiaObj 60
28 Front-end::Model::Objects::ActivityObjects::VariableObj 63
29 Front-end::Model::Objects::ActivityObjects::ExistingVariableObj 64
30 Front-end::Model::Objects::ActivityObjects::MethodCallObj 65
31 Front-end::Model::Objects::ActivityObjects::CycleObj 67
32 Front-end::Model::Objects::ActivityObjects::IfElseObj 68
33 Front-end::Model::Objects::ActivityObjects::OperatorObj 69
34 Front-end::Model::Objects::ActivityObjects::StepObj 71
35 Front-end::Model::Objects::ActivityObjects::JollyObj 73
36 Front-end::Model::Commands::Command 74
37 Front-end::Model::Commands::RequestCode 76
38 Front-end::Model::Commands::GetTemplate 77
39 Front-end::Model::Services::ServerConnector 79
40 Diagramma dei package Back-end 81
41 Back-end::PresentationTier::Middleware::MiddlewareLoader 83
42 Back-end::PresentationTier::Middleware::ErrorHandler 84
43 Back-end::PresentationTier::Middleware::Router 85
44 Back-end::PresentationTier::Middleware::NotFoundHandler 87
45 Back-end::PresentationTier::Controller::IndexGiver 89
46 Back-end::ApplicationTier::ApplicationController 90
47 Back-end::ApplicationTier::Generator::BaseGenerator 93
48 Back-end::ApplicationTier::Generator::JavaGenerator::JavaGenerator . . . 95
49 Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::ClassDiaJavaGenerator 97
50 Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::ClassJavaGenerator 99
51 Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::AttributeJavaGenerator 101
52 Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::MethodJavaGenerator 102
53 Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::RelationJavaGenerator 104

54 Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::CommentJavaGenerator 106
55 Back-end::ApplicationTier::Generator::JavaGenerator::
ActivityDiaJavaGenerator::ActivityDiaJavaGenerator 108
56 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::InstructionJavaGenerator 110
57 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::VariableJavaGenerator 113
58 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::VariableConnectJavaGenerator 114
59 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::MethodCallJavaGenerator 115
60 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::JollyJavaGenerator 117
61 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::StepJavaGenerator 118
62 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::CycleJavaGenerator 119
63 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::OperatorJavaGenerator 121
64 Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::IfElseJavaGenerator 124
65 Back-end::ApplicationTier::Error::ErrorApplication 127
66 Back-end::DataTier::Template 129
67 Diagramma di Sequenza Generazione Codice 131
68 Diagramma di Sequenza Generazione Codice Classe 132
69 Diagramma di Sequenza Generazione Codice Metodo 133
70 Diagramma di Sequenza Ottenimento template 135
71 Diagramma di Sequenza Richiesta Generazione Codice 136

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire in dettaglio la struttura e le relazioni tra le componenti di SWEDesigner, riprendendo e approfondendo quanto già descritto nel documento *Specifica Tecnica v3.0.0*.
Tale documento servirà da guida per i Programmatori, fornendo loro le direttive per l'implementazione del sistema e l'attività di codifica.

1.2 Scopo del prodotto

Lo scopo del *prodotto*$_G$ è creare un software di costruzione di diagrammi *UML*$_G$ con relativa generazione di codice *Java*$_G$. Il codice potrà essere generato dall'utente a partire dai diagrammi UML delle *classi*$_G$ e da una versione modificata del diagramma delle *attività*$_G$.

L'utente, interagendo con il sistema, sarà in grado di:

- delineare la struttura delle classi utilizzando lo standard UML;
- definire il corpo dei metodi delle classi sfruttando una versione modificata del diagramma delle attività;
- generare un applicativo scritto in codice Java a partire dai diagrammi sopracitati.

L'utente potrà inoltre sfruttare la *libreria*$_G$ fornita con il prodotto per generare con facilità diagrammi relativi al dominio dei giochi di carte.

L'*editor*$_G$ sarà fruibile dall'utente attraverso un *browser*$_G$ desktop idoneo all'utilizzo delle tecnologie *HTML5*$_G$, *CSS3*$_G$ e *JavaScript*$_G$.

1.3 Ambiguità

Al fine di evitare ogni ambiguità relativa al linguaggio impiegato nei documenti viene fornito il *Glossario v4.0.0*, contenente la definizione dei termini in corsivo marcati con una G pedice.

1.4 Riferimenti

1.4.1 Normativi

- ***Norme di Progetto v4.0.0***;

- ***Analisi dei Requisiti v3.0.0***.
- **Capitolato d'appalto C6: SWEDesigner editor di diagrammi UML con generazione di codice:** http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6p.pdf (ultima consultazione effettuata in data 24-04-2017).
- **Verbale di incontro interno** con i componenti del gruppo del 21-04-2017;
- **Verbale di incontro interno** con i componenti del gruppo del 27-04-2017;
- **Verbale di incontro esterno** con i componenti del gruppo e il proponente Zucchetti S.p.A. del 04-05-2017;
- **Verbale di incontro esterno** con i componenti del gruppo e il proponente Zucchetti S.p.A. del 07-06-2017;
- **Verbale di incontro interno** con i componenti del gruppo del 09-06-2017.

1.4.2 Informativi

- **Design Patterns** - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - 1a edizione italiana (2006) (sezioni §3, §4);
- **Martin Fowler - UML Distilled** - 2nd edition (sezioni §4, §5);
- **Interfaccia REST**: https://it.wikipedia.org/wiki/Representational_State_Transfer (ultima consultazione effettuata in data 24-05-2017);
- ***GoJS***$_G$: Interactive JavaScript Diagrams in HTML, OpenSource https://gojs.net/latest/index.html (ultima consultazione effettuata in data 22-05-2017);
- **Express**: http://expressjs.com/it/guide/using-middleware.html (ultima consultazione avvenuta in data 19-05-2017)
- **Angular2**: JavaScript framework http://www.angular2.com/ (ultima consultazione effettuata in data 20-05-2017);
- **Node.js**: https://nodejs.org/en/docs/guides/ (ultima consultazione effettuata in data 20-05-2017);
- **Mongodb**: https://docs.mongodb.com/manual/introduction/ (ultima consultazione effettuata in data 21-05-2017);
- **Mongoose**: http://mongoosejs.com/docs/api.html (ultima consultazione effettuata in data 23-05-2017);

2 Standard di progetto

2.1 Standard di progettazione architetturale

Gli standard di progettazione architetturale seguiti sono definiti nel documento *Specifica Tecnica v3.0.0*. Si faccia riferimento ad esso per approfondimenti.

2.2 Standard di documentazione del codice

Gli standard di documentazione del codice sono definiti e descritti nel documento *Norme di Progetto v4.0.0*. Si faccia riferimento ad esso per approfondimenti.

2.3 Standard di denominazione di entità e relazioni

Tutti gli elementi che saranno definiti nel seguente documento, siano essi $package_G$, $classi_G$, metodi o attributi, devono avere una denominazione chiara e concisa. La chiarezza del nome sarà maggiormente importante rispetto alla sua lunghezza, che potrà venire appositamente abbreviata. Sono ammesse abbreviazioni qualora:

- non si altera l'immediata complessità del nome;
- non si incorre nel rischio di essere ambigui.

Per tutte le regole tipografiche adottate si faccia riferimento al documento *Norme di Progetto v4.0.0*.

2.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nel documento *Norme di Progetto v4.0.0*. Si rimanda ad esso per le regole da seguire durante la codifica.

2.5 Standard di lavoro

Tutti gli strumenti di lavoro e le procedure da seguire per la corretta realizzazione del prodotto sono definiti nel documento *Norme di Progetto v4.0.0*. Si faccia riferimento ad esso per approfondimenti.

3 Front-end

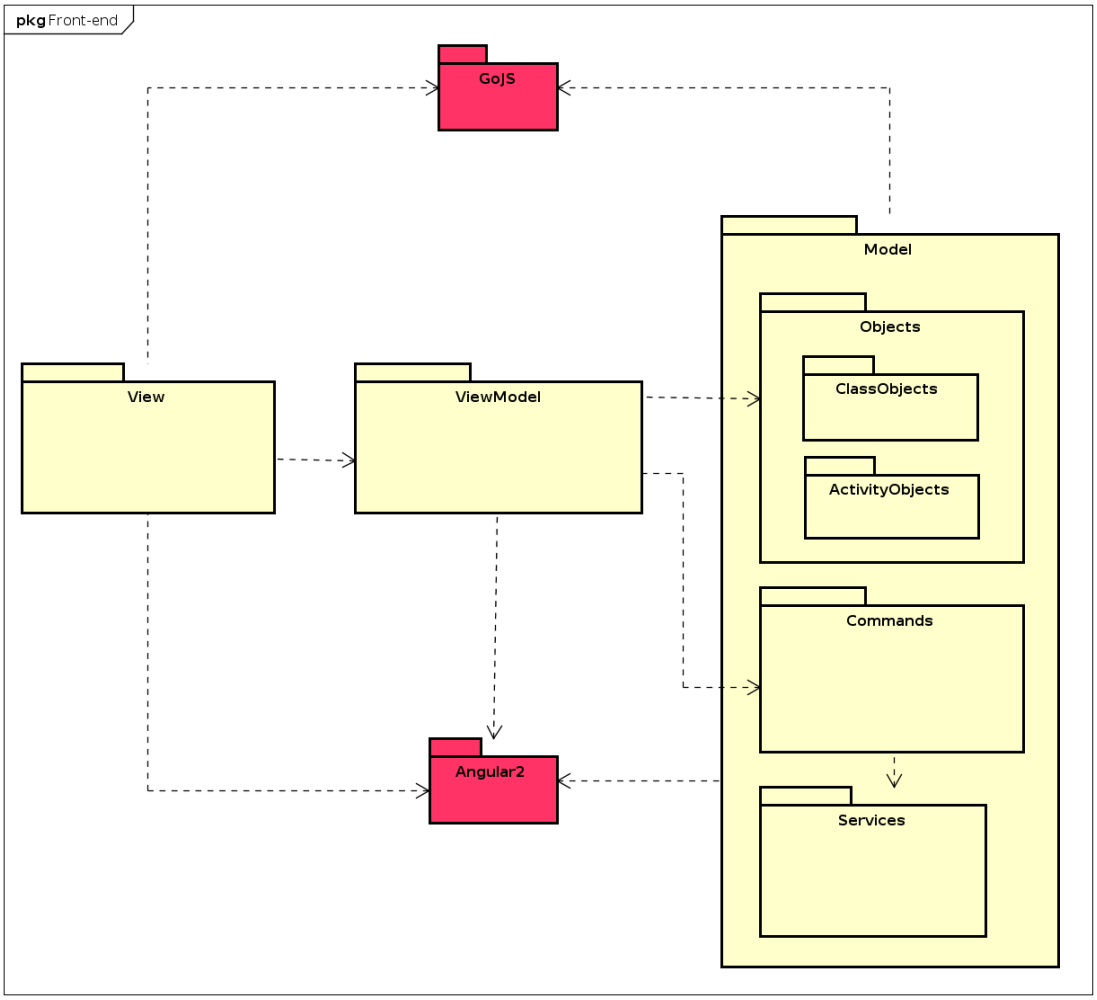

Figura 1: Diagramma dei package Front-end

3.1 Descrizione packages e classi

3.1.1 Front-end::View

3.1.1.1 Informazioni sul package

- **Descrizione**:
  Questo package raccoglie le classi che rappresentano l'editor dei diagrammi delle classi e delle attività, l'editor che permette di modificare il codice prodotto dai

diagrammi precedentemente creati. Il package contiene inoltre le palette e i menù laterali che fanno parte della $classe_G$ dell'editor dei diagrammi;

- **Framework esterni**:
  - Angular2
  - GoJS

3.1.1.2 Classi

3.1.1.2.1 SinglePageApp

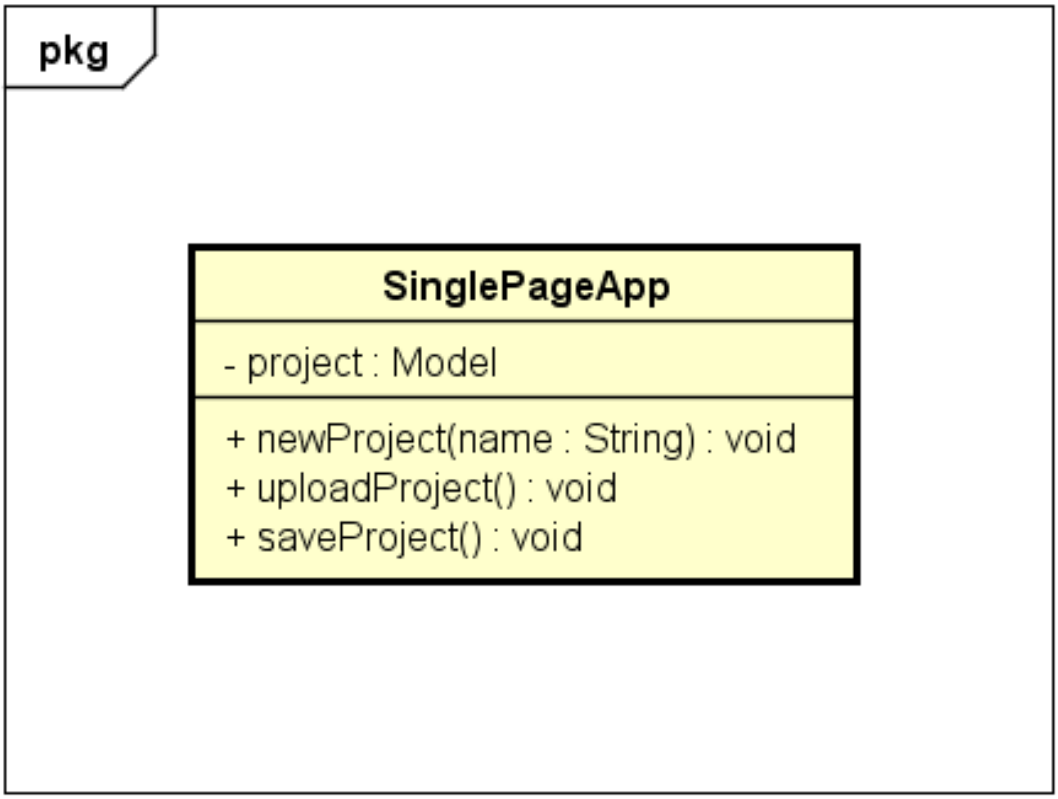

Figura 2: Front-end::View::SinglePageApp

- **Descrizione**:
  Questa classe gestisce la visualizzazione dell'*interfaccia*$_G$ grafica di SWEDesigner. Contiene le classi riguardanti l'editor dei diagrammi e l'editor del codice generato. Rappresenta il *Client*$_G$ del *design pattern*$_G$ *Command*$_G$ implementato in `Front-end::Model::Commands`.
- **Utilizzo**:
  Viene utilizzata per generare e gestire l'interfaccia grafica della web-app. Importa l'interfaccia `Front-end::View::DiagramFactory` per la generazione dell'interfaccia grafica dei diagrammi. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:

  - OUT: `Front-end::View::CodeEditor`: necessaria perché la single page app deve visualizzare l'editor del codice generato dal `Back-end`;
  - OUT: `Front-end::View::DiagzramEditor`: necessaria perché la single page app deve visualizzare due tipi di diagrammi quello delle classi e quello delle attività;
  - OUT: `Front-end::View::DiagramPalette`: necessaria perché la single page app deve visualizzare la palette per ciascun diagramma;
  - OUT: `Front-end::View::DiagramFactory`: necessaria perché rappresenta la struttura per le operazioni di creazione dei diagrammi.

- **Attributi**:
  - `- project: Model`
    contiene le informazioni sul progetto caricato e i suoi diagrammi.
- **Metodi**:
  - `+ newProject(name: String): void`
    si occupa di creare un nuovo progetto.
    Parametri:
    * `name: String`
      rappresenta il nome da assegnare al nuovo progetto.
  - `+ uploadProject(): void`
    si occupa di gestire il caricamento di un progetto precedentemente salvato dall'utente.
  - `+ saveProject(): void`
    si occupa di salvare il progetto corrente.

3.1.1.2.2 CodeEditor

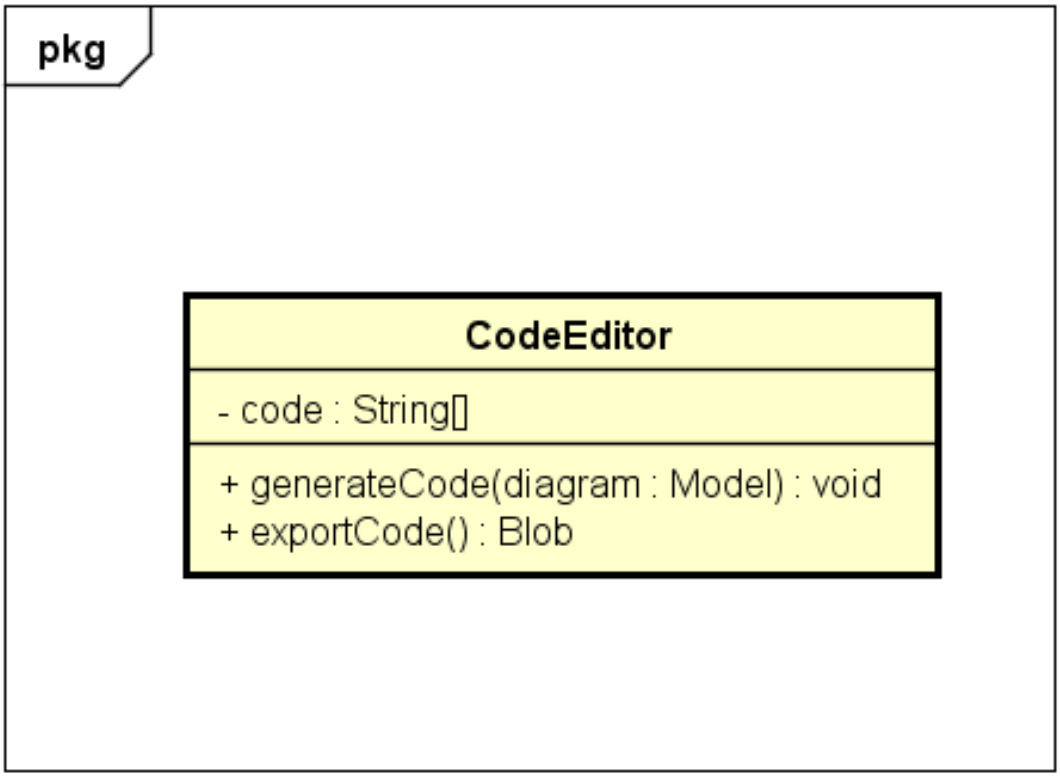

Figura 3: Front-end::View::CodeEditor

- **Descrizione**:
  Questa classe gestisce la visualizzazione dell'interfaccia grafica dell'editor del codice generato dal `Back-end`.
- **Utilizzo**:
  Viene utilizzata per generare e gestire l'interfaccia grafica dell'editor del codice generato.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::SinglePageApp`: utilizza `Front-end::View::CodeEditor` perché deve visualizzare l'editor del codice generato dal `Back-end`;
  - OUT: `Front-end::ViewModel::CodeCommandController`: necessaria perché deve comunicare con il `Front-end::Model` per processare le richieste provenienti dal `Front-end::View::CodeEditor`.

- **Attributi**:
  - `- code: String[]`
    contiene un insieme di stringhe rappresentanti i vari $file_G$ di codice generato.
- **Metodi**:
  - `+ generateCode(diagram: Model): void`
    si occupa di generare il codice partendo dal diagramma creato dall'utente.
    Parametri:
    * `diagram: Model`
      rappresenta il diagramma creato dall'utente.
  - `+ exportCode(): Blob`
    si occupa di gestire l'esportazione del codice generato (e eventualmente modificato) dall'utente.

3.1.1.2.3 DiagramEditor

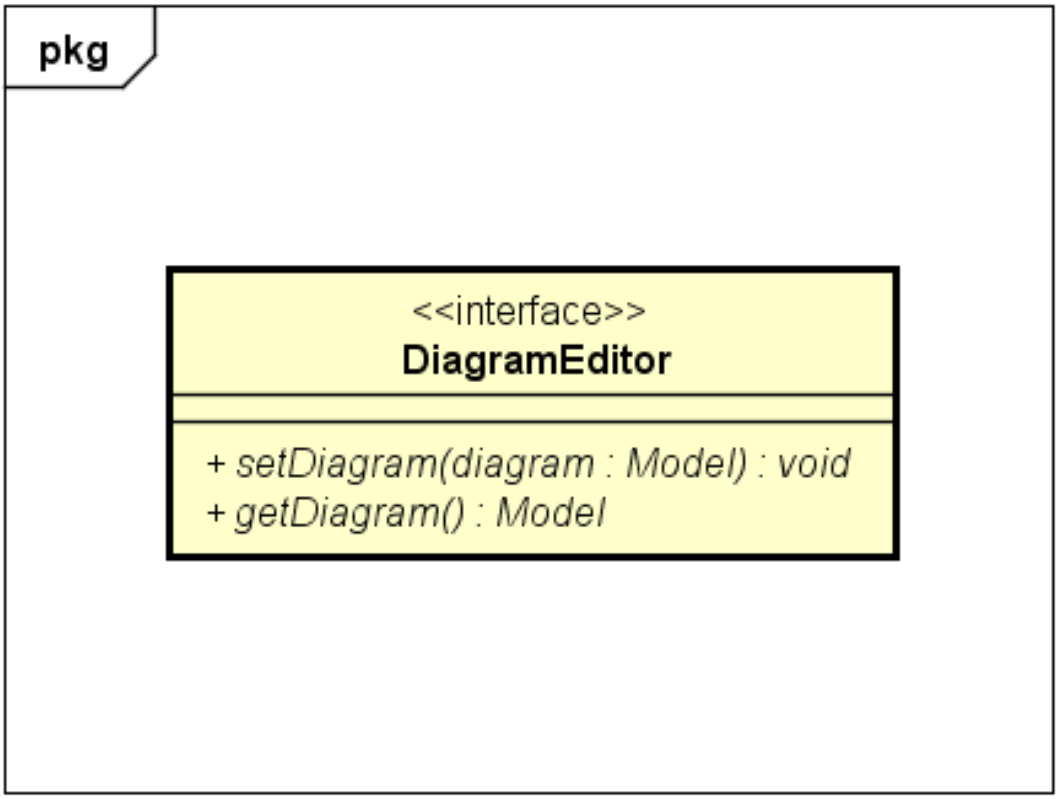

Figura 4: Front-end::View::DiagramEditor

- **Descrizione**:
  Questa interfaccia rappresenta la struttura per ciascun editor di diagramma. Rappresenta l'interfaccia AbstractProduct del design pattern *Abstract Factory*$_G$.
- **Utilizzo**:
  Viene utilizzata per fornire un'interfaccia grafica comune agli editor dei diagrammi. È contenuta in `Front-end::View::SinglePageApp`. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::SinglePageApp`: utilizza `Front-end::View::DiagramEditor` perché deve visualizzare la struttura per ciascun editor del diagramma.
- **Sottoclassi**:
  - `Front-end::View::ClassDiagramEditor`
  - `Front-end::View::ActivityDiagramEditor`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ setDiagram(diagram: Model): void`
    si occupa di settare il diagramma rappresentato dall'editor.
    Parametri:
    * `diagram: Model`
      contiene il diagramma da settare.
  - `+ getDiagram(): Model`
    si occupa di ritornare il diagramma rappresentato dall'editor.

3.1.1.2.4 ClassDiagramEditor

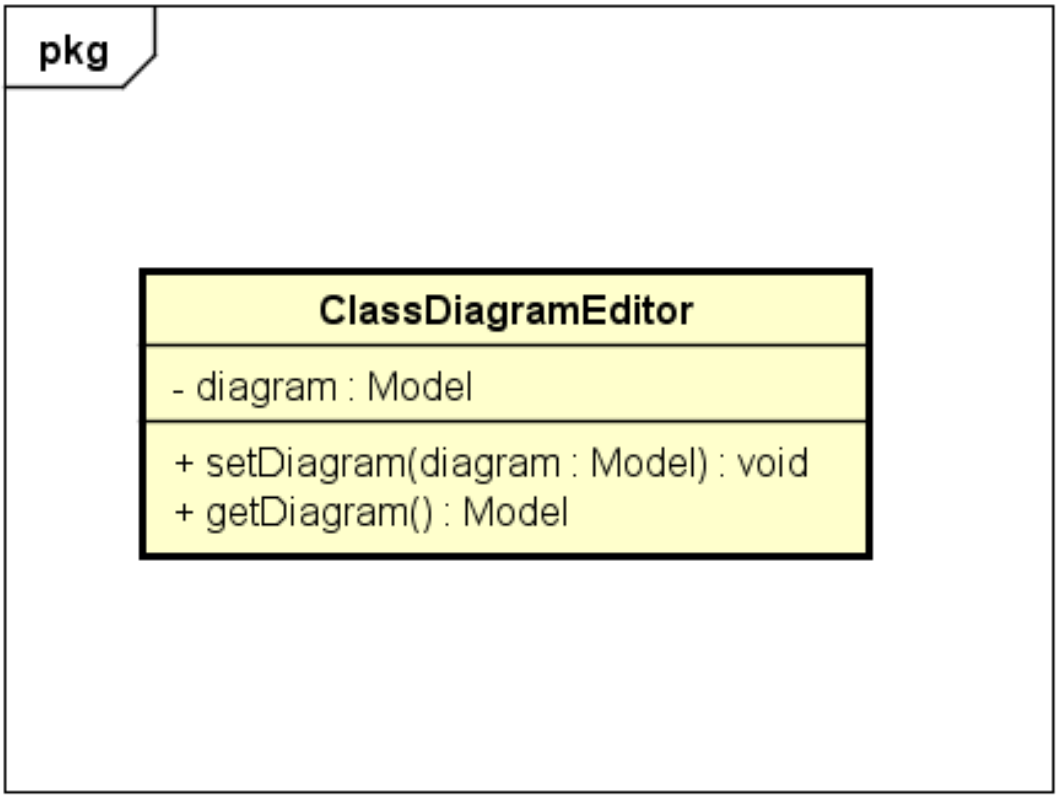

Figura 5: Front-end::View::ClassDiagramEditor

- **Descrizione**:
  Questa classe gestisce la visualizzazione dell'interfaccia grafica dell'editor del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per generare e gestire l'interfaccia grafica dell'editor del diagramma delle classi. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::ClassFactory`: utilizza `Front-end::View::ClassDiagramEditor` perché implementa le operazioni di creazione dei componenti del diagramma delle classi.
  - OUT: `Front-end::ViewModel::EditorObjController`: processare le richieste riguardanti la modifica dei blocchi nel diagramma delle classi.
- **Interfacce implementate**:

  - `Front-end::View::DiagramEditor`

- **Attributi**:
  - `diagram: Model`
    $oggetto_G$ che descrive il diagramma delle classi.
- **Metodi**:
  - `+ setDiagram(diagram: Model): void`
    si occupa di settare il diagramma delle classi rappresentato dall'editor.
    Parametri:
    * `diagram: Model`
      contiene il diagramma delle classi da settare.
  - `+ getDiagram(): Model`
    si occupa di ritornare il diagramma delle classi rappresentato dall'editor.

3.1.1.2.5 ActivityDiagramEditor

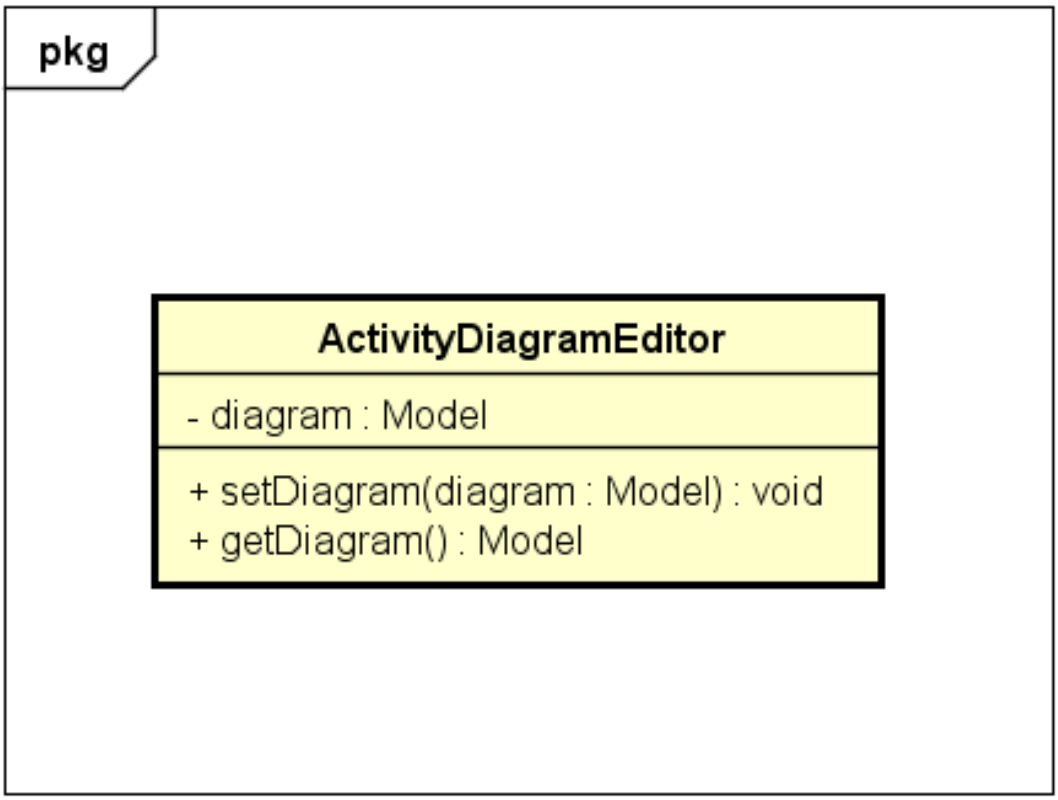

Figura 6: Front-end::View::ActivityDiagramEditor

- **Descrizione**:
  Questa classe gestisce la visualizzazione dell'interfaccia grafica dell'editor del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per generare e gestire l'interfaccia grafica dell'editor del diagramma delle attività. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::ActivityFactory`: utilizza `Front-end::View ::ActivityDiagramEditor` perché implementa le operazioni di creazione dei componenti del diagramma delle attività;
  - OUT: `Front-end::ViewModel::EditorObjController`: necessaria per processare le richieste riguardanti la modifica dei blocchi nel diagramma delle attività.

- **Interfacce implementate**:
  - `Front-end::View::DiagramEditor`
- **Attributi**:
  - `- diagram: Model`
    oggetto che descrive il diagramma delle attività.
- **Metodi**:
  - `+ setDiagram(diagram: Model): void`
    si occupa di settare il diagramma delle attività rappresentato dall'editor.
    Parametri:
    - `diagram: Model`
      contiene il diagramma delle attività da settare.
  - `+ getDiagram(): Model`
    si occupa di ritornare il diagramma delle attività rappresentato dall'editor.

3.1.1.2.6 DiagramPalette

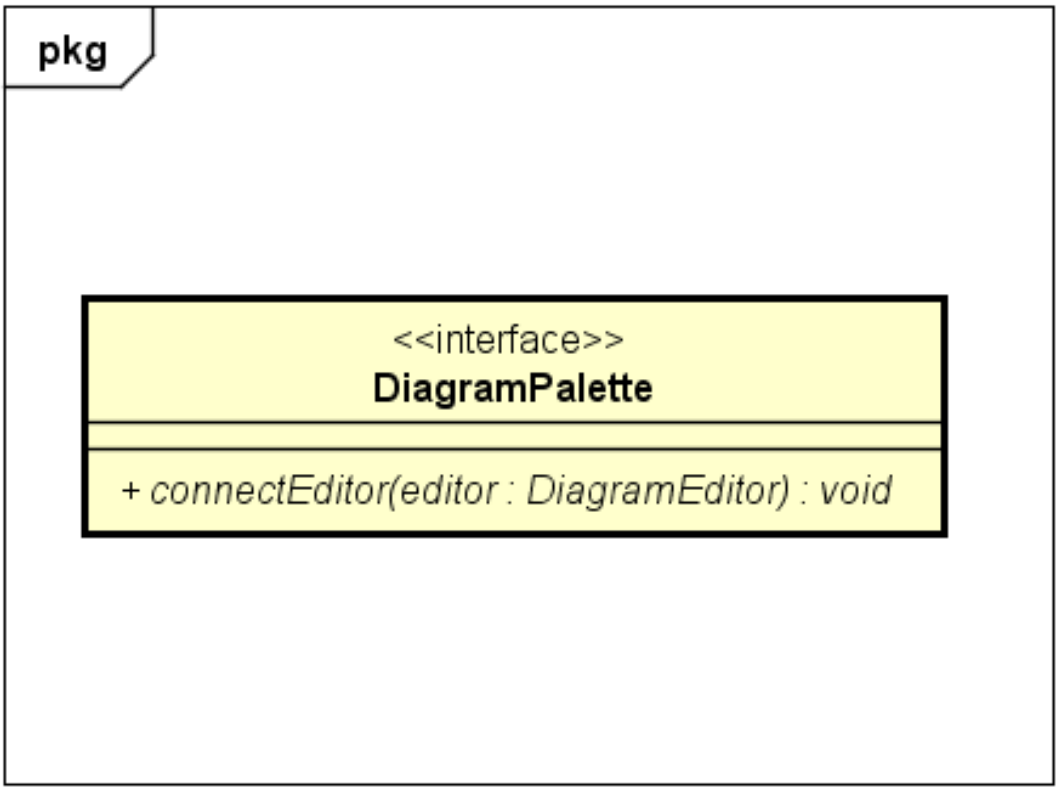

Figura 7: Front-end::View::DiagramPalette

- **Descrizione**:
  Questa interfaccia rappresenta la struttura per ciascuna palette del diagramma. Rappresenta l'interfaccia AbstractProduct del design pattern Abstract Factory.
- **Utilizzo**:
  Viene utilizzata per fornire un'interfaccia grafica comune alle palette dei diagrammi. È contenuta in `Front-end::View::SinglePageApp`. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::SinglePageApp`: utilizza `Front-end::View::DiagramPalette` perché deve visualizzare la palette corretta per ciascun diagramma.
- **Sottoclassi**:
  - `Front-end::View::ClassDiagramPalette`

  - `Front-end::View::ActivityDiagramPalette`

- **Attributi**: Assenti.
- **Metodi**:
  - `+ connectEditor(editor: DiagramEditor): void`
    si occupa di collegare la palette al rispettivo editor.
    Parametri:
    * `editor: DiagramEditor`
      rappresenta l'editor a cui collegare la palette.

3.1.1.2.7 ClassDiagramPalette

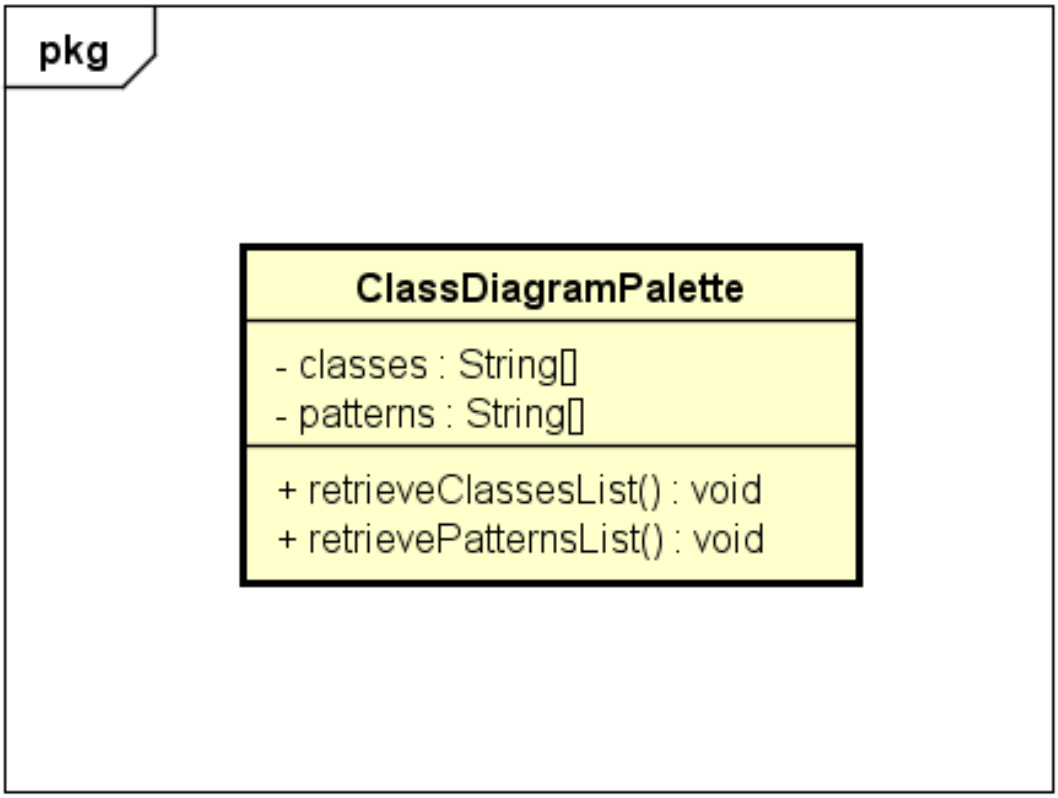

Figura 8: Front-end::View::ClassDiagramPalette

- **Descrizione**:
  Questa classe gestisce la visualizzazione dell'interfaccia grafica della palette del diagramma delle classi.

- **Utilizzo**:
  Viene utilizzata per generare e gestire l'interfaccia grafica della palette del diagramma delle classi. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::ClassFactory`: utilizza `Front-end::View::ClassDiagramPalette` perché implementa le operazioni di creazione dei componenti del diagramma delle classi all'interno della palette del diagramma delle classi;
  - OUT: `Front-end::ViewModel::PaletteObjController`: necessaria per processare le richieste di inserimento blocchi dalla palette all'interno del diagramma delle classi;
  - OUT: `Front-end::ViewModel::PaletteCommandController`: necessaria per processare le richieste di inserimento di elementi da libreria all'interno del diagramma delle classi.
- **Interfacce implementate**:
  - `Front-end::View::DiagramPalette`
- **Attributi**:
  - `- classes: String[]`
    array che contiene la lista dei diagrammi delle classi presenti in libreria.
  - `- patterns: String[]`
    array che contiene la lista dei diagrammi dei design pattern presenti in libreria.
- **Metodi**:
  - `+ retrieveClassesList(): void`
    si occupa di popolare l'array `classes` con la lista dei diagrammi delle classi presenti in libreria. La lista viene fornita dal back-end al quale la classe si interfaccia tramite chiamate REST (sfruttando la classe di servizio apposita).
  - `+ retrievePatternsList(): void`
    si occupa di popolare l'array `patterns` con la lista dei diagrammi dei design pattern presenti in libreria. La lista viene fornita dal back-end al quale la classe si interfaccia tramite chiamate REST (sfruttando la classe di servizio apposita).

3.1.1.2.8 ActivityDiagramPalette

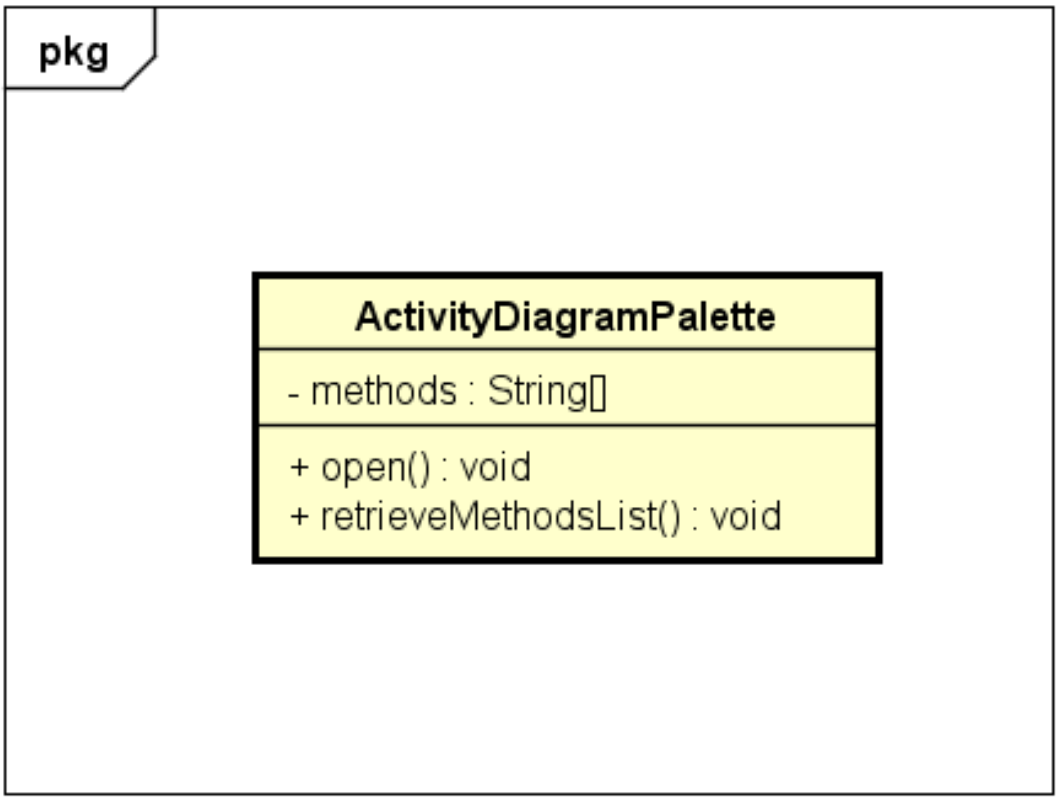

Figura 9: Front-end::View::ActivityDiagramPalette

- **Descrizione**:
  Questa classe gestisce la visualizzazione dell'interfaccia grafica delle palette del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per generare e gestire l'interfaccia grafica della palette del diagramma delle attività. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::ActivityFactory`: utilizza `Front-end::View ::ActivityDiagramPalette` perché implementa le operazioni di creazione dei componenti del diagramma delle attività all'interno della palette del diagramma delle attività.
  - OUT: `Front-end::ViewModel::PaletteObjController`: necessaria per processare le richieste di inserimento blocchi dalla palette all'interno del diagram-

ma delle attività;

- OUT: `Front-end::ViewModel::PaletteCommandController`: necessaria per processare le richieste di inserimento di elementi da libreria all'interno del diagramma delle attività.

- **Interfacce implementate**:
  - `Front-end::View::DiagramPalette`
- **Attributi**:
  - `- methods: String[]`
    array che contiene la lista dei diagrammi dei metodi presenti in libreria.
- **Metodi**:
  - `+ open(): void`
    si occupa di effettuare lo switch tra le funzionalità di palette di libreria o palette di blocchi.
  - `+ retrieveMethodsList(): void`
    si occupa di popolare l'array `methods` con la lista dei diagrammi dei metodi presenti in libreria.

3.1.1.2.9 DiagramFactory

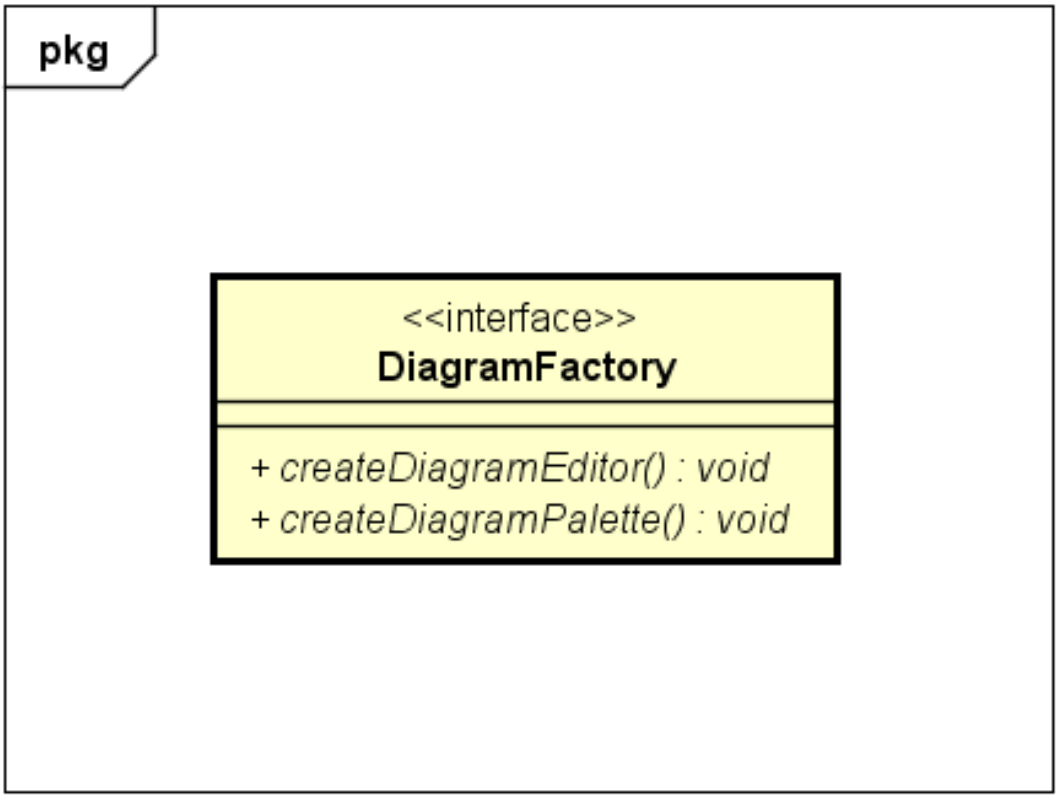

Figura 10: Front-end::View::DiagramFactory

- **Descrizione**:
  Questa interfaccia rappresenta la struttura per le operazioni di creazione dei diagrammi. Rappresenta l'interfaccia AbstractFactory del design pattern Abstract Factory.
- **Utilizzo**:
  Viene utilizzata per fornire un interfaccia comune alle factory dei vari tipi di diagrammi. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - IN: `Front-end::View:SinglePageApp`: utilizza `Front-end::View::DiagramFactory` perché deve fornire un interfaccia comune alle factory dei vari tipi di diagrammi.
- **Sottoclassi**:
  - `Front-end::View::ClassFactory`

  - `Front-end::View::ActivityFactory`

- **Attributi**: Assenti.
- **Metodi**:
  - `+ createDiagramEditor(): void`
    si occupa di creare l'editor dei diagrammi.
  - `+ createDiagramPalette(): void`
    si occupa di creare la palette associata all'editor dei diagrammi.

3.1.1.2.10 ClassFactory

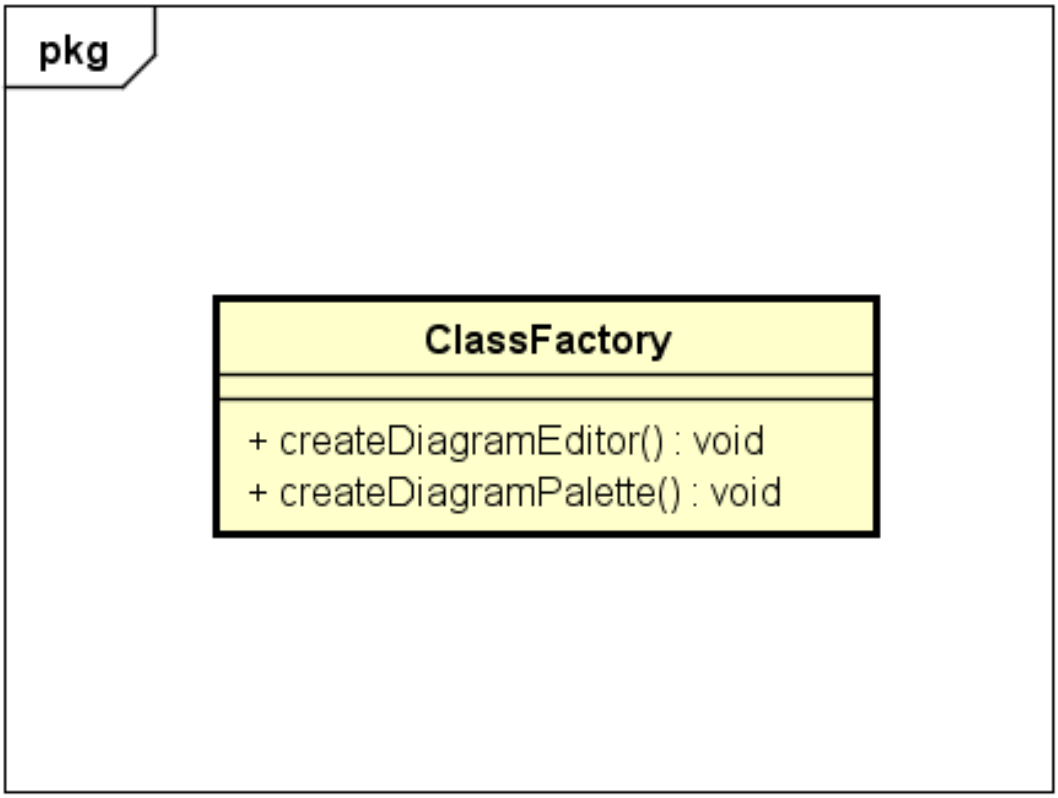

Figura 11: Front-end::View::ClassFactory

- **Descrizione**:
  Questa classe implementa le operazioni di creazione dei componenti del diagramma delle classi. Rappresenta la classe ConcreteFactory del design pattern Abstract Factory.

- **Utilizzo**:
  Viene utilizzata per instanziare i componenti concreti del diagramma delle classi, implementando l'interfaccia `Front-end::View::DiagramFactory`. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - OUT: `Front-end::View:ClassDiagramEditor`: necessaria per istanziare il componente concreto editor del diagramma delle classi;
  - OUT: `Front-end::View:ClassDiagramPalette`: necessaria per istanziare il componente concreto palette del diagramma delle classi.
- **Interfacce implementate**:
  - `Front-end::View::DiagramFactory`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ createDiagramEditor(): void`
    si occupa di creare l'editor dei diagrammi delle classi.
  - `+ createDiagramPalette(): void`
    si occupa di creare la palette associata all'editor dei diagrammi delle classi.

3.1.1.2.11 ActivityFactory

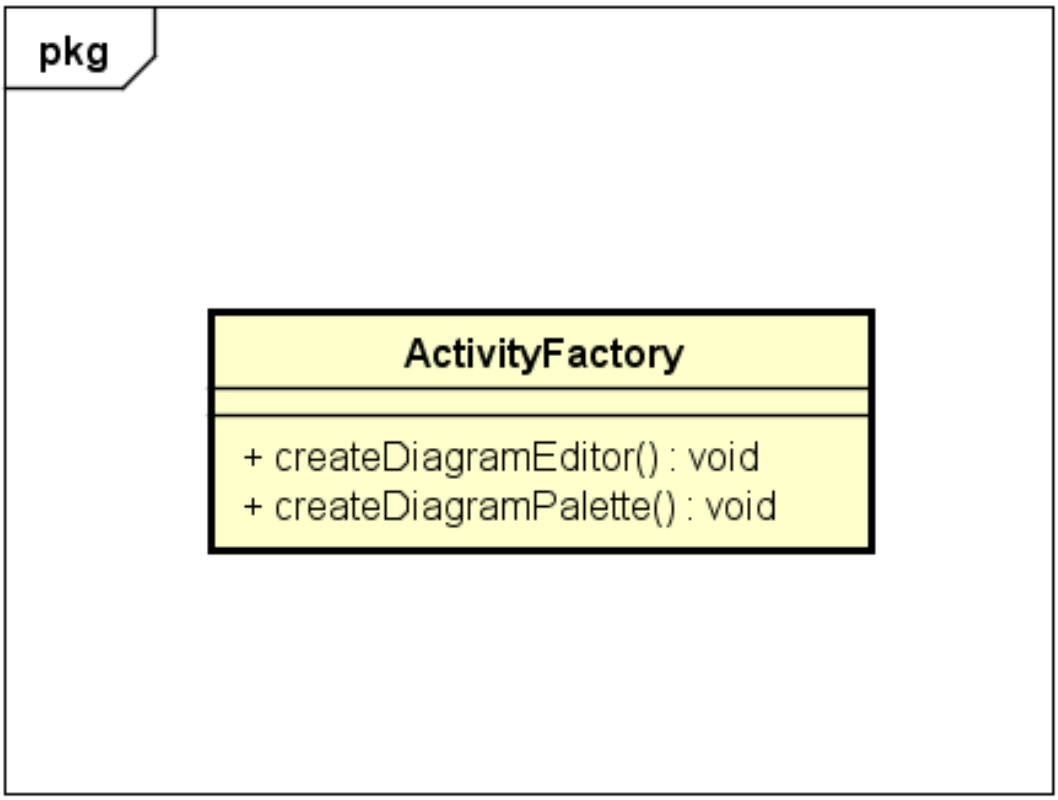

Figura 12: Front-end::View::ActivityFactory

- **Descrizione**:
  Questa classe implementa le operazioni di creazione dei componenti del diagramma delle attività. Rappresenta la classe ConcreteFactory del design pattern Abstract Factory.
- **Utilizzo**:
  Viene utilizzata per instanziare i componenti concreti del diagramma delle attività, implementando l'interfaccia `Front-end::View::DiagramFactory`. Utilizza la libreria GoJS.
- **Relazioni con altre classi**:
  - OUT: `Front-end::View:ActivityDiagramEditor`: necessaria per istanziare il componente concreto editor del diagramma delle attività;
  - OUT: `Front-end::View:ActivityDiagramPalette`: necessaria per istanziare il componente concreto palette del diagramma delle attività.

- **Interfacce implementate**:
  - `Front-end::View::DiagramFactory`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ createDiagramEditor(): void`
    si occupa di creare l'editor dei diagrammi delle attività.
  - `+ createDiagramPalette(): void`
    si occupa di creare la palette associata all'editor dei diagrammi delle attività.

3.1.2 Front-end::ViewModel

3.1.2.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene tutte le classi necessarie a regolare la comunicazione e l'interazione tra i package `Front-end::View` e `Front-end::Model` e funge da responsabile per la gestione logica della `Front-end::View`. Fornisce quindi i dati dal modello in una forma che la vista può usare facilmente.
- **Framework esterni**:
  - Angular2

3.1.2.2 Classi

3.1.2.2.1 EditorObjController

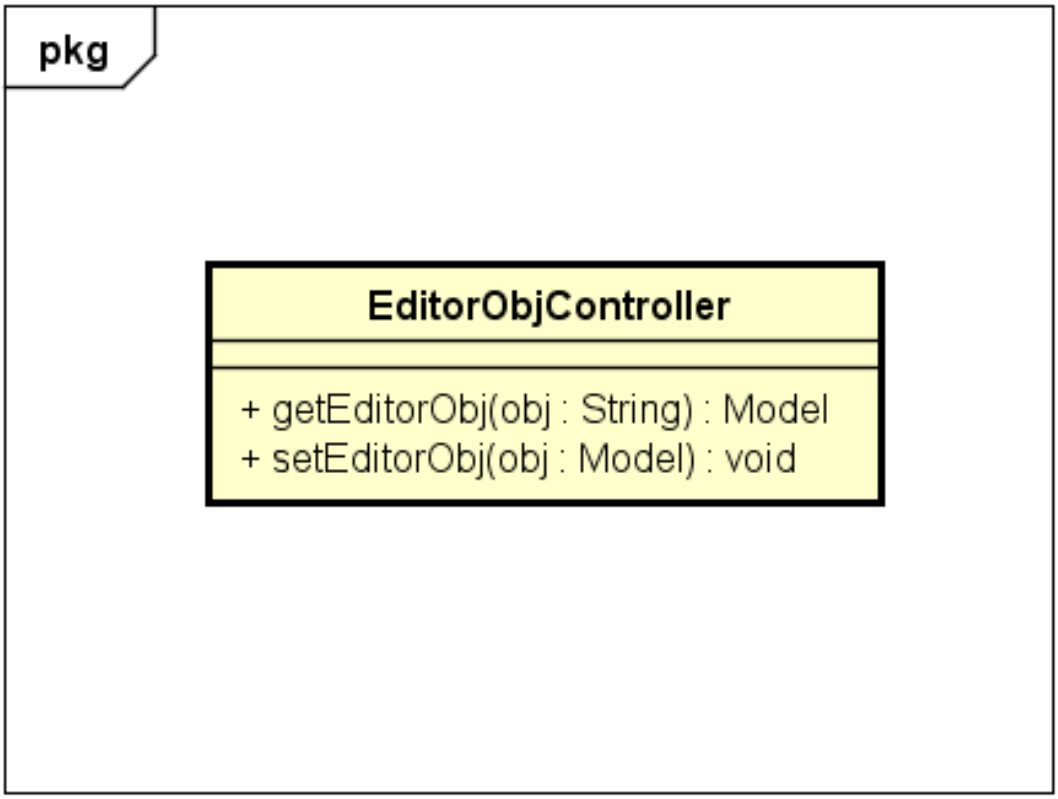

Figura 13: Front-end::ViewModel::EditorObjController

- **Descrizione**:
  Questa classe gestisce le richieste degli editor dei diagrammi interfacciandosi con la parte del
  `Front-end::Model`.
- **Utilizzo**:
  Viene utilizzata per processare le richieste riguardanti la modifica dei blocchi negli editor dei diagrammi.
- **Relazioni con altre classi**:
  - IN: `Front-end::Model::View::ClassDiagramEditor`: utilizza `Front-end ::ViewModel::EditorObjController` per comunicare con il `Front-end::Model`;

  - IN: `Front-end::Model::View::ActivityDiagramEditor`: utilizza `Front-end ::ViewModel::EditorObjController` per comunicare con il `Front-end::Model`;
  - OUT: `Front-end::Model::Objects::BaseDiaObj`: necessaria per ricevere le richieste provenienti da `Front-end::View::ClassDiagramEditor` e `Front-end::View ::ActivityDiagramEditor`.

- **Attributi**: Assenti.
- **Metodi**:
  - `+ getEditorObj(obj: String): Model`
    si occupa di ritornare l'oggetto $JSON_G$ rappresentante un oggetto GoJs di tipo Model contenente le informazioni riguardanti l'oggetto del diagramma richiesto.
    Parametri:
    * `obj: String`
      stringa che descrive l'id dell'oggetto del diagramma di cui restituire il JSON rappresentante un oggetto GoJs di tipo Model..
  - `+ setEditorObj(obj: Model): void`
    si occupa di aggiornare le informazioni di un oggetto del diagramma modificato.
    Parametri:
    * `obj: Model`
      oggetto che contiene le nuove informazioni su un oggetto del diagramma.

3.1.2.2.2 PaletteObjController

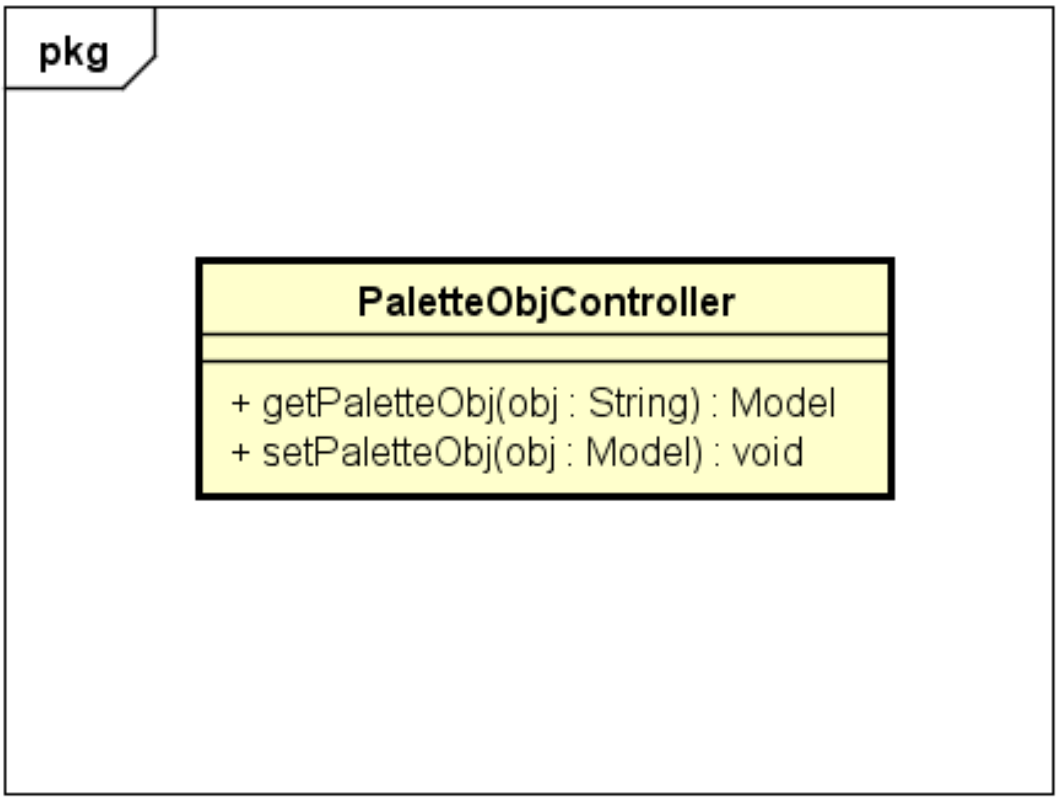

Figura 14: Front-end::ViewModel::PaletteObjController

- **Descrizione**:
  Questa classe gestisce le richieste della palette dei diagrammi interfacciandosi con la parte del
  `Front-end::Model`.
- **Utilizzo**:
  Viene utilizzata per processare le richieste di inserimento blocchi dalla palette all'interno dei diagrammi.
- **Relazioni con altre classi**:
  - IN: `Front-end::Model::View::ClassDiagramPalette`: utilizza `Front-end::ViewModel::PaletteObjController` per gestisce la visualizzazione dell'interfaccia grafica della palette del diagramma delle classi e comunicare con il `Front-end::Model`;

  - IN: `Front-end::Model::View::ActivityDiagramPalette`: utilizza `Front-end ::ViewModel::PaletteObjController` per gestisce la visualizzazione dell'interfaccia grafica della palette del diagramma delle attività e comunicare con il `Front-end::Model`;
  - OUT: `Front-end::Model::Objects::BaseDiaObj`: necessaria per ricevere le richieste provenienti da `Front-end::View::ClassDiagramEditor` e `Front-end::View ::ActivityDiagramEditor`.

- **Attributi**: Assenti.
- **Metodi**:
  - `+ getPaletteObj(obj: String): Model`
    si occupa di ritornare l'oggetto JSON rappresentante un oggetto GoJs di tipo Model1 contenente le informazioni riguardanti l'oggetto della palette del diagramma richiesto.
    Parametri:
    * `obj: String`
      stringa che descrive l'id dell'oggetto della palette del diagramma di cui restituire il JSON rappresentante un oggetto GoJs di tipo Model.
  - `+ setPaletteObj(obj: Model): void`
    si occupa di aggiornare le informazioni di un oggetto della palette del diagramma modificato.
    Parametri:
    * `obj: Model`
      oggetto che contiene le nuove informazioni su un oggetto della palette del diagramma.

3.1.2.2.3 CodeCommandController

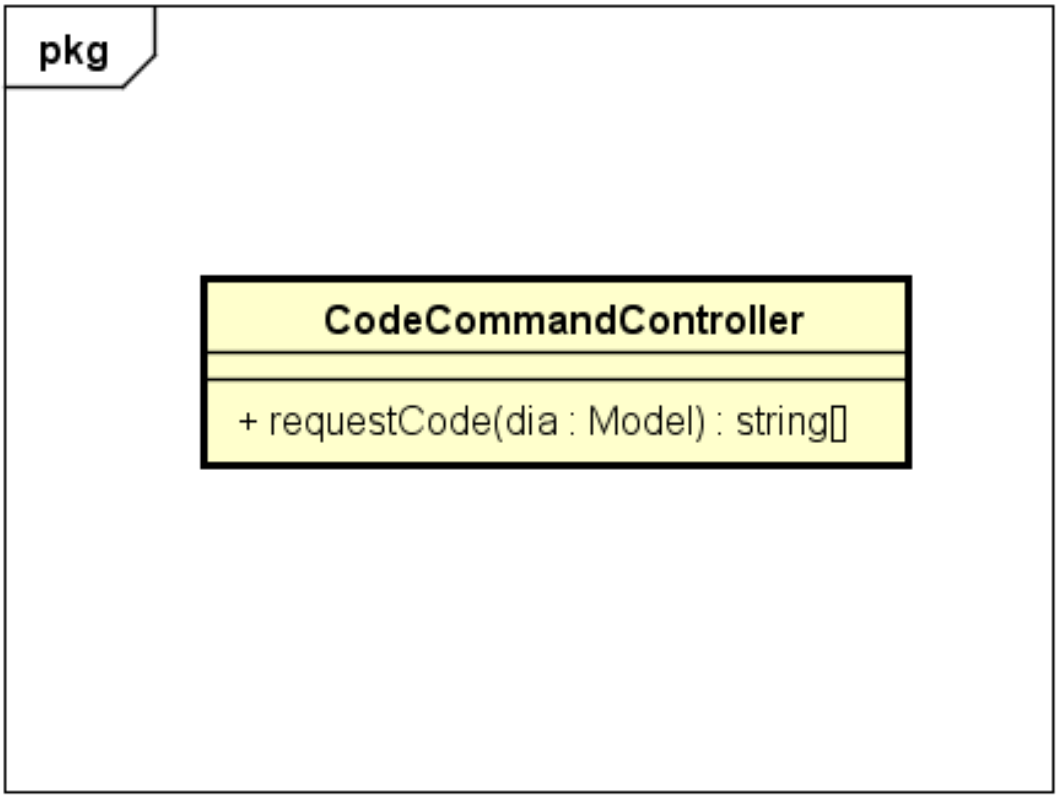

Figura 15: Front-end::ViewModel::CodeCommandController

- **Descrizione**:
  Questa classe gestisce le richieste dell'editor del codice generato interfacciandosi con la parte del `Front-end::Model`.
- **Utilizzo**:
  Viene utilizzata per processare le richieste provenienti dall'editor del codice riguardanti la generazione di quest'ultimo.
- **Relazioni con altre classi**:
  - IN: `Front-end::View::CodeEditor`: utilizza `Front-end::ViewModel ::CodeCommandController` per la visualizzazione dell'interfaccia grafica dell'editor del codice generato e comunicare con il `Front-end::Model`;
  - OUT: `Front-end::Model::Commands::Command`: necessaria per ricevere la richiesta processata di generazione codice dell'utente proveniente da `Front-end::View::CodeEdi`
- **Attributi**: Assenti.

- **Metodi**:
  - `+ requestCode(dia: Model): String`
    si occupa di ritornare il codice generato a partire dai diagrammi creati.
    Parametri:
    * `dia: Model`
      oggetto contenente il diagramma delle classi e i diagrammi delle attività per ciascun metodo definito all'interno delle classi.

3.1.2.2.4 PaletteCommandController

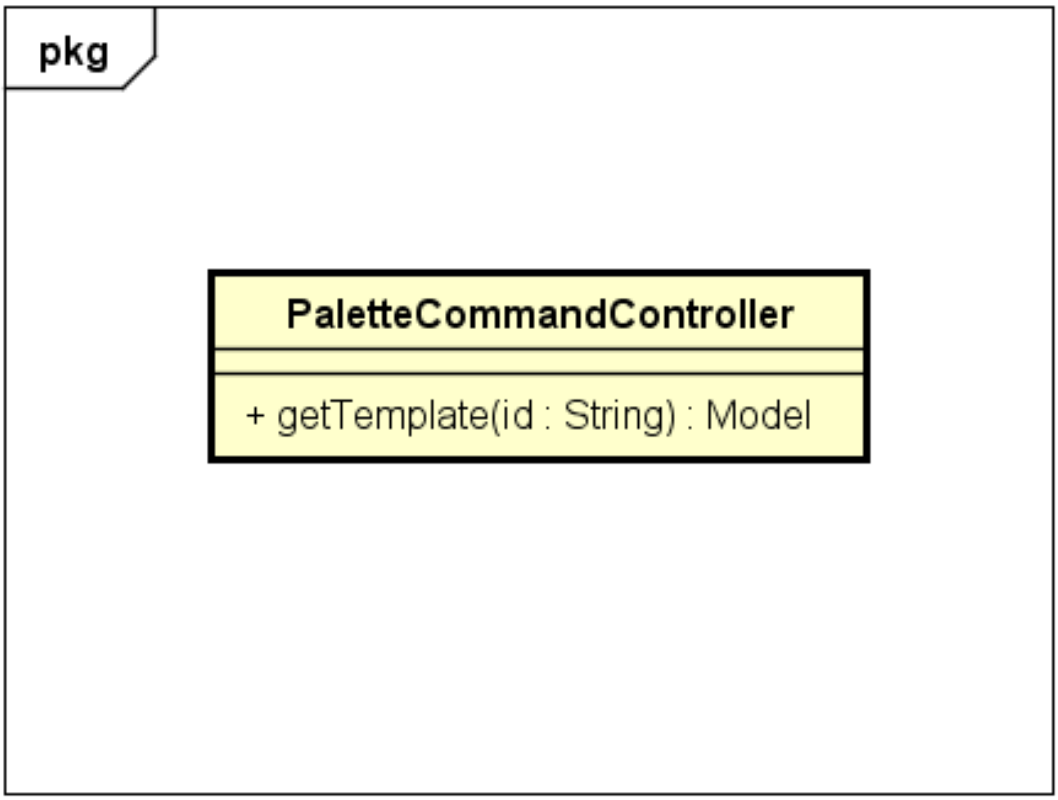

Figura 16: Front-end::ViewModel::PaletteCommandController

- **Descrizione**:
  Questa classe gestisce le richieste della palette dei diagrammi interfacciandosi con la parte del `Front-end::Model`.

- **Utilizzo**:
  Viene utilizzata per processare le richieste di inserimento di elementi da libreria all'interno dei diagrammi.
- **Relazioni con altre classi**:
  - IN: `Front-end::Model::View::ClassDiagramPalette`: utilizza `Front-end ::ViewModel::PaletteCommandController` per gestire l'interfaccia grafica della palette del diagramma delle classi e comunicare con il `Front-end::Model`;
  - IN: `Front-end::Model::View::ActivityDiagramPalette`: utilizza `Front-end ::ViewModel::PaletteCommandController` per gestire l'interfaccia grafica della palette del diagramma delle attività e comunicare con il `Front-end::Model`;
  - OUT: `Front-end::Model::Commands::Command`: necessaria per ricevere le richieste processate di $template_G$ provenienti da `Front-end::View::ClassDiagramPalette` e `Front-end::View ::ActivityDiagramPalette`.
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getTemplate(id: String): Model`
    si occupa di ritornare l'oggetto JSON rappresentante un oggetto GoJs di tipo Model contenente il template di libreria specificato.
    Parametri:
    * `id: String`
      stringa contenente l'identificativo del template di libreria da ritornare.

3.1.3 Front-end::Model

3.1.3.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene i modelli dei blocchi utilizzati dalla `Front-end::view`, i comandi che permettono la richiesta di generazione codice e richiesta di template con il package `Front-end::Model::Services` che gestisce l'interazione con il back-end.
- **Package contenuti**:
  - Front-end::Model::Objects
  - Front-end::Model::Commands
  - Front-end::Model::Services
- **Framework esterni**:

  - GoJS
  - Angular2

3.1.4 Front-end::Model::Objects

3.1.4.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene tutti i blocchi che costituiscono il diagramma delle classi `Front-End::Model::Objects::ClassObjects` ed il diagramma delle attività `Front-End::Model::Objects::ActivityObjects`.
- **Package contenuti**:
  - `Front-end::Model::Objects::ClassObjects`: contiene tutti i blocchi che vanno a comporre il diagramma delle classi
  - `Front-end::Model::Objects::ActivityObjects`: contiene tutti i blocchi che vanno a comporre il diagramma delle attività.
- **Framework esterni**:
  - GoJS

3.1.4.2 Classi

3.1.4.2.1 BaseDiaObj

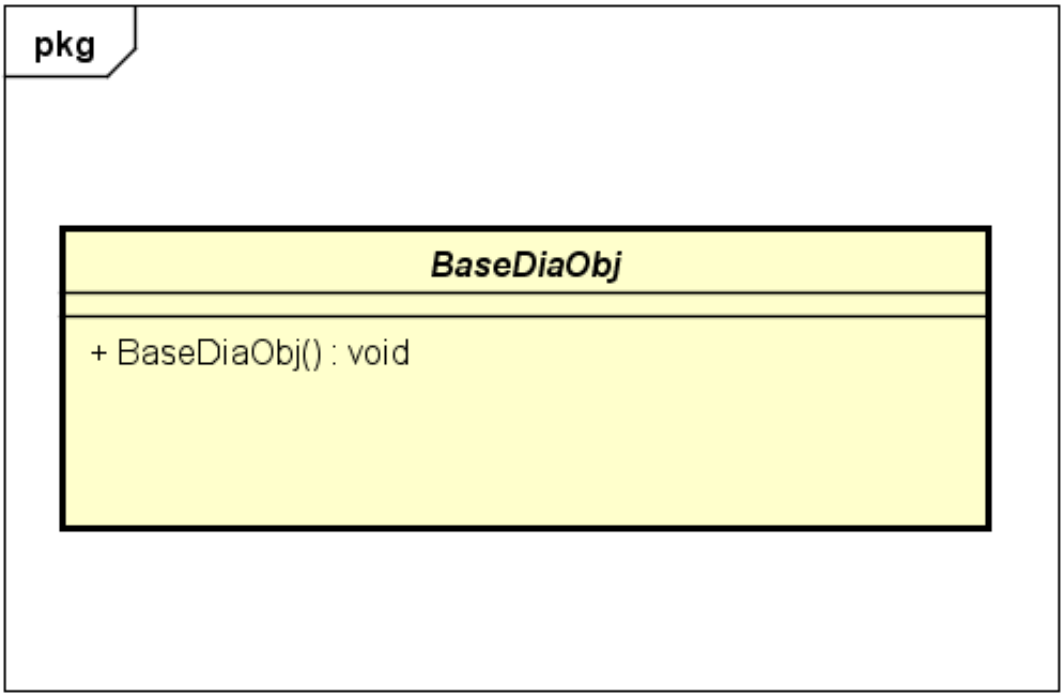

Figura 17: Front-end::Model::Objects::BaseDiaObj

- **Descrizione**:
  Questa classe astratta rappresenta un contratto comune tra le classi `Front-end::Model::Objects::ClassObjects::ClassDiaObj` e `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`.
- **Utilizzo**:
  Viene utilizzata per rappresentare un oggetto base comune ai diagrammi delle classi e delle attività.
- **Relazioni con altre classi**:
  - IN: `Front-end::ViewModel::EditorObjController`: utilizza `Front-end::Model::Objects::BaseDiaObj` per inoltrare le richieste provenienti da `Front-end::View::ClassDiagramEditor` e `Front-end::View::ActivityDiagramEditor`.
  - IN: `Front-end::ViewModel::PaletteObjController`: utilizza `Front-end::Model::Objects::BaseDiaObj` per inoltrare le richieste di inserimento blocchi dalla palette all'interno dei diagrammi.

- **Sottoclassi**:
  - `Front-end::Model::Objects::ClassObjects::ClassDiaObj`
  - `Front-end::Model::Objects::ClassObjects::ActivityDiaObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ BaseDiaObj(): void`
    si occupa di costruire la classe BaseDiaObj vista come un oggetto base comune ai diagrammi delle classi e delle attività.

3.1.5 Front-end::Model::Objects::ClassObjects

3.1.5.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene tutti i blocchi che vanno a comporre il diagramma delle classi.
- **Framework esterni**:
  - GoJS

3.1.5.2 Classi

3.1.5.2.1 ClassDiaObj

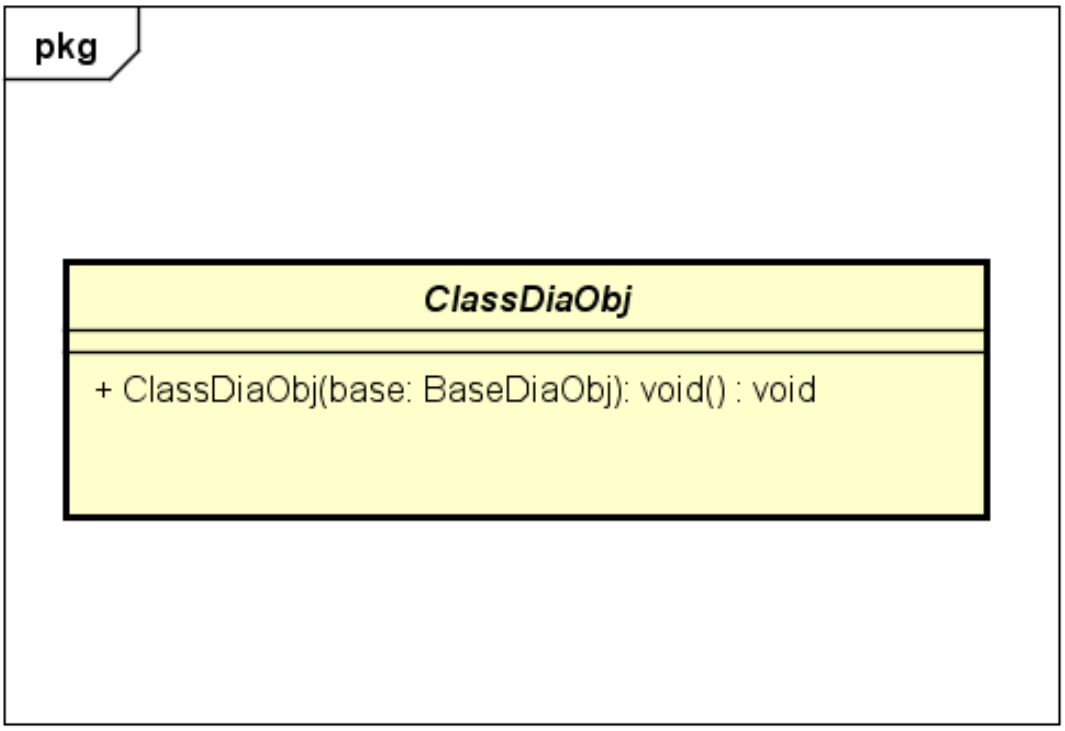

Figura 18: Front-end::Model::Objects::ClassObjects::ClassDiaObj

- **Descrizione**:
  Questa classe astratta rappresenta un contratto comune tra le classi che rappresentano i blocchi del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per rappresentare un oggetto base comune a tutti i blocchi del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::BaseDiaObj`
- **Sottoclassi**:
  - `Front-end::Model::Objects::ClassObjects::Class`
  - `Front-end::Model::Objects::ClassObjects::Comment`

  - `Front-end::Model::Objects::ClassObjects::Relation`

- **Attributi**: Assenti.
- **Metodi**:
  - `+ ClassDiaObj(base: BaseDiaObj): void`
    si occupa di costruire la classe BaseDiaObj vista come un contratto comune tra le classi che rappresentano i blocchi del diagramma delle classi a partire dalla classe base astratta BaseDiaObj.
    Parametri:
    * `base: BaseDiaObj`
      rappresenta un oggetto base comune ai diagrammi delle classi e delle attività.

3.1.5.2.2 Class

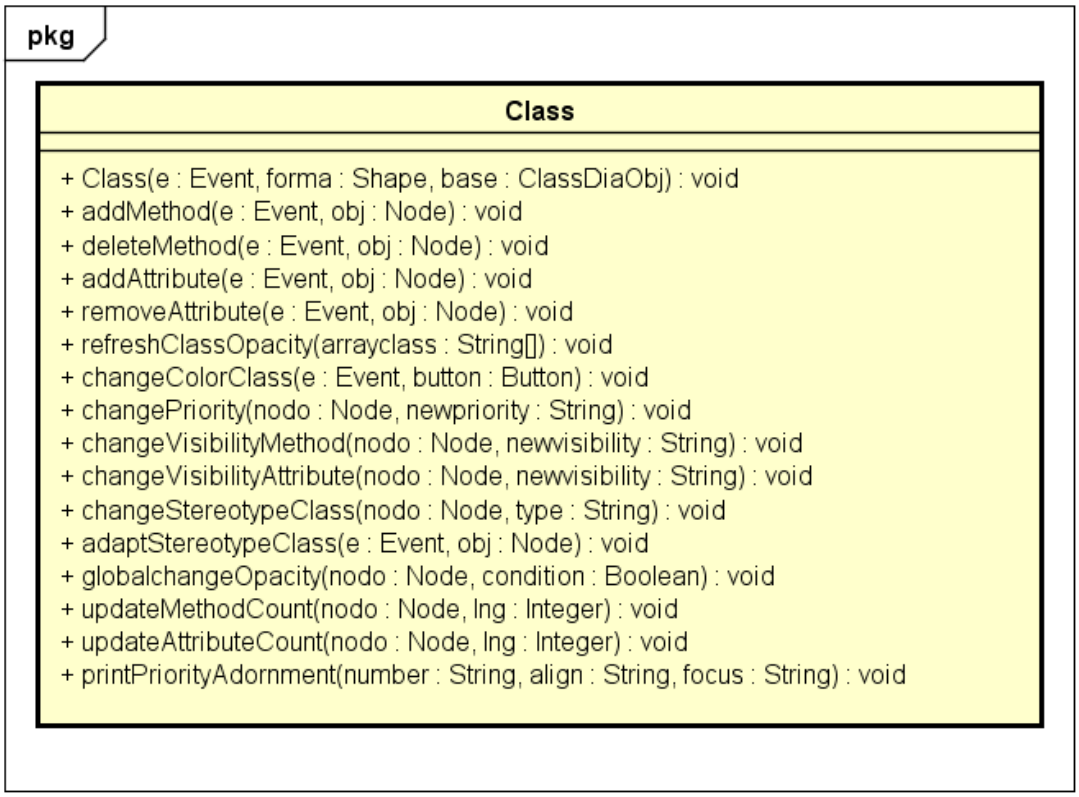

Figura 19: Front-end::Model::Objects::ClassObjects::Class

- **Descrizione**:
  Questa classe rappresenta un blocco classe del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti classe all'interno del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::ClassDiaObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ Class(nodo: Node, forma: Shape, base: ClassDiaObj): void`
    si occupa di costruire il blocco classe del diagramma delle classi a partire dalla classe base ClassDiaObj.
    Parametri:
    * `nodo: Node`
      rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.
    * `forma: Shape`
      rappresenta la forma di un blocco del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.
  - `+ addMethod(e: Event, obj: Node): void`
    si occupa di aggiungere un metodo alla classe del diagramma delle classi.
    Parametri:
    * `e: Event`
      rappresenta un evento che si $verifica_G$ nel diagramma delle classi. Il tipo Event è definito dalla libreria GoJS.
    * `obj: Node`
      si riferisce a un nodo di tipo Node.
  - `+ deleteMethod(e: Event, obj: Node): void`
    si occupa di rimuovere un metodo dalla classe del diagramma delle classi.
    Parametri:
    * `e: Event`
      rappresenta un evento che si verifica nel diagramma delle classi. Il tipo Event è definito dalla libreria GoJS.
    * `obj: Node`
      il parametro obj si riferisce a un nodo di tipo Node.

- `+ addAttribute(e: Event, obj: Node): void`
  si occupa di aggiungere un attributo alla classe del diagramma delle classi.
  Parametri:
  - `e: Event`
    rappresenta un evento che si verifica nel diagramma delle classi. Il tipo Event è definito dalla libreria GoJS.
  - `obj: Node`
    si riferisce a un nodo di tipo Node.
- `+ removeAttribute(e: Event, obj: Node): void`
  si occupa di rimuovere un attributo dalla classe del diagramma delle classi.
  Parametri:
  - `e: Event`
    rappresenta un evento che si verifica nel diagramma delle classi. Il tipo Event è definito dalla libreria GoJS.
  - `obj: Node`
    si riferisce a un nodo di tipo Node.
- `+ refreshClassOpacity(arrayclass: String[]): void`
  si occupa di aggiornare la priorità di tutte le classi quando l'utente modifica la priorità della classe scelta del diagramma delle classi.
  Parametri:
  - `arrayclass: String[]`
    rappresenta un array che contiene nodi classe passato come stringa al metodo.
- `+ changeColorClass(e: Event, button: Button): void`
  si occupa di cambiare il colore di sfondo di un classe del diagramma delle classi.
  Parametri:
  - `e: Event`
    rappresenta un evento che si verifica nel diagramma delle classi. Il tipo Event è definito dalla libreria GoJS.
  - `button: Button`
    rappresenta un bottone. Il tipo Button è definito dalla libreria GoJS.
- `+ changePriority(nodo: Node, newpriority: String): void`
  si occupa di modificare la priorità della classe del diagramma delle classi.
  Parametri:

    * `nodo: Node`
      rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

    * `newpriority: String`
      rappresenta la stringa della nuova priorità di un blocco del diagramma delle classi.

- `+ changeVisibilityMethod(nodo: Node, newvisibility: String): void`
  si occupa di modificare la visibilità di un metodo della classe del diagramma delle classi.
  Parametri:

    * `nodo: Node`
      rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

    * `newvisibility: String`
      rappresenta la stringa della nuova visibilità di un metodo della classe del diagramma delle classi.

- `+ changeVisibilityAttribute(nodo: Node, newvisibility: String): void`
  si occupa di modificare la visibilità di un attributo della classe del diagramma delle classi.
  Parametri:

    * `nodo: Node`
      rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

    * `newvisibility: String`
      rappresenta la stringa della nuova visibilità di un attributo di una classe del diagramma delle classi.

- `+ changeStereotypeClass(nodo: Node, type: String): void`
  si occupa di modificare lo stereotipo di una classe del diagramma delle classi.
  Parametri:

    * `nodo: Node`
      rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

    * `type: String`
      rappresenta la stringa della nuova visibilità di una classe del diagramma delle classi.

- `+ adaptStereotypeClass(e: Event, obj: Node): void`
  si occupa di modificare l'opacità dello stereotipo. Se il valore è default la

scritta è resa opaca, se ha un tipo la scritta è ben visibile.
Parametri:

  * `e: Event`
    rappresenta un evento che si verifica nel diagramma delle classi. Il tipo Event è definito dalla libreria GoJS.

  * `obj: Node`
    si riferisce a un nodo di tipo Node.

- `+ globalchangeOpacity(nodo: Node, condition: Boolean): void`
  si occupa di modificare l'opacità della classe in base al valore della condizione booleana.
  Parametri:

  * `nodo: Node`
    rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

  * `condizione: Boolean`
    rappresenta un valore booleano.

- `+ updateMethodCount(nodo: Node, lng: Boolean): void`
  si occupa di aggiornare il contatore della lista dei metodi della classe.
  Parametri:

  * `nodo: Node`
    rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

  * `lng: Integer`
    rappresenta la lunghezza dell'array della lista dei metodi.

- `+ updateAttributeCount(nodo: Node, lng: Integer): void`
  si occupa di aggiornare il contatore della lista degli attributi della classe.
  Parametri:

  * `nodo: Node`
    rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

  * `lng: Integer`
    rappresenta la lunghezza dell'array della lista degli attributi.

- `+ printPriorityAdornment(number: String, align: String, focus: String): void`
  si occupa di creare l'adornment per rappresentare la priorità di una classe del diagramma delle classi.
  Parametri:

* `number: String`
  rappresenta la stringa numero da visualizzare come priorità della classe.
* `align: String`
  rappresenta il posizionamento dell'adornment all'interno della classe.
* `focus: String`
  contribuisce a posizionare l'adornment all'interno della classe.

3.1.5.2.3 Comment

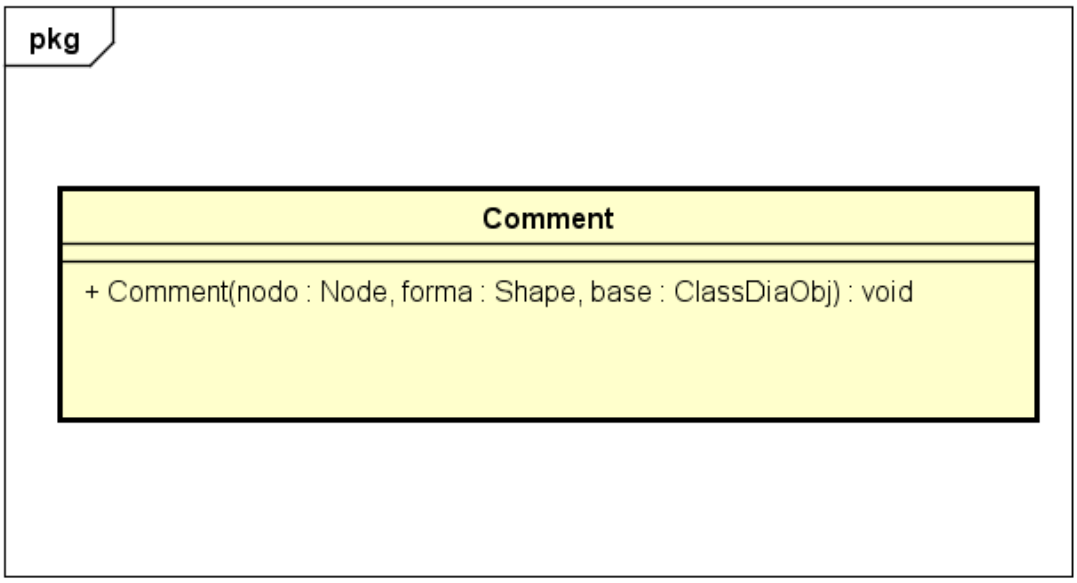

Figura 20: Front-end::Model::Objects::ClassObjects::Comment

- **Descrizione**:
  Questa classe rappresenta un blocco commento del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti commento all'interno del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::ClassDiaObj`
- **Attributi**: Assenti.
- **Metodi**:

- `+ Comment(nodo: Node, forma: Shape, base: ClassDiaObj): void`
  si occupa di costruire il blocco commento del diagramma delle classi a partire dalla classe base astratta ClassDiaObj.
  Parametri:
  - `nodo: Node`
    rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.
  - `forma: Shape`
    rappresenta la forma di un blocco del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.
  - `base: ClassDiaObj`
    rappresenta un oggetto base comune a tutti i blocchi del diagramma delle classi.

3.1.5.2.4 Relation

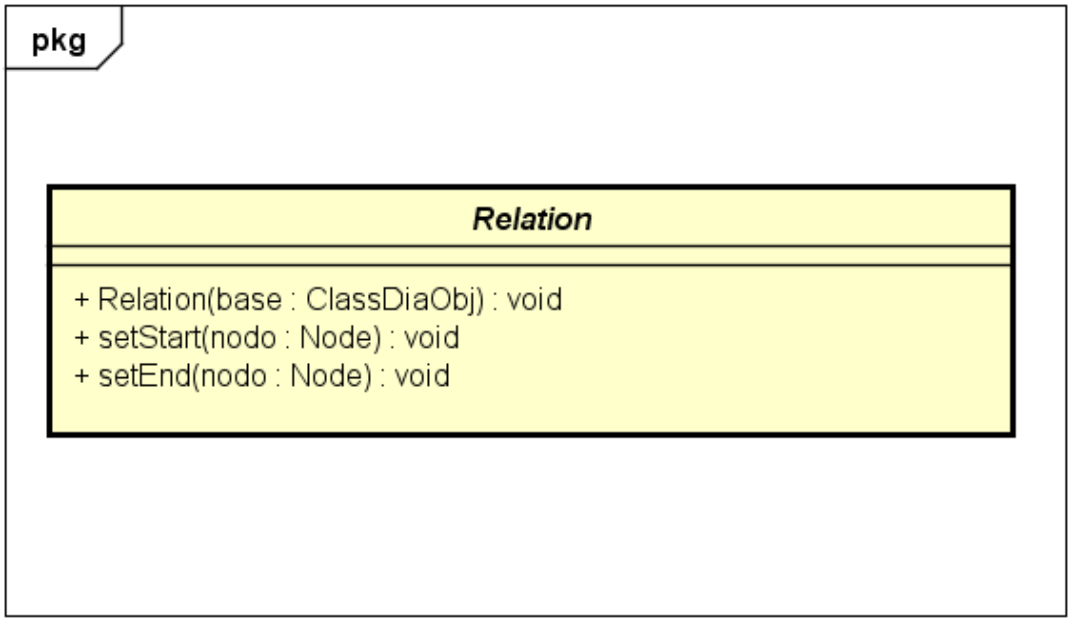

Figura 21: Front-end::Model::Objects::ClassObjects::Relation

- **Descrizione**:
  Questa classe astratta rappresenta un contratto comune tra le classi che rappresentano le relazioni del diagramma delle classi.

- **Utilizzo**:
  Viene utilizzata per rappresentare un oggetto base comune a tutte le relazioni del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::ClassDiaObj`
- **Sottoclassi**:
  - `Front-end::Model::Objects::ClassObjects::Dependency`
  - `Front-end::Model::Objects::ClassObjects::Association`
  - `Front-end::Model::Objects::ClassObjects::Aggregation`
  - `Front-end::Model::Objects::ClassObjects::Composition`
  - `Front-end::Model::Objects::ClassObjects::Generalization`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ Relation(base: ClassDiaObj): void`
    si occupa di costruire la classe Relation vista come un contratto comune tra le classi che rappresentano le relazioni del diagramma delle classi a partire dalla classe base astratta ClassDiaObj.
    Parametri:
    - `arrow: Link`
      rappresenta una relazione del diagramma delle classi. Il tipo Link è definito dalla libreria GoJS.
    - `forma: Shape`
      rappresenta la forma di una relazione del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.
    - `base: ClassDiaObj`
      rappresenta un oggetto base comune a tutti i blocchi del diagramma delle classi.
  - `+ setStart(nodo: Node): void`
    si occupa di selezionare il nodo di partenza al quale agganciare la relazione del diagramma delle classi.
    Parametri:
    - `nodo: Node`
      rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

- `+ setEnd(nodo: Node): void`
  si occupa di selezionare il nodo di fine al quale agganciare la relazione del diagramma delle classi.
  Parametri:
  - `nodo: Node`
    rappresenta un blocco del diagramma delle classi. Il tipo Node è definito dalla libreria GoJS.

3.1.5.2.5 Dependency

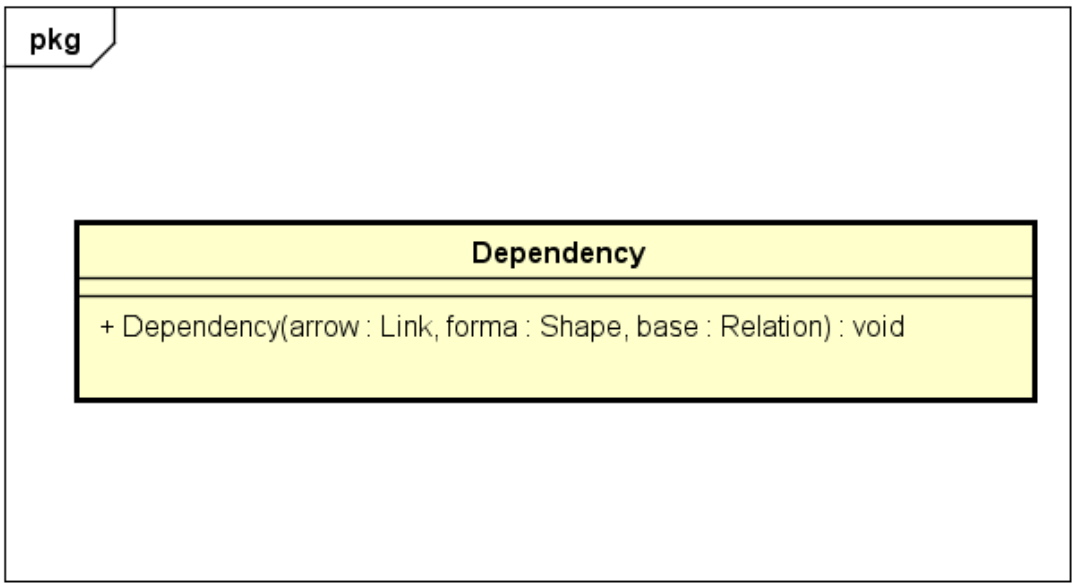

Figura 22: Front-end::Model::Objects::ClassObjects::Dependency

- **Descrizione**:
  Questa classe rappresenta una relazione di tipo dipendenza del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti relazione di tipo dipendenza all'interno del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::Relation`
- **Attributi**: Assenti.
- **Metodi**:

– `+ Dependency(arrow: Link, forma: Shape, base: Relation): void`
si occupa di costruire la relazione di tipo dipendenza del diagramma delle classi a partire dalla classe base astratta Relation.
Parametri:

* `arrow: Link`
rappresenta una relazione del diagramma delle classi. Il tipo Link è definito dalla libreria GoJS.

* `forma: Shape`
rappresenta la forma di una relazione del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.

* `base: Relation`
rappresenta un oggetto base comune a tutte le relazioni del diagramma delle classi.

3.1.5.2.6 Association

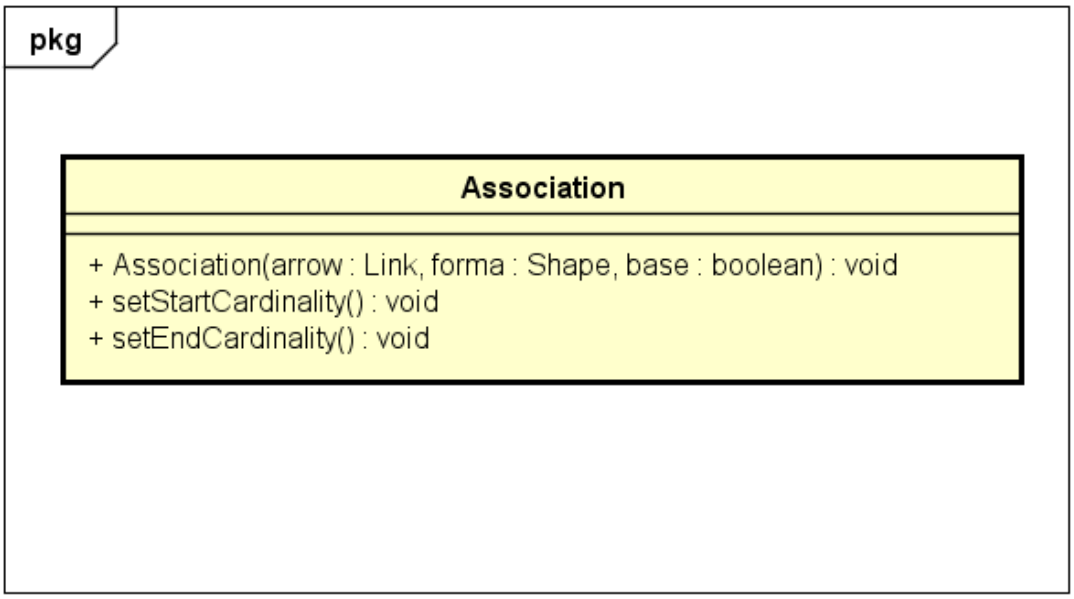

Figura 23: Front-end::Model::Objects::ClassObjects::Association

- **Descrizione**:
Questa classe rappresenta una relazione di tipo associazione del diagramma delle classi.

- **Utilizzo**:
  Viene utilizzata per istanziare oggetti relazione di tipo associazione all'interno del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::Relation`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ Association(arrow: Link, forma: Shape, base: Relation): void`
    si occupa di costruire la relazione di tipo associazione del diagramma delle classi a partire dalla classe base astratta Relation.
    Parametri:
    - `arrow: Link`
      rappresenta una relazione del diagramma delle classi. Il tipo Link è definito dalla libreria GoJS.
    - `forma: Shape`
      rappresenta la forma di una relazione del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.
    - `base: Relation`
      rappresenta un oggetto base comune a tutte le relazioni del diagramma delle classi.
  - `+ setStartCardinality(): void`
    si occupa di modificare la cardinalità di partenza della relazione associazione del diagramma delle classi.
  - `+ setEndCardinality(): void`
    si occupa di modificare la cardinalità di fine della relazione associazione del diagramma delle classi.

3.1.5.2.7 Aggregation

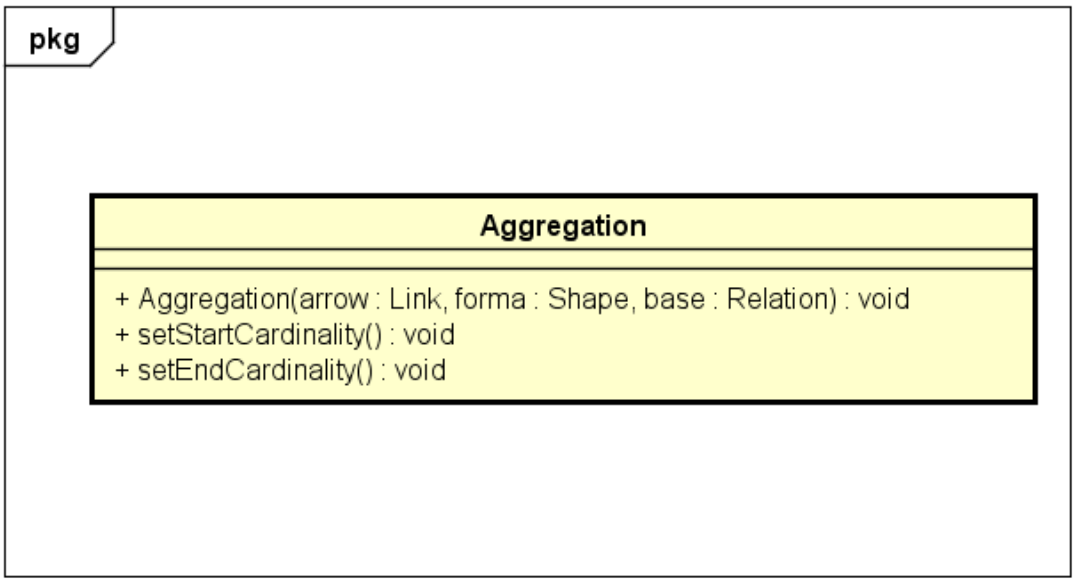

Figura 24: Front-end::Model::Objects::ClassObjects::Aggregation

- **Descrizione**:
  Questa classe rappresenta una relazione di tipo aggregazione del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti relazione di tipo aggregazione all'interno del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::Relation`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ Aggregation(arrow: Link, forma: Shape, base: Relation): void`
    si occupa di costruire la relazione di tipo aggregazione del diagramma delle classi a partire dalla classe base astratta Relation.
    Parametri:
    - `arrow: Link`
      rappresenta una relazione del diagramma delle classi. Il tipo Link è definito dalla libreria GoJS.

    * `forma: Shape`
      rappresenta la forma di una relazione del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.

    * `base: Relation`
      rappresenta un oggetto base comune a tutte le relazioni del diagramma delle classi.

  – `+ setStartCardinality(): void`
    si occupa di modificare la cardinalità di partenza della relazione aggregazione del diagramma delle classi.

  – `+ setEndCardinality(): void`
    si occupa di modificare la cardinalità di fine della relazione aggregazione del diagramma delle classi.

3.1.5.2.8 Composition

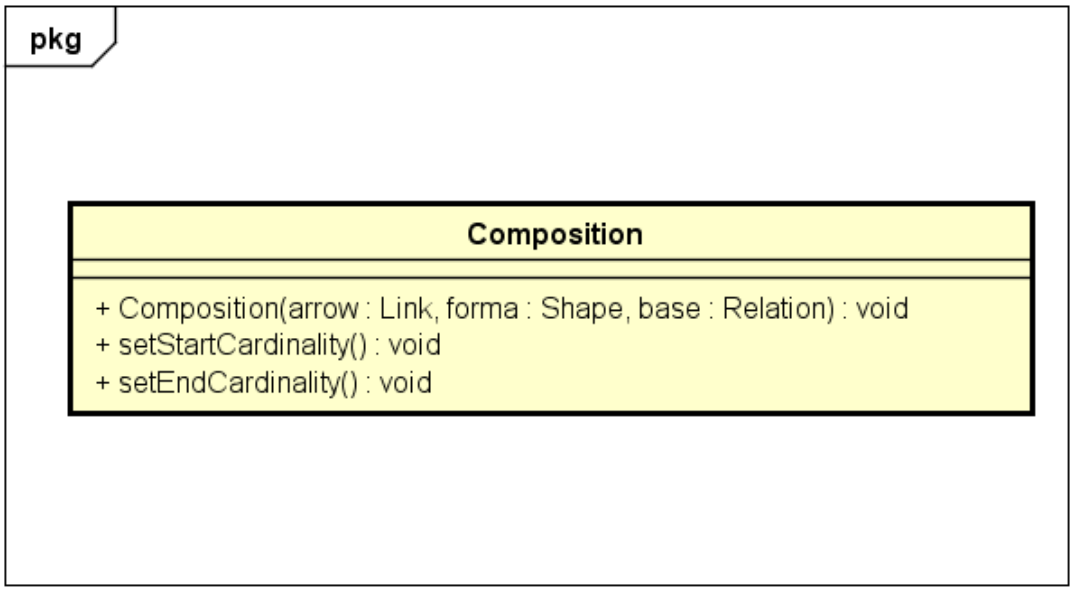

Figura 25: Front-end::Model::Objects::ClassObjects::Composition

- **Descrizione**:
  Questa classe rappresenta una relazione di tipo composizione del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti relazione di tipo composizione all'interno del diagramma delle classi. Utilizza la libreria GoJS.

- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::Relation`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ Composition(arrow: Link, forma: Shape, base: Relation): void`
    si occupa di costruire la relazione di tipo composizione del diagramma delle classi a partire dalla classe base astratta Relation.
    Parametri:
    - `arrow: Link`
      rappresenta una relazione del diagramma delle classi. Il tipo Link è definito dalla libreria GoJS.
    - `forma: Shape`
      rappresenta la forma di una relazione del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.
    - `base: Relation`
      rappresenta un oggetto base comune a tutte le relazioni del diagramma delle classi.
  - `+ setStartCardinality(): void`
    si occupa di modificare la cardinalità di partenza della relazione composizione del diagramma delle classi.
  - `+ setEndCardinality(): void`
    si occupa di modificare la cardinalità di fine della relazione composizione del diagramma delle classi.

3.1.5.2.9 Generalization

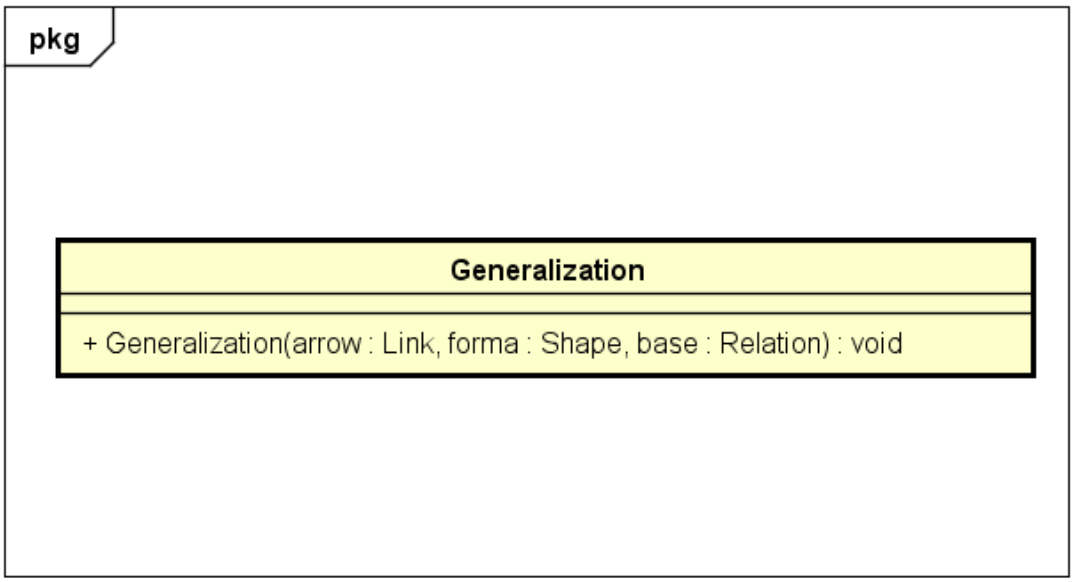

Figura 26: Front-end::Model::Objects::ClassObjects::Genralization

- **Descrizione**:
  Questa classe rappresenta una relazione di tipo generalizzazione del diagramma delle classi.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti relazione di tipo generalizzazione all'interno del diagramma delle classi. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ClassObjects::Relation`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ Generalization(arrow: Link, forma: Shape, base: Relation): void`
    si occupa di costruire la relazione di tipo generalizzazione del diagramma delle classi a partire dalla classe base astratta Relation.
    Parametri:
    - `arrow: Link`
      rappresenta una relazione del diagramma delle classi. Il tipo Link è definito dalla libreria GoJS.

* `forma: Shape`
  rappresenta la forma di una relazione del diagramma delle classi. Il tipo Shape è definito dalla libreria GoJS.

* `base: Relation`
  rappresenta un oggetto base comune a tutte le relazioni del diagramma delle classi.

3.1.6 Front-end::Model::Objects::ActivityObjects

3.1.6.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene tutti i blocchi che vanno a comporre il diagramma delle attività.
- **Framework esterni**:
  - GoJS

3.1.6.2 Classi

3.1.6.2.1 ActivityDiaObj

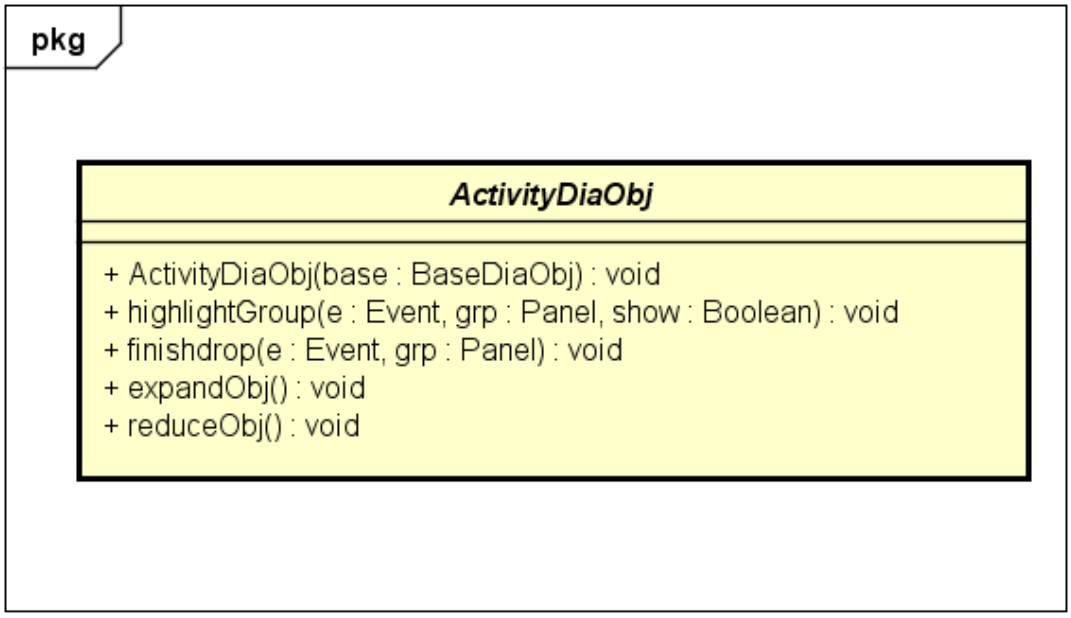

Figura 27: Front-end::Model::Objects::ActivityObjects::ActivityDiaObj

- **Descrizione**:
  Questa classe astratta rappresenta un contratto comune tra le classi che rappresentano i blocchi del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per rappresentare un oggetto base comune a tutti i blocchi del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::BaseDiaObj`
- **Sottoclassi**:
  - `Front-end::Model::Objects::ActivityObjects::VariableObj`
  - `Front-end::Model::Objects::ActivityObjects::MethodCallObj`
  - `Front-end::Model::Objects::ActivityObjects::CycleObj`
  - `Front-end::Model::Objects::ActivityObjects::IfElseObj`
  - `Front-end::Model::Objects::ActivityObjects::OperatorObj`
  - `Front-end::Model::Objects::ActivityObjects::StepObj`
  - `Front-end::Model::Objects::ActivityObjects::JollyObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ ActivityDiaObj(base: BaseDiaObj): void`
    si occupa di costruire la classe ActivityDiaObj vista come un contratto comune tra le classi che rappresentano i blocchi del diagramma delle attività partendo dalla classe base astratta BaseDiaObj.
    Parametri:
    * `base: BaseDiaObj`
      rappresenta un oggetto base comune ai diagrammi delle classi e delle attività.
  - `+ highlightGroup(e: Event, grp: Panel, show: Boolean): void`
    si occupa di di illuminare l'area del blocco nel quale viene inserito un altro blocco del diagramma delle attività.
    Parametri:
    * `e: Event`
      rappresenta un evento che si verifica nel diagramma delle attività. Il tipo Event è definito dalla libreria GoJS.

    * `grp: Panel`
      rappresenta una componente del blocco. Il tipo Panel è definito dalla libreria GoJS.

    * `show: Boolean`
      rappresenta un booleano che è legato agli eventi mouseDragEnter e mouseDragLeave. Tali eventi sono definiti dalla libreria GoJS.

- `+ finishdrop(e: Event, grp: Panel): void`
  si occupa di inserire un blocco all'interno di un altro blocco del diagramma delle attività.
  Parametri:

    * `e: Event`
      rappresenta un evento che si verifica nel diagramma delle attività. Il tipo Event è definito dalla libreria GoJS.

    * `grp: Panel`
      rappresenta una componente del blocco. Il tipo Panel è definito dalla libreria GoJS.

    * `show: Boolean`
      rappresenta un booleano che è legato agli eventi mouseDragEnter e mouseDragLeave. Tali eventi sono definiti dalla libreria GoJS.

- `+ expandObj(): void`
  si occupa di espandere un blocco del diagramma delle attività.

- `+ reduceObj(): void`
  si occupa di ridurre un blocco del diagramma delle attività.

3.1.6.2.2 VariableObj

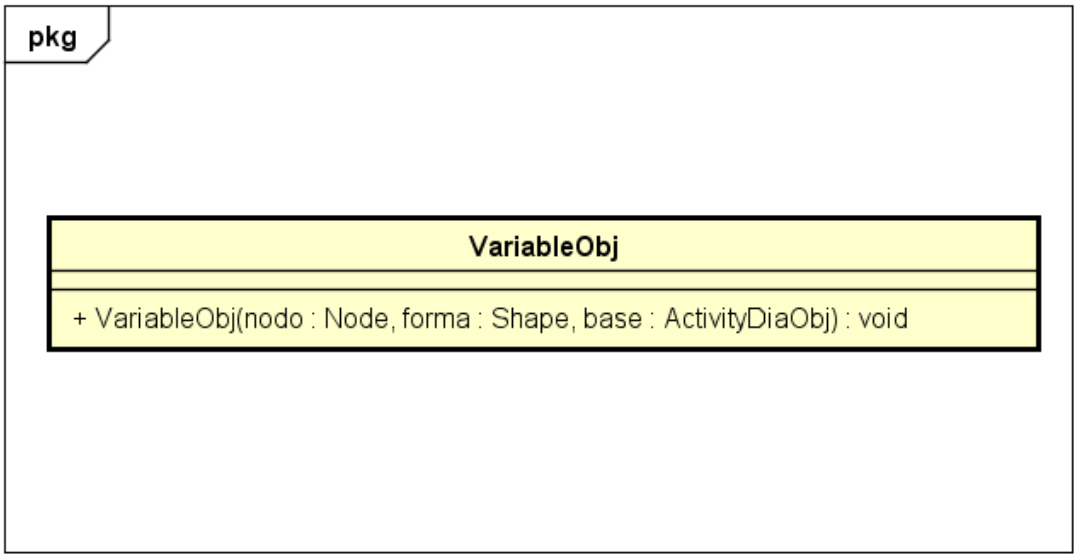

Figura 28: Front-end::Model::Objects::ActivityObjects::VariableObj

- **Descrizione**:
  Questa classe rappresenta un blocco variabile del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti variabile all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`
- **Sottoclassi**:
  - `Front-end::Model::Objects::ActivityObjects::ExistingVariableObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ VariableObj(nodo: Node, forma: Shape, base: ActivityDiaObj): void`
    si occupa di costruire il blocco variabile del diagramma delle attività partendo dalla classe base astratta ActivityDiaObj.
    Parametri:
    - `nodo: Node`
      rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.

* `forma: Shape`
  rappresenta la forma di un blocco del diagramma delle attività. Il tipo Shape è definito dalla libreria GoJS.
* `base: ActivityDiaObj`
  rappresenta un oggetto base comune a tutti i blocchi del diagramma delle attività.

3.1.6.2.3 ExistingVariableObj

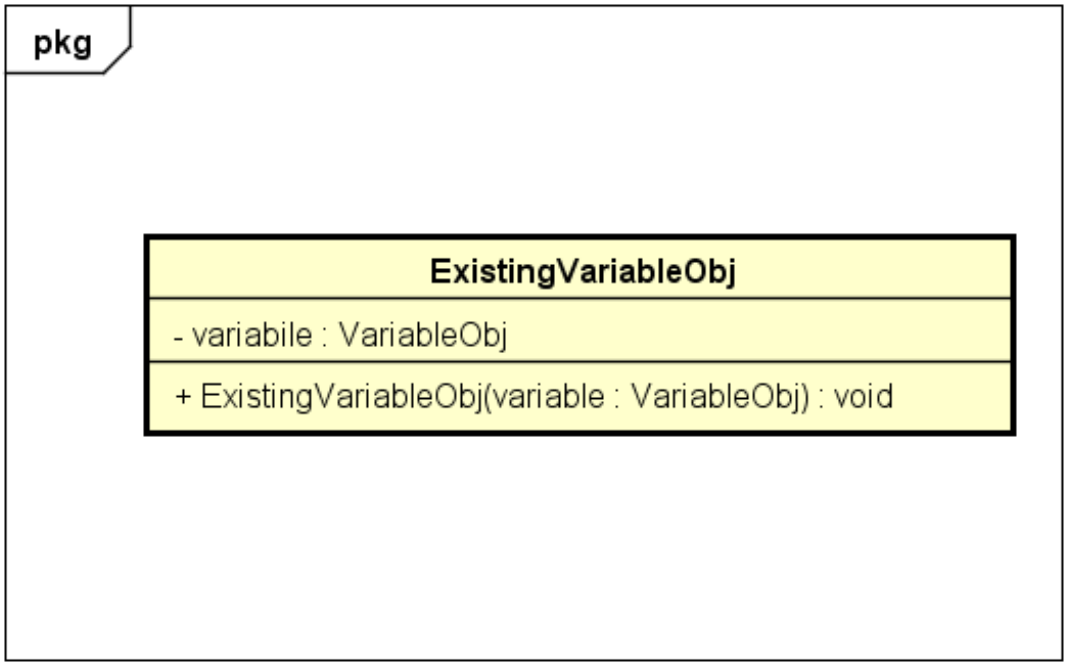

Figura 29: Front-end::Model::Objects::ActivityObjects::ExistingVariableObj

- **Descrizione**:
  Questa classe rappresenta un blocco variabile esistente del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti variabile esistente all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::VariableObj`
- **Attributi**:

  - `- variabile: VariableObj`
    si riferisce ad un oggetto della classe VariableObj perché rappresenta una variabile che è già stata definita.

- `+ ExistingVariableObj(variable: VariableObj): void`
  si occupa di costruire il blocco variabile esistente partendo dalla classe VariableObj.
  Parametri:

  - `variabile: VariableObj`
    rappresenta un oggetto della classe VariableObj.

3.1.6.2.4 MethodCallObj

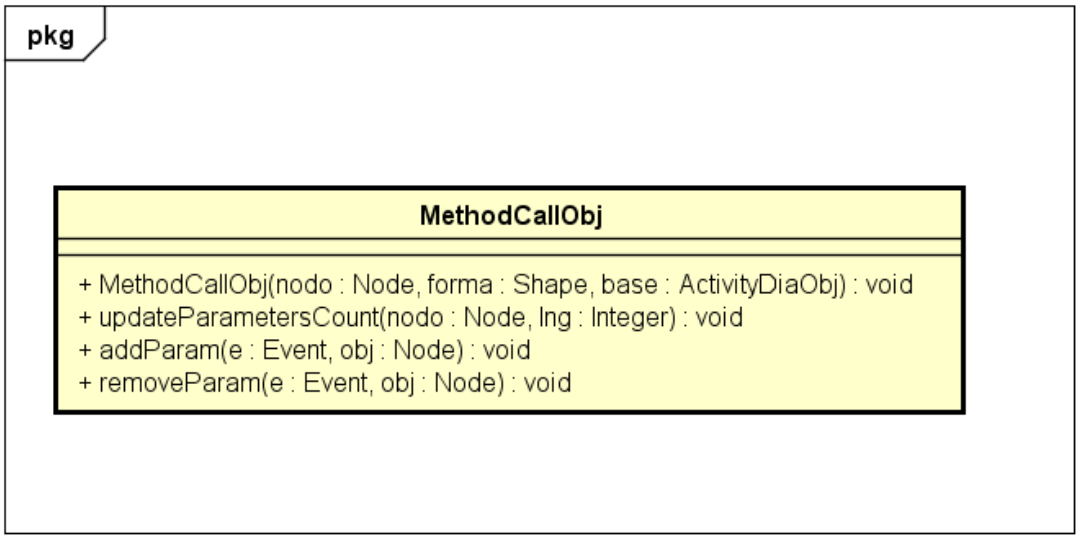

Figura 30: Front-end::Model::Objects::ActivityObjects::MethodCallObj

- **Descrizione**:
  Questa classe rappresenta un blocco chiamata del metodo del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti chiamata del metodo all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`
- **Attributi**: Assenti.
- **Metodi**:

– `+ MethodCallObj(nodo: Node, forma:Shape, base: ActivityDiaObj): void`
si occupa di costruire il blocco chiamatametodo del diagramma delle attività partendo dalla classe base astratta ActivityDiaObj.
Parametri:

  * `nodo: Node`
  rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.

  * `forma: Shape`
  rappresenta la forma di un blocco del diagramma delle attività. Il tipo Shape è definito dalla libreria GoJS.

  * `base: ActivityDiaObj`
  rappresenta un oggetto base comune a tutti i blocchi del diagramma delle attività.

– `+ updateParametersCount(nodo: Node, lng: Integer): void`
si occupa di aggiornare il contatore della lista di parametri nel blocco chiamatametodo.
Parametri:

  * `nodo: Node`
  rappresenta un nodo di tipo nodo. Il tipo Node è definito dalla libreria GoJS.

  * `lng: Integer`
  rappresenta la lunghezza dell'array della lista dei parametri del blocco chiamatametodo.

– `+ addParam(e: Event, obj: Node): void`
si occupa di aggiungere una variabile al blocco chiamatametodo del diagramma delle attività.
Parametri:

  * `e: Event`
  rappresenta un evento che si verifica nel diagramma delle attività. Il tipo Event è definito dalla libreria GoJS.

  * `obj: Node`
  si riferisce a un nodo di tipo Node. Il tipo Node è definito dalla libreria GoJS.

– `+ removeParam(e: Event, obj: Node): void`
si occupa di rimuovere una variabile dal blocco chiamatametodo del diagramma delle attività.
Parametri:

* `e: Event`
  rappresenta un evento che si verifica nel diagramma delle attività. Il tipo Event è definito dalla libreria GoJS.

* `obj: Node`
  si riferisce a un nodo di tipo Node. Il tipo Node è definito dalla libreria GoJS.

3.1.6.2.5 CycleObj

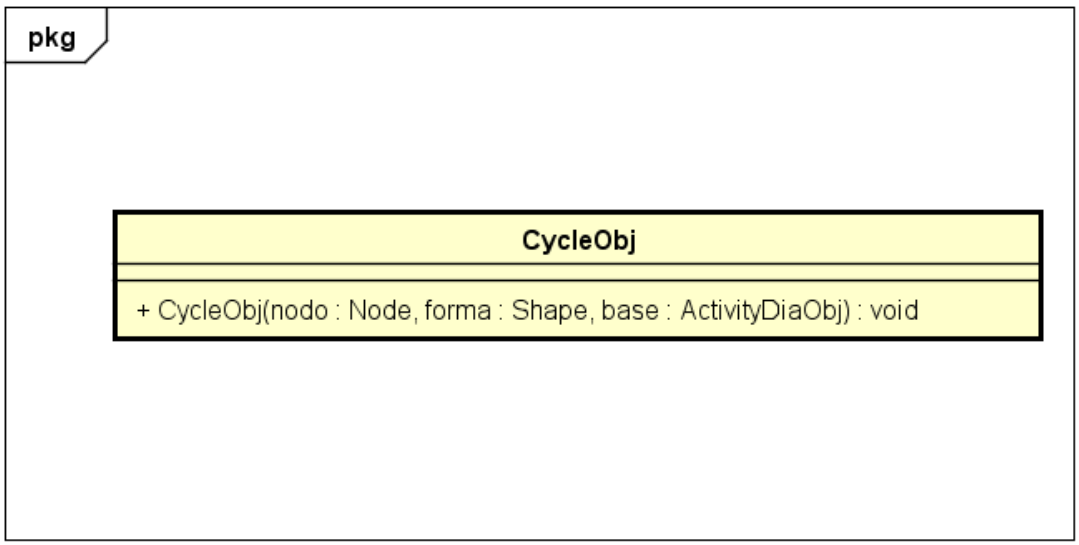

Figura 31: Front-end::Model::Objects::ActivityObjects::CycleObj

- **Descrizione**:
  Questa classe rappresenta un blocco ciclo del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti ciclo all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ CycleObj(nodo: Node, forma: Shape, base: ActivityDiaObj): void`
    si occupa di costruire il blocco ciclo del diagramma delle attività partendo dal-

la classe base astratta ActivityDiaObj.
Parametri:

* `nodo: Node`
  rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.
* `forma: Shape`
  rappresenta la forma di un blocco del diagramma delle attività. Il tipo Shape è definito dalla libreria GoJS.
* `base: ActivityDiaObj`
  rappresenta un oggetto base comune a tutti i blocchi del diagramma delle attività.

3.1.6.2.6 IfElseObj

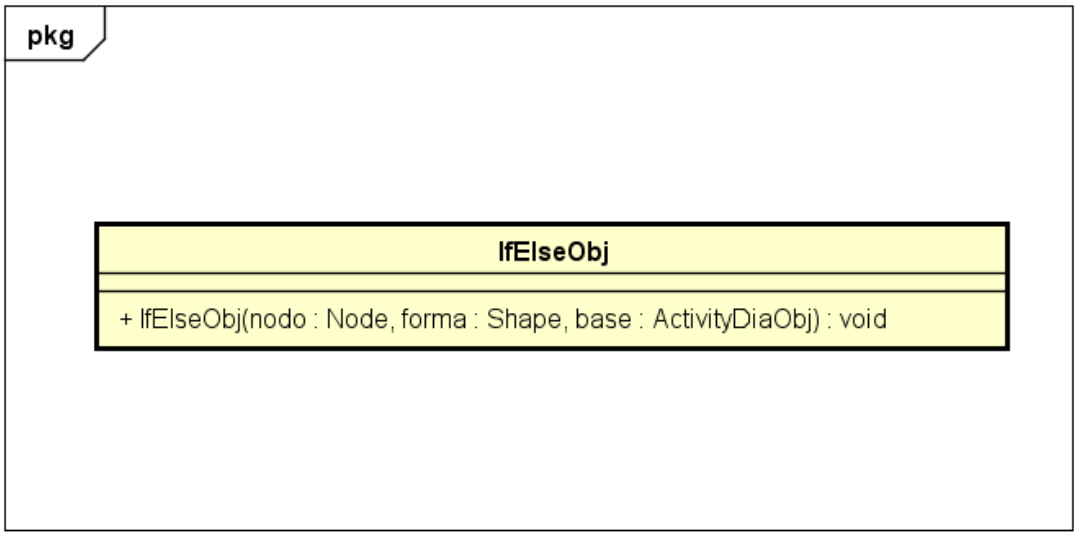

Figura 32: Front-end::Model::Objects::ActivityObjects::IfElseObj

- **Descrizione**:
  Questa classe rappresenta un blocco if/else del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti if/else all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`

- **Attributi**: Assenti.
- **Metodi**:
  - `+ IfElseObj(nodo: Node, forma: Shape, base: ActivityDiaObj): void` si occupa di costruire il blocco if/else del diagramma delle attività partendo dalla classe base astratta ActivityDiaObj.
    Parametri:
    * `nodo: Node`
      rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.
    * `forma: Shape`
      rappresenta la forma di un blocco del diagramma delle attività. Il tipo Shape è definito dalla libreria GoJS.
    * `base: ActivityDiaObj`
      rappresenta un oggetto base comune a tutti i blocchi del diagramma delle attività.

3.1.6.2.7 OperatorObj

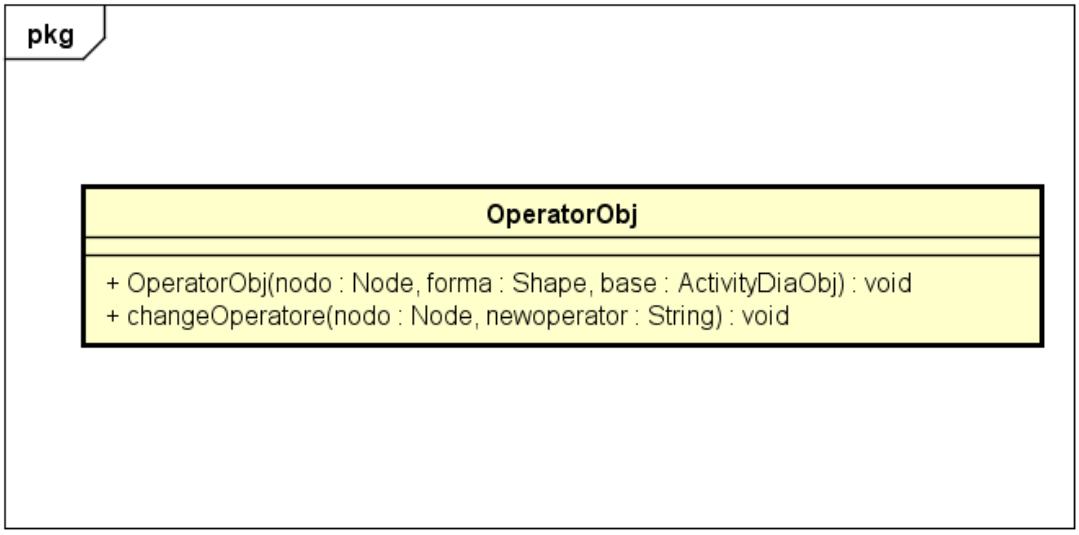

Figura 33: Front-end::Model::Objects::ActivityObjects::OperatorObj

- **Descrizione**:
  Questa classe rappresenta un blocco operatore del diagramma delle attività.

- **Utilizzo**:
  Viene utilizzata per istanziare oggetti operatore all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ OperatorObj(nodo: Node, forma: Shape, base: ActivityDiaObj): void`
    si occupa di costruire il blocco operatore del diagramma delle attività partendo dalla classe base astratta ActivityDiaObj.
    Parametri:
    * `nodo: Node`
      rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.
    * `forma: Shape`
      rappresenta la forma di un blocco del diagramma delle attività. Il tipo Shape è definito dalla libreria GoJS.
    * `base: ActivityDiaObj`
      rappresenta un oggetto base comune a tutti i blocchi del diagramma delle attività.
  - `+ changeOperatore(nodo: Node, newoperator: String): void`
    si occupa di modificare il tipo di operatore del blocco operatore del diagramma delle attività.
    Parametri:
    * `nodo: Node`
      rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.
    * `newoperator: String`
      rappresenta il tipo di operatore da inserire.

3.1.6.2.8 StepObj

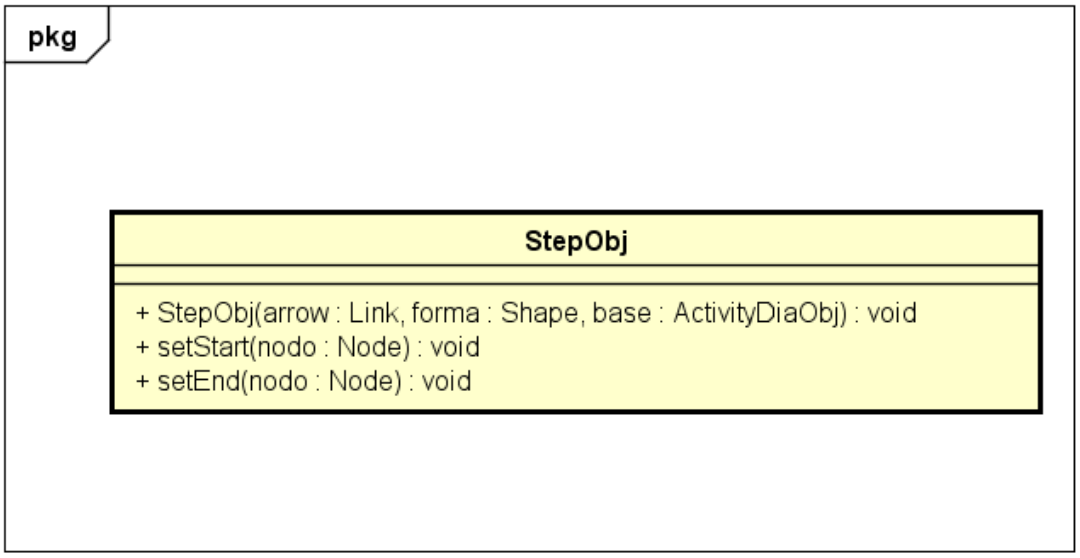

Figura 34: Front-end::Model::Objects::ActivityObjects::StepObj

- **Descrizione**:
  Questa classe rappresenta un blocco avanzamento del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti avanzamento all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ StepObj(arrow: Link, forma: Shape, base: ActivityDiaObj): void`
    si occupa di costruire il blocco Step del diagramma delle attività partendo dalla classe base astratta ActivityDiaObj.
    Parametri:
    * `arrow: Link`
      rappresenta una relazione del diagramma delle attività. Il tipo Link è definito dalla libreria GoJS.

    * `forma: Shape`
      rappresenta la forma di un blocco del diagramma delle attività. Il tipo Shape è definito dalla libreria GoJS.

    * `base: ActivityDiaObj`
      rappresenta un oggetto base comune a tutti i blocchi del diagramma delle attività.

- `+ setStart(nodo: Node): void`
  si occupa di selezionare il nodo di partenza al quale agganciare la relazione del diagramma delle attività.
  Parametri:

    * `nodo: Node`
      rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.

- `+ setEnd(nodo: Node): void`
  si occupa di selezionare il nodo di fine al quale agganciare la relazione del diagramma delle attività.
  Parametri:

    * `nodo: Node`
      rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.

3.1.6.2.9 JollyObj

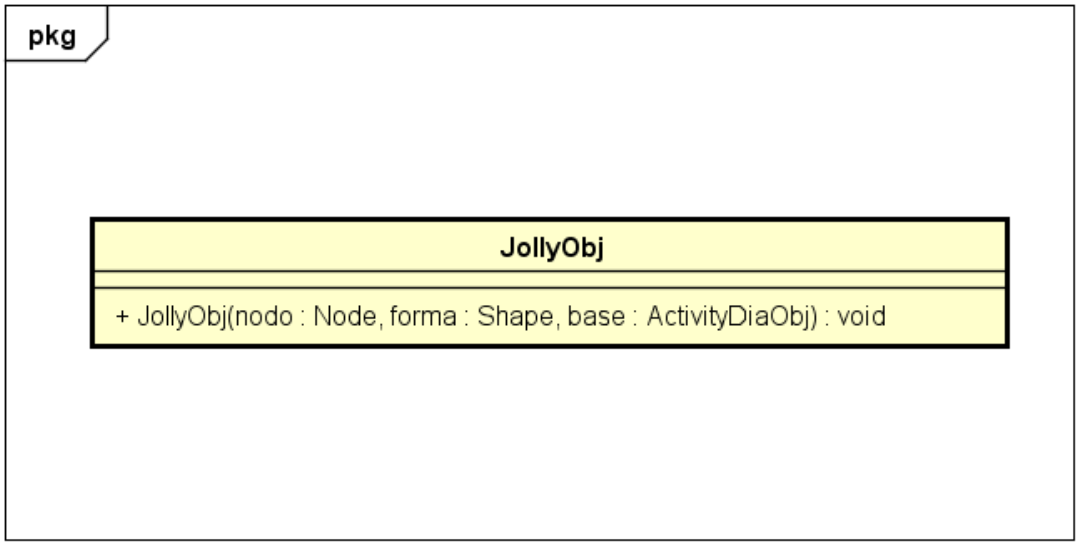

Figura 35: Front-end::Model::Objects::ActivityObjects::JollyObj

- **Descrizione**:
  Questa classe rappresenta un blocco jolly del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per istanziare oggetti jolly all'interno del diagramma delle attività. Utilizza la libreria GoJS.
- **Classi ereditate**:
  - `Front-end::Model::Objects::ActivityObjects::ActivityDiaObj`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ JollyObj(nodo: Node, forma:Shape, base: ActivityDiaObj): void`
    si occupa di costruire il blocco Jolly del diagramma delle attività partendo dalla classe base astratta BaseDiaObj.
    Parametri:
    - `nodo: Node`
      rappresenta un blocco del diagramma delle attività. Il tipo Node è definito dalla libreria GoJS.
    - `forma: Shape`
      rappresenta la forma di un blocco del diagramma delle attività. Il tipo Shape è definito dalla libreria GoJS.

* `base: ActivityDiaObj`
  rappresenta un oggetto base comune a tutti i blocchi del diagramma delle attività.

3.1.7 Front-end::Model::Commands

3.1.7.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene i comandi relativi al progetto di richiesta codice e richiesta template.
- **Framework esterni**:
  - Angular2

3.1.7.2 Classi

3.1.7.2.1 Command

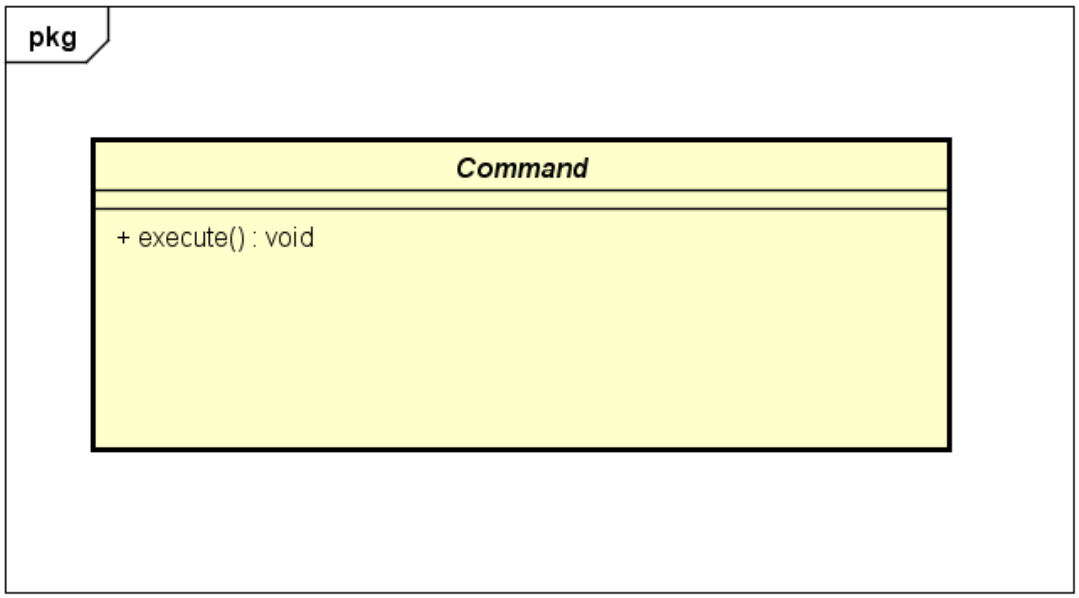

Figura 36: Front-end::Model::Commands::Command

- **Descrizione**:
  Questa classe astratta rappresenta un contratto comune per l'esecuzione di un comando che viene chiamato dalle classi `Front-end::ViewModel::PaletteCommandController`

o `Front-end::ViewModel::CodeCommandController` quando l'utente decide di richiedere un template o la generazione di codice. Rappresenta il Command del design pattern Command.

- **Utilizzo**:
  Viene utilizzata dalle classi `Front-end::ViewModel::PaletteCommandController` o `Front-end::ViewModel::CodeCommandController` che rappresentano l'invoker del design pattern Command, che chiederà al `Front-end::Model::Commands::Command` di eseguire le richieste in base all'input dell'utente.
- **Relazioni con altre classi**:
  - IN: `Front-end::ViewModel::PaletteCommandController`: utilizza `Front-end::Model::Commands::Command` per inoltrare le azioni dell'utente quando decide di richiedere un template;
  - IN: `Front-end::ViewModel::CodeCommandController`: utilizza `Front-end::Model::Commands::Command` per inoltrare le azioni dell'utente quando decide di richiedere la generazione di codice.
- **Sottoclassi**:
  - `Front-end::Model::Commands::RequestCode`
  - `Front-end::Model::Commands::GetTemplate`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ execute()`
    si occupa di eseguire il comando relativo alle classi `Front-end::Model::Commands::RequestCode` e `Front-end::Model::Commands::GetTemplate`. Non presenta un tipo di ritorno in quanto quest'ultimo dipende dalle due sottoclassi.

3.1.7.2.2 RequestCode

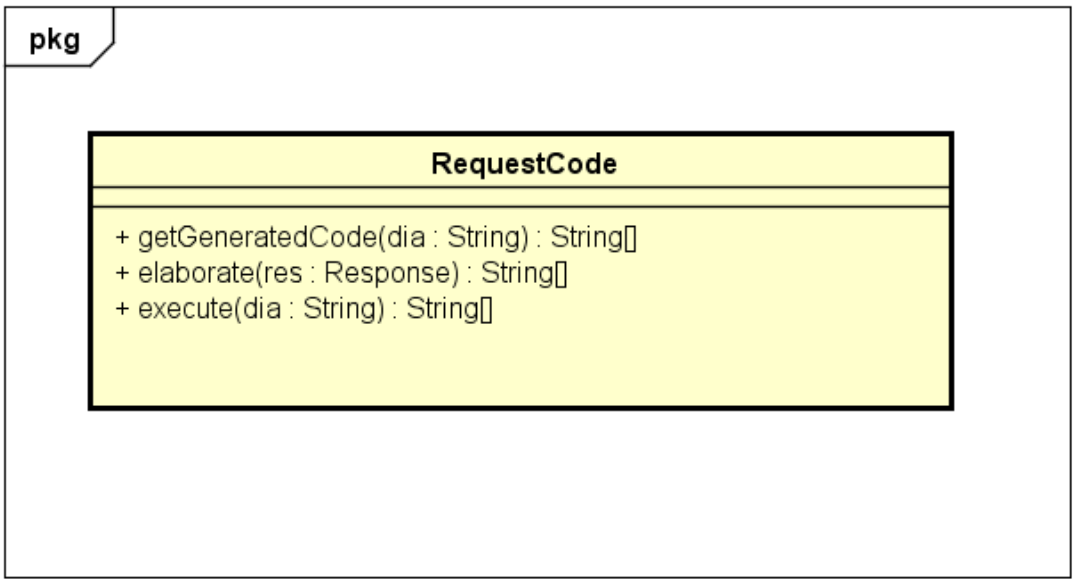

Figura 37: Front-end::Model::Commands::RequestCode

- **Descrizione**:
  Questa classe rappresenta un comando di richiesta generazione codice del progetto richiesto dall'utente. Rappresenta il Concrete Command del design pattern Command.
- **Utilizzo**:
  Viene utilizzata da `Front-end::ViewModel::CodeCommandController`, per mezzo della classe `Front-end::Model::Commands::Command` per le operazioni legate alla richiesta di generazione del codice del progetto.
- **Relazioni con altre classi**:
  - OUT: `Front-end::Model::Services::ServerConnector`: necessaria per ottenere la generazione di codice del progetto come richiesto dall'utente.
- **Classi ereditate**:
  - `Front-end::Model::Commands::Command`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedCode(dia: String): String[]`
    riceve una stringa contenente il diagramma e ritorna un array di stringhe

contenente il codice.
Parametri:

* `dia: String`
  contiene il diagramma di cui si vuole generare il codice.

– `+ elaborate(res: Response): String[]`
elabora la risposta ricevuta dal metodo `getGeneratedCode(file: Model)` presente in `Front-end::Model::Services::ServerConnector` e ritorna un array di stringhe contenente il codice.
Parametri:

* `res: Response`
  contiene il codice generato.

– `+ execute(dia: String): String[]`
viene invocata per eseguire il comando relativo alla generazione di codice mediante la chiamata alla funzione `getGeneratedCode(dia: String)`.
Parametri:

* `dia: String`
  contiene il diagramma di cui si vuole generare il codice.

3.1.7.2.3 GetTemplate

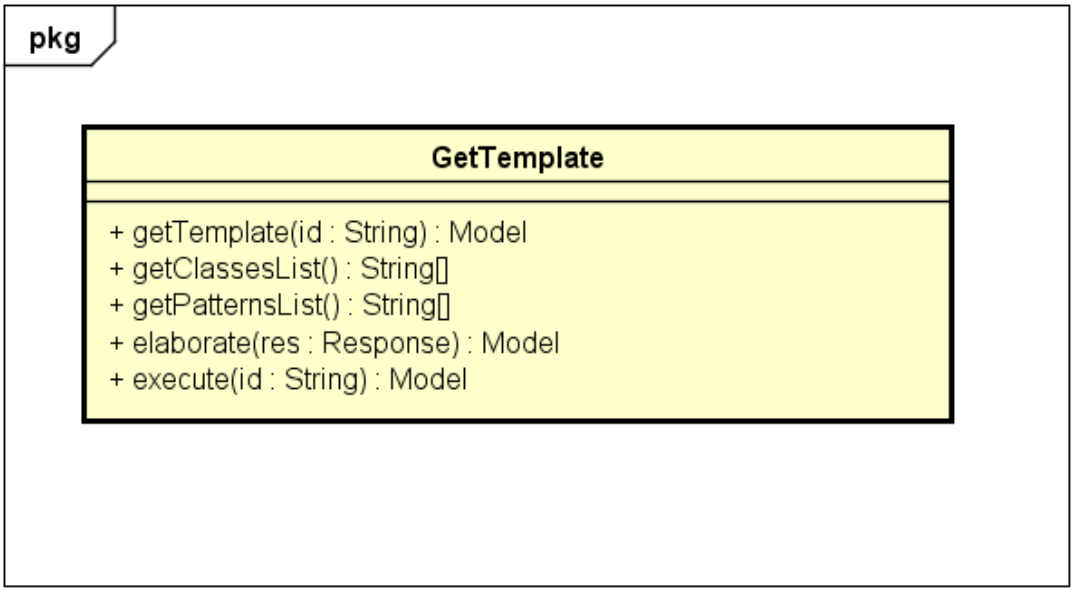

Figura 38: Front-end::Model::Commands::GetTemplate

- **Descrizione**:
  Questa classe rappresenta un comando di richiesta template richiesto dall'utente. Rappresenta il Concrete Command del design pattern Command.
- **Utilizzo**:
  Viene utilizzata da `Front-end::ViewModel::PaletteCommandController`, per mezzo della classe `Front-end::Model::Commands::Command` per le operazioni legate alla richiesta di template.
- **Classi ereditate**:
  - `Front-end::Model::Commands::Command`
- **Relazioni con altre classi**:
  - OUT: `Front-end::Model::Services::ServerConnector`: necessaria per ottenere un template richiesto dall'utente.
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getTemplate(id: String): Model`
    richiede la stringa dell'id del template di cui vuole restituire il JSON rappresentante un oggetto GoJs di tipo Model.
    Parametri:
    * `id: String`
      contiene l'id del template richiesto.
  - `+ getClassesList(): String[]`
    restituisce la lista dei template di classi contenuta nella libreria.
  - `+ getPatternsList(): String[]`
    restituisce la lista dei template dei pattern contenuta nella libreria.
  - `+ elaborate(res: Response): Model`
    elabora la risposta ricevuta dal metodo `getTemplate(id: String)` presente in `Front-end::Model::Services::ServerConnector` contenente il template richiesto restituendolo sotto forma di JSON rappresentante un oggetto GoJs di tipo Model.
    Parametri:
    * `res: Response`
      contiene il template richiesto.
  - `+ execute(id: String): Model`
    viene invocata per eseguire il comando relativo alla richiesta dei template mediante la chiamata alla funzione `getTemplate(id: String)`.
    Parametri:

* `id: String`
  contiene l'id del template richiesto.

3.1.8 Front-end::Model::Services

3.1.8.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene tutte le classi necessarie per gestire la comunicazione tra Front-end e *Back-end*$_G$.
- **Framework esterni**:
  - Angular2

3.1.8.2 Classi

3.1.8.2.1 ServerConnector

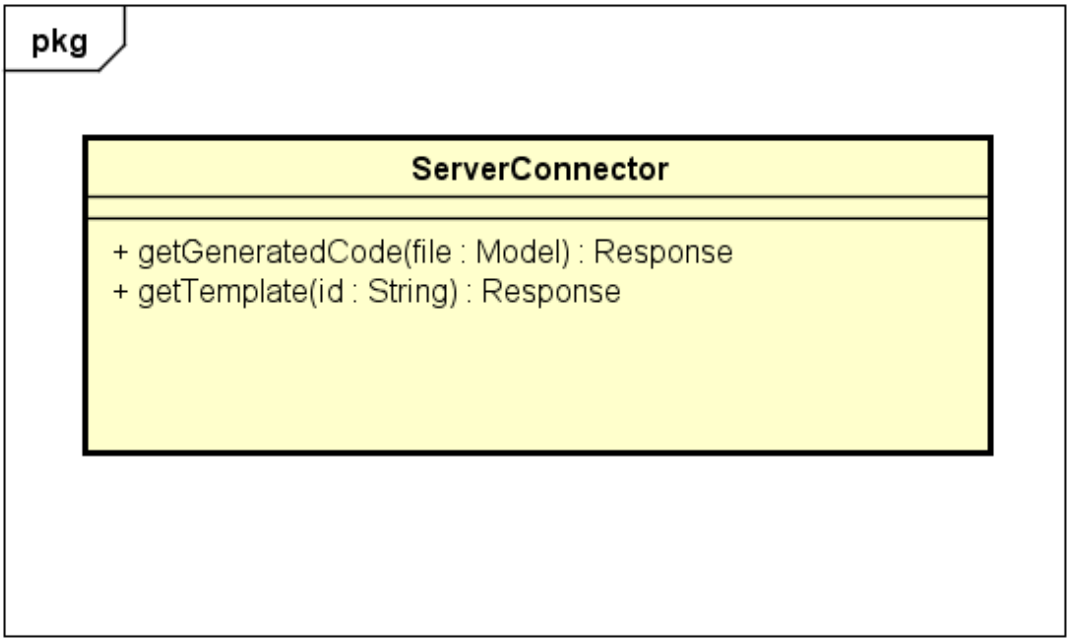

Figura 39: Front-end::Model::Services::ServerConnector

- **Descrizione**:
  Questa classe gestisce le comunicazioni con l'applicazione back-end, sfruttando le

funzionalità `http` offerte dal *framework*$_G$ Angular2. Rappresenta il Receiver del design pattern Command implementato in `Front-end::Model::Commands`.

- **Utilizzo**:
  Viene utilizzata per l'invio del file JSON rappresentante un oggetto GoJs di tipo Model contenente il progetto al `Back-end` e la ricezione del codice generato da quest'ultimo dopo aver effettuato la richiesta. Inoltre viene utilizzata per richiedere e ricevere i file JSON rappresentanti oggetti GoJs di tipo Model relativi ai template presenti nella libreria. Le richieste alla parte di back-end aderiscono ai principi *REST*$_G$.
- **Relazioni con altre classi**:
  - IN: `Front-end::Model::Commands::GetTemplate`: utilizza `Front-end::Model::Services::ServerConnector` per ottenere un template richiesto dall'utente;
  - IN: `Front-end::Model::Commands::RequestCode`: utilizza `Front-end::Model::Services::ServerConnector` per richiedere la generazione di codice del progetto come chiesto dall'utente.
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedCode(file: Model): Response`
    si occupa di inviare il file JSON rappresentante un oggetto GoJs di tipo Model al *server*$_G$ e quest'ultimo gli restituisce una risposta che contiene il codice, il tutto mediante chiamata REST.
    Parametri:
    - `file: Model`
      contiene il diagramma di cui si vuole ottenere il codice.
  - `+ getTemplate(id: String): Response`
    si occupa di richiedere al server un JSON rappresentante un oggetto GoJs di tipo Model che contiene il template richiesto.
    Parametri:
    - `id: String`
      contiene l'id associato ad un determinato template.

4 Back-end

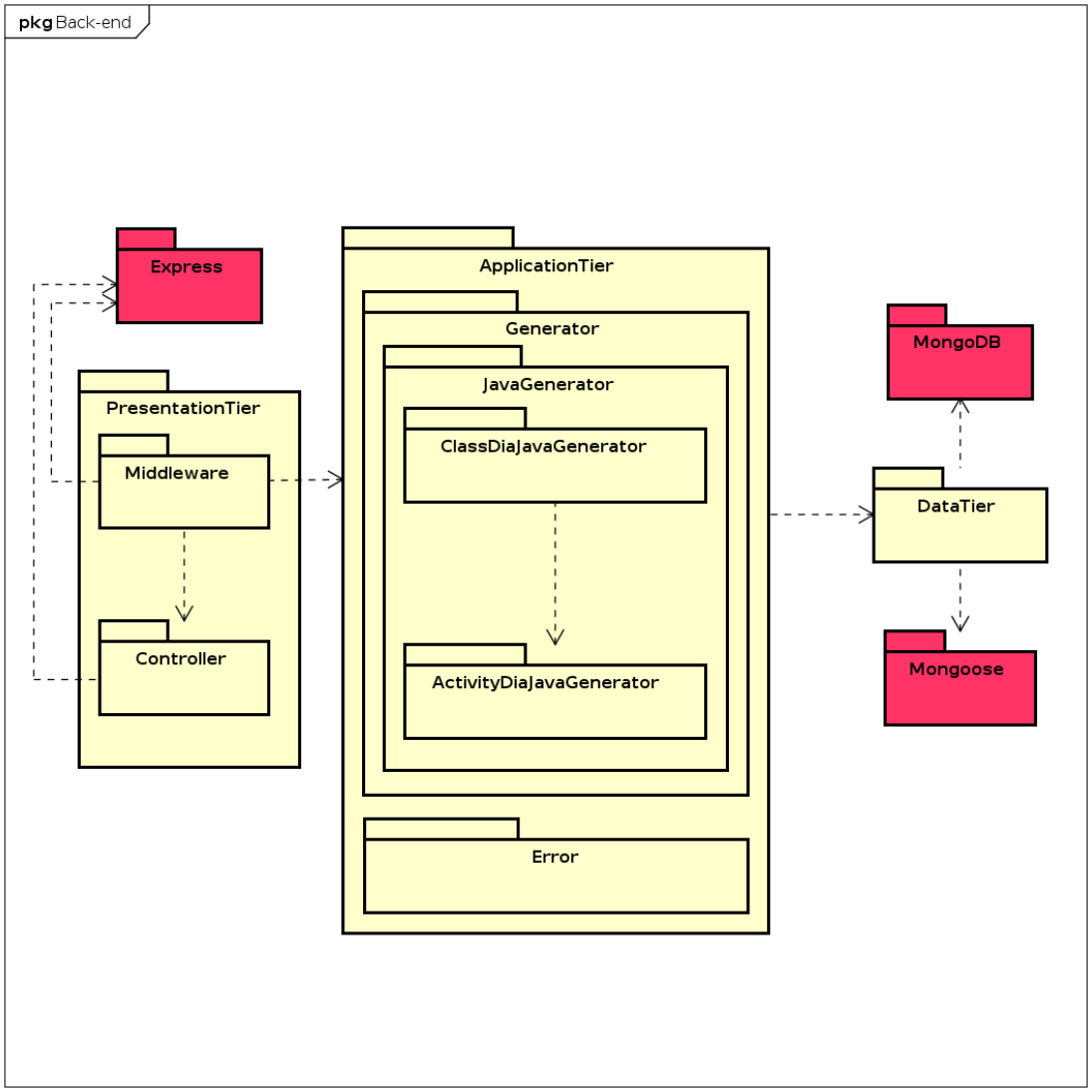

Figura 40: Diagramma dei package Back-end

4.1 Descrizione packages e classi

4.1.1 Back-end::PresentationTier

4.1.1.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene tutte le classi che gestiscono le richieste da parte del client. Costituisce la parte Presentation dell'architettura *Three-tier*$_G$ del back-end.
- **Package contenuti**:
  - `Back-end::PresentationTier::Middleware`
  - `Back-end::PresentationTier::Controller`
- **Framework esterni**:
  - Express

4.1.2 Back-end::PresentationTier::Middleware

4.1.2.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene le classi per la gestione delle comunicazioni tra server e client, e le classi per la gestione dei possibili errori nel momento di richiesta delle risorse.
- **Framework esterni**:
  - Express

4.1.2.2 Classi

4.1.2.2.1 MiddlewareLoader

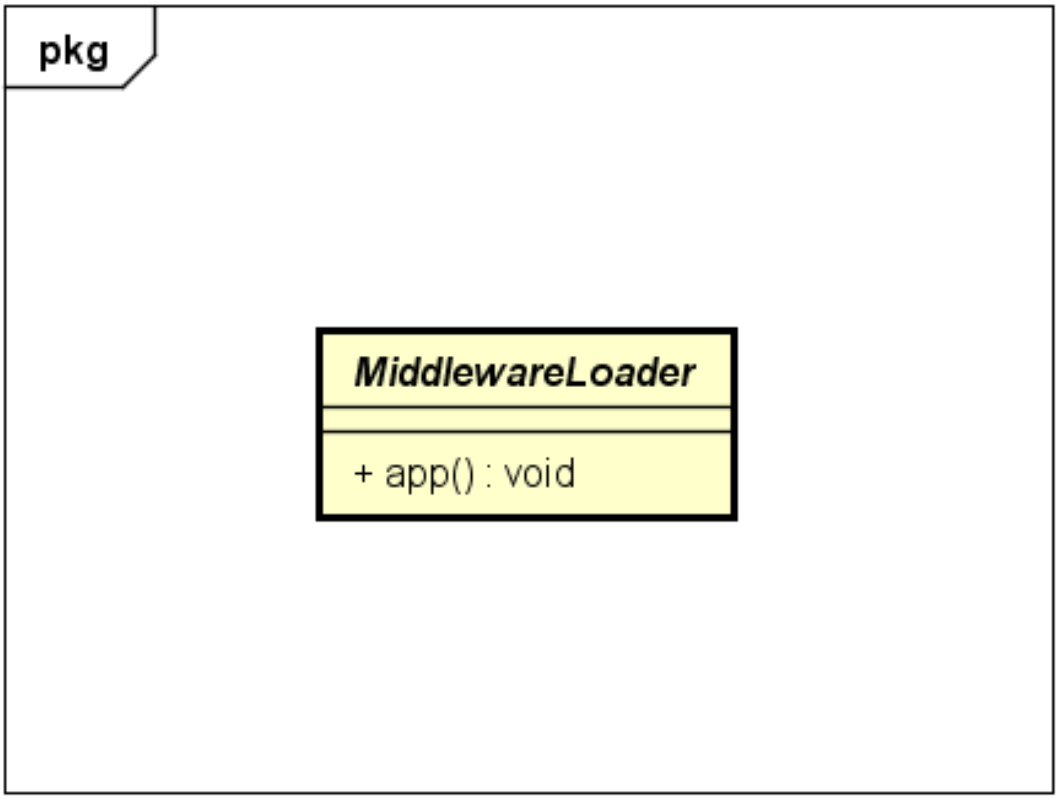

Figura 41: Back-end::PresentationTier::Middleware::MiddlewareLoader

- **Descrizione**:
  Questa classe astratta definisce un contratto comune per tutte le richieste REST provenienti dal client. È uno dei componenti ConcreteHandler del design pattern *Chain of responsibility*$_G$. Utilizza il framework Express.
- **Utilizzo**:
  Viene utilizzato per istanziare in modo nascosto all'applicazione tutti i *middleware*$_G$ presenti nel `Back-end::PresentationTier::Middleware`.
- **Sottoclassi**:
  - `Back-end::PresentationTier::Middleware::Router`
  - `Back-end::PresentationTier::Middleware::ErrorHandler`

  - `Back-end::PresentationTier::Middleware::NotFoundHandler`

- **Attributi**: Assenti.
- **Metodi**:
  - `+ app(): void`
    si occupa di inizializzare i vari componenti del middleware.

4.1.2.2.2 ErrorHandler

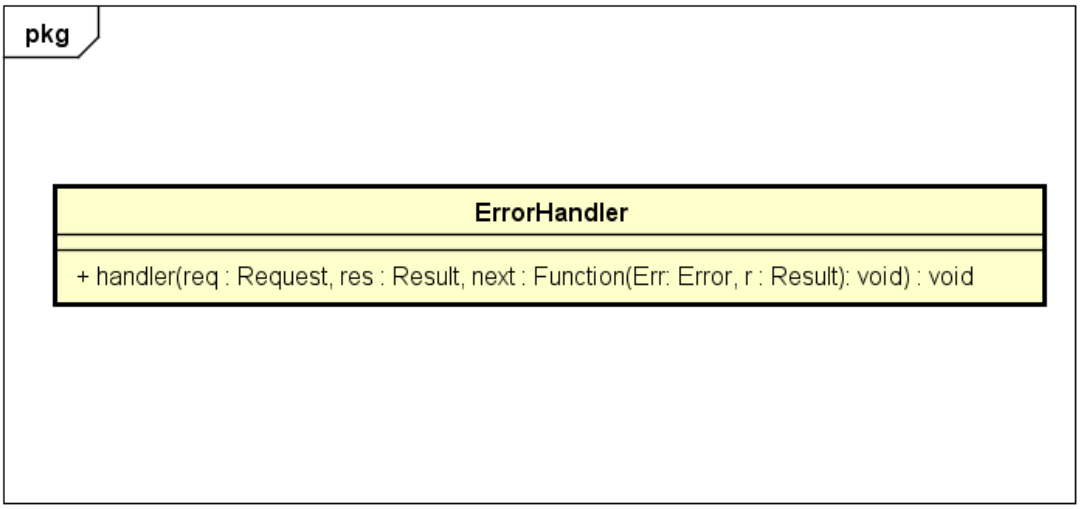

Figura 42: Back-end::PresentationTier::Middleware::ErrorHandler

- **Descrizione**:
  Questa classe gestisce gli errori generati nei precedenti middleware. Invia al client uno stato di risposta HTTP 500 (server error) con una descrizione dell'errore nel formato JSON. È uno dei componenti ConcreteHandler del Design Pattern Chain of responsibility.
- **Utilizzo**:
  Questo middleware viene utilizzato per ultimo nella catena di gestione delle richieste di Express, in modo da gestire tutti gli errori generati precedentemente.
- **Classi ereditate**:
  - `Back-end::PresentationTier::Middleware::MiddlewareLoader`
- **Attributi**: Assenti.
- **Metodi**:

- `+ handler(req: Request, res: Result, next: Function(Err: Error, r: Result): void): void`
  si occupa di gestire gli errori verificatisi precedentemente.
  Parametri:
  - `req: Request`
    rappresenta la richiesta arrivata al server che il metodo deve gestire;
  - `res: Result`
    rappresenta la risposta che il server ritorna al termine dell'elaborazione;
  - `next: Function(Err: Error, r: Result): void`
    rappresenta la $callback_G$ che il metodo chiamerà al termine dell'elaborazione per passare il controllo ai successivi middleware. La presenza del parametro facoltativo Err attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste. Il risutlato sarà ritornato dall'oggetto risposta r.

4.1.2.2.3 Router

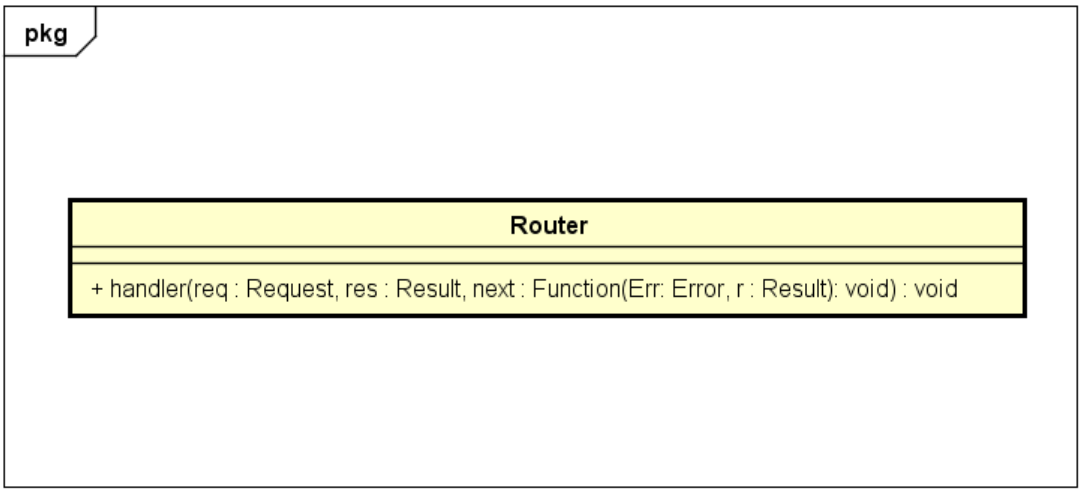

Figura 43: Back-end::PresentationTier::Middleware::Router

- **Descrizione**:
  Classe che si occupa della richiesta di risorse. È uno dei componenti Handler del Design Pattern Chain of responsibility.
- **Utilizzo**:
  Si occupa di smistare la richiesta in base all'URI ricevuto e ad invocare l'opportuno metodo sulla classe `Back-end::ApplicationTier::ApplicationController`. Utilizza il framework Express.

- **Relazioni con altre classi**:
  - OUT: `Back-end::PresentationTier::Services::IndexGiver`: necessaria per smistare le richieste provenienti dalla single page app;
  - OUT: `Back-end::ApplicationTier::ApplicationController`: necessaria per smistare le richieste ed invocare l'opportuno metodo di `Back-end::ApplicationTier::ApplicationController`.
- **Classi ereditate**:
  - `Back-end::PresentationTier::Middleware::MiddlewareLoader`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ handler(req: Request, res: Result, next: Function(err: Error, r: Result): void): void`
    si occupa di gestire le richieste di routing.
    Parametri:
    - `req: Request`
      rappresenta la richiesta arrivata al server che il metodo deve gestire;
    - `res: Result`
      rappresenta la risposta che il server ritorna al termine dell'elaborazione;
    - `Function(err: Error, r: Result): void`
      rappresenta la callback che il metodo chiamerà al termine dell'elaborazione per passare il controllo ai successivi middleware. La presenza del parametro facoltativo Err attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste. Il risutlato sarà ritornato dall'oggetto risposta r.

4.1.2.2.4 NotFoundHandler

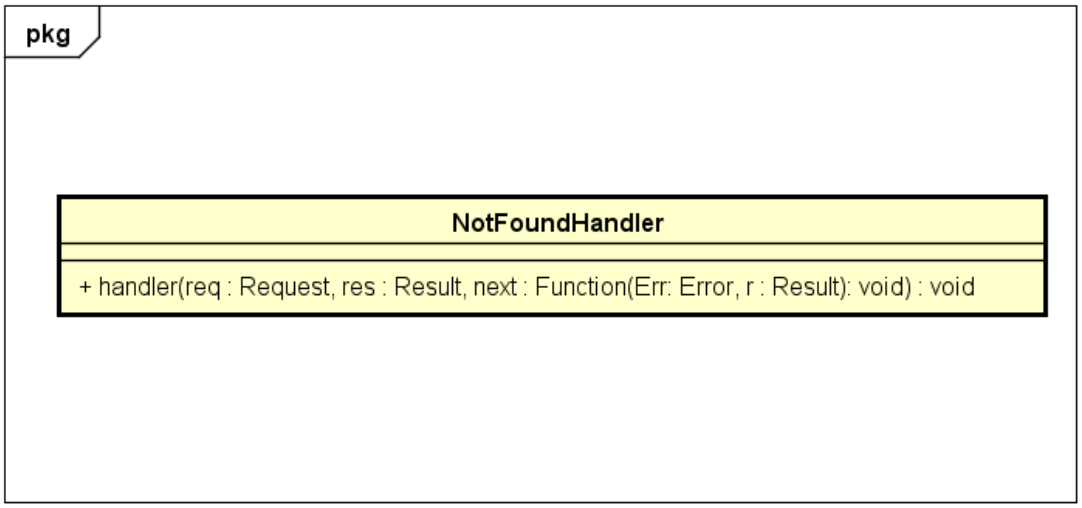

Figura 44: Back-end::PresentationTier::Middleware::NotFoundHandler

- **Descrizione**:
  Classe che si occupa della gestione dell'errore di pagina non trovata. È uno dei componenti ConcreteHandler del Design Pattern Chain of responsibility.
- **Utilizzo**:
  Viene utilizzata per generare una pagina 404 di errore nel caso in cui l'URI passato non corrisponda ad una risorsa presente nell'applicazione. Utilizza il framework Express.
- **Classi ereditate**:
  - `Back-end::PresentationTier::Middleware::MiddlewareLoader`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ handler(req: Request, res: Result,`
    `next: Function(Err: Error, r: Result):void): void`
    si occupa di gestire le situazioni in cui un determinato oggetto richiesto non venga trovato all'interno del server.
    Parametri:
    - `req: Request`
      rappresenta la richiesta arrivata al server che il metodo deve gestire;
    - `res: Result`
      rappresenta la risposta che il server ritorna al termine dell'elaborazione;

* `next: Function(Err: Error, r: Result): void`
  rappresenta la callback che il metodo chiamerà al termine dell'elaborazione per passare il controllo ai successivi middleware. La presenza del parametro facoltativo Err attiva la catena di gestione dell'errore in sostituzione della normale catena di gestione delle richieste. Il risutlato sarà ritornato dall'oggetto risposta r.

4.1.3 Back-end::PresentationTier::Controller

4.1.3.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene la classe che gestisce la richiesta della index-page.
- **Framework esterni**:
  - Express

4.1.3.2 Classi

4.1.3.2.1 IndexGiver

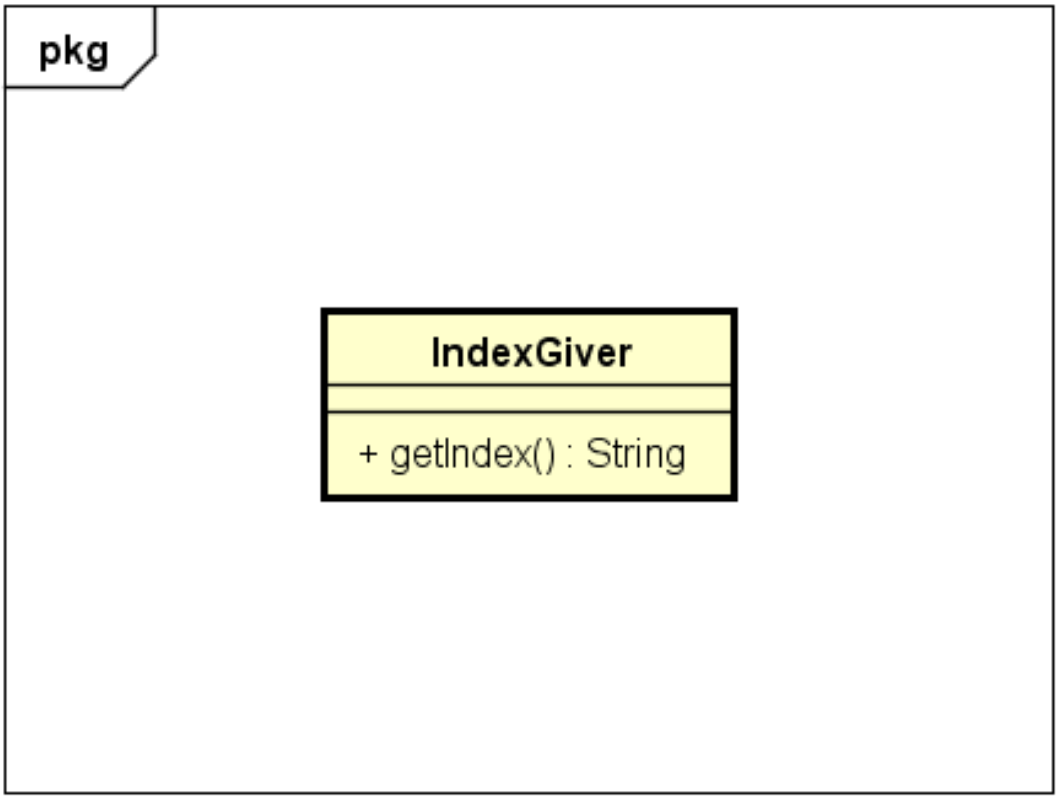

Figura 45: Back-end::PresentationTier::Controller::IndexGiver

- **Descrizione**:
  Questa classe gestisce la richiesta della single page da parte del client.
- **Utilizzo**:
  Viene utilizzata per fornire al client la single page prelevandola dal server. Utilizza il framework Express.
- **Relazioni con altre classi**:
  - IN: `Back-end::PresentationTier::Middleware::Router`: utilizza `Back-end::PresentationTier::Controller::IndexGiver` per fornirgli le richieste smistate provenienti dalla single page app.
- **Attributi**: Assenti.

- **Metodi**:
  - `+ getIndex(): String`
    si occupa di ritornare il file `index.html`.

4.1.4 Back-end::ApplicationTier

4.1.4.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene tutte le componenti inerenti la *business logic*$_G$ dell'applicazione Back-end, scritte in JavaScript.
- **Package contenuti**:
  - Back-end::Generator
  - Back-end::Error

4.1.4.2 Classi

4.1.4.2.1 ApplicationController

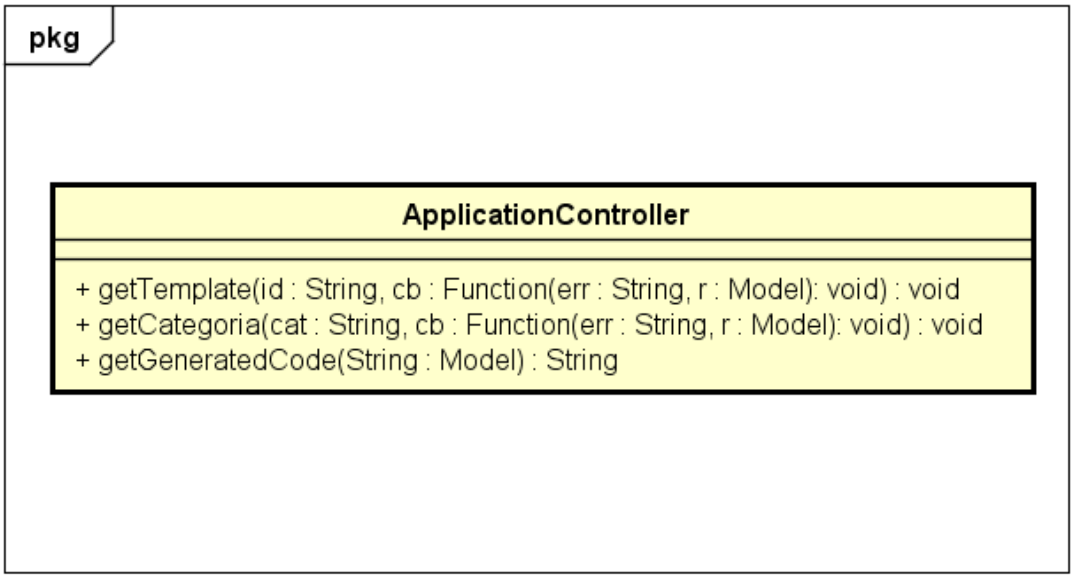

Figura 46: Back-end::ApplicationTier::ApplicationController

- **Descrizione**:
  Questa classe gestisce le comunicazioni tra i tre package della struttura three-tier.
- **Utilizzo**:
  Viene utilizzata per raccogliere le richieste del package `Back-end::PresentationTier` delegandole all'interfaccia `Back-end::ApplicationTier::Generator::BaseGenerator` per la generazione del codice e alla classe `Back-end::DataTier::Template` per la richiesta di dati dal $database_G$ $MongoDB_G$.
- **Relazioni con altre classi**:
  - IN: `Back-end::PresentationTier::Middleware::Router`: utilizza `Back-end::ApplicationTier::ApplicationController` per fornirgli le richieste smistate in base all'URI ed invocare l'opportuno metodo di quest'ultima;
  - OUT: `Back-end::ApplicationTier::Generator::BaseGenerator`: necessaria per invocare l'algoritmo di generazione codice;
  - OUT: `Back-end::DataTier::Template`: necessaria per gestire la richiesta di un template JSON.
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedCode(String: Model): String`
    si occupa di chiamare la funzione di generazione codice e passare il parametro contenente il JSON rappresentante un oggetto GoJs di tipo Model con le informazioni dei diagrammi creati dall'utente.
    Parametri:
    - `String: Model`
      contiene il JSON rappresentante un oggetto GoJs di tipo Model che rappresenta l'insieme di blocchi del diagramma delle classi e dei diagrammi delle attività.
  - `+ getTemplate(id: String, cb: Function(err: String, r: Model): void): void`
    si occupa di chiamare la funzione in DataTier che si occupa di prelevare un template dal database con un determinato id.
    Parametri:
    - `id: String`
      contiene l'id del template che si vuole prelevare dal database.
    - `cb: Function(err: String, r: Model): void`
      rappresenta la callback che ritornerà il risultato alla funzione chiamante tramite l'oggetto di risposta r.
  - `+ getCategoria(cat: String, cb: Function(err: String, r: Model): void): void`
    si occupa di chiamare la funzione in DataTier che si occupa di prelevare i tem-

plate di una determinata categoria dal database.
Parametri:

* `cat: String`
  contiene la categoria di cui si vogliono prelevare i template dal database.
* `cb: Function(err: String, r: Model): void`
  rappresenta la callback che ritornerà il risultato alla funzione chiamante tramite l'oggetto di risposta r.

4.1.5 Back-end::ApplicationTier::Generator

4.1.5.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene le classi necessarie per trasformare gli oggetti JSON in codice sorgente di un un particolare linguaggio target. Tali linguaggi sono identificati da ulteriori package interni che possono essere aggiunti nel tempo.
- **Package contenuti**:
  - `Back-end::Generator::JavaGenerator`

4.1.5.2 Classi

4.1.5.2.1 BaseGenerator

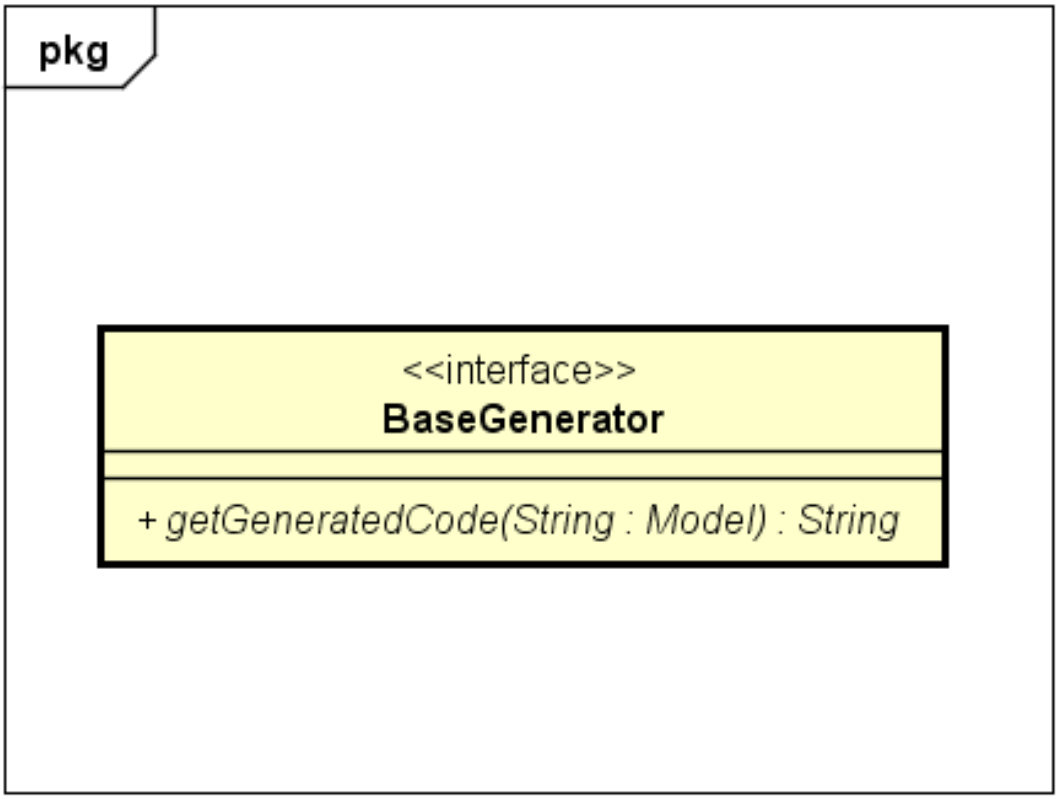

Figura 47: Back-end::ApplicationTier::Generator::BaseGenerator

- **Descrizione**:
  Questa interfaccia rappresenta un contratto comune a tutte le classi che generano codice. Rappresenta l'interfaccia *Strategy*$_G$ del design pattern Strategy.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::ApplicationController` per invocare l'algoritmo di generazione codice nel linguaggio voluto attraverso le sue sottoclassi concrete.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::ApplicationController`: utilizza `Back-end::ApplicationTier::Generator::BaseGenerator` per invocare l'algoritmo di generazione codice.

- **Sottoclassi**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::JavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedCode(String: Model): String`
    si occupa di chiamare la funzione di generazione del codice.
    Parametri:
    * `String: Model`
      contiene il JSON rappresentante un oggetto GoJs di tipo Model che rappresenta l'insieme di blocchi del diagramma delle classi e dei diagrammi delle attività.

4.1.6 Back-end::ApplicationTier::Generator::JavaGenerator

4.1.6.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene le classi relative alla generazione di codice Java partendo dal file JSON ricavato dai diagrammi delle classi e delle attività.
- **Package contenuti**:
  - `Back-end::Generator::JavaGenerator::ClassDiaJavaGenerator`
  - `Back-end::Generator::JavaGenerator::ActivityDiaJavaGenerator`

4.1.6.2 Classi

4.1.6.2.1 JavaGenerator

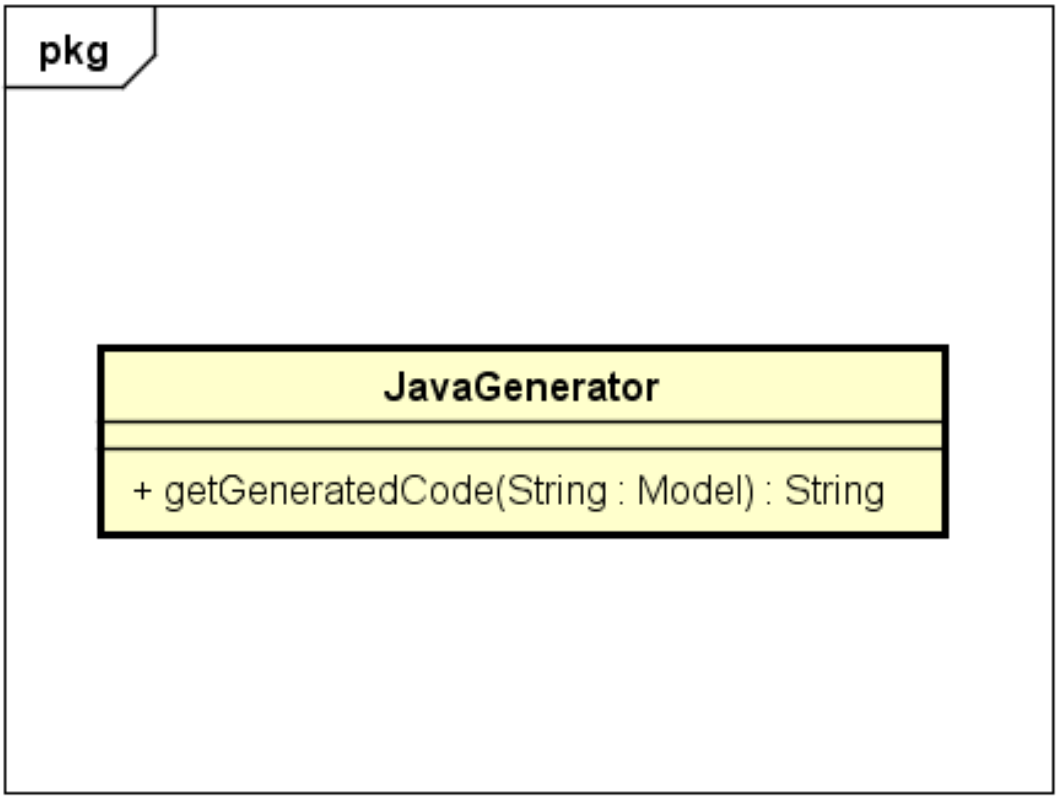

Figura 48: Back-end::ApplicationTier::Generator::JavaGenerator::JavaGenerator

- **Descrizione**:
  Questa classe implementa l'interfaccia `Back-end::ApplicationTier::Generator::JavaGenerator::JavaGenerator` e si occupa della generazione di codice Java partendo dal file JSON rappresentante un oggetto GoJs di tipo Model ricavato dai diagrammi delle classi e delle attività. Rappresenta una classe ConcreteStrategy del design pattern *Strategy*.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::ApplicationController` per generare codice Java fornendo il file JSON rappresentante un oggetto GoJs di tipo Model proveniente dal client.
- **Relazioni con altre classi**:

  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator`: necessaria per ottenere il codice Java partendo dal file JSON rappresentante un oggetto GoJs di tipo Model ricavato dal diagramma delle classi.

- **Interfacce implementate**:
  - `Back-end::ApplicationTier::Generator::BaseGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedCode(String: Model): String`
    si occupa di chiamare la funzione di generazione del codice.
    Parametri:
    * `String: Model`
      contiene il JSON rappresentante un oggetto GoJs di tipo Model che rappresenta l'insieme di blocchi del diagramma delle classi e dei diagrammi delle attività.

4.1.7 Back-end::ApplicationTier::Generator::JavaGenerator ::ClassDiaJavaGenerator

4.1.7.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene le classi relative alla generazione di codice Java partendo dal file Model ricavato dai diagrammi delle classi.

4.1.7.2 Classi

4.1.7.2.1 ClassDiaJavaGenerator

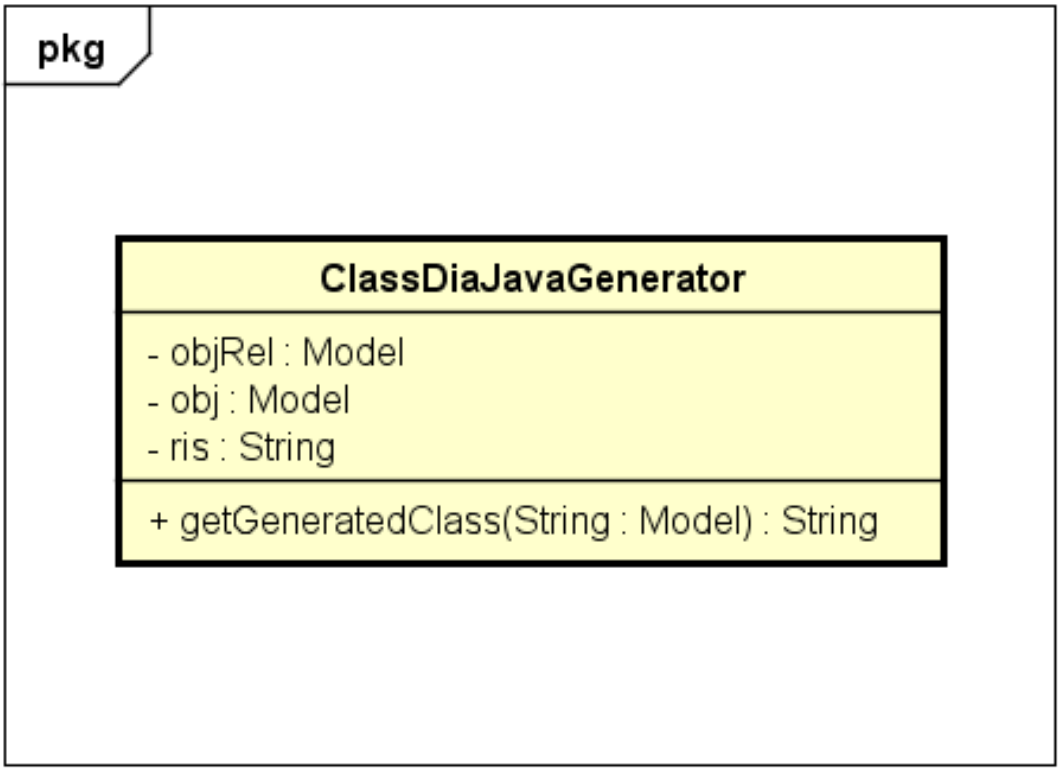

Figura 49: Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::ClassDiaJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione di codice Java di un diagramma delle classi partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da
  `Back-end::ApplicationTier::Generator::JavaGenerator::JavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un diagramma delle classi.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator ::JavaGenerator`: utilizza `Back-end::ApplicationTier::Generator`

`::JavaGenerator::ClassDiaJavaGenerator` per ottenere il codice Java partendo dal file Model ricavato dal diagramma delle classi;

- OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassJavaGenerator`: necessaria perché la generazione del codice di un diagramma delle classi comprende la generezione del codice di una classe partendo dal Model;
- OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::RelationJavaGenerator`: necessaria perché la generazione del codice di un diagramma delle classi comprende la generezione del codice di una relazione partendo dal Model;
- OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::CommentJavaGenerator`: necessaria perché la generazione del codice di un diagramma delle classi comprende la generezione del codice di un commento partendo dal Model;

- **Attributi**:
  - `- obj: Model`
    contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle classi creato dall'utente.
  - `- objRel: Model`
    contiene il Model che rappresenta tutte le relazioni presenti nel diagramma delle classi creato dall'utente.
  - `- ris: String`
    rappresenta il risultato del codice generato in forma di stringa .
- **Metodi**:
  - `+ getGeneratedClass(String: Model): String`
    si occupa di generare il codice di tutte le classi, relativi metodi e attributi creati dall'utente.
    Parametri:
    * `String: Model`
      contiene il Model che rappresenta l'insieme di blocchi del diagramma delle classi e dei diagrammi delle attività.

4.1.7.2.2 ClassJavaGenerator

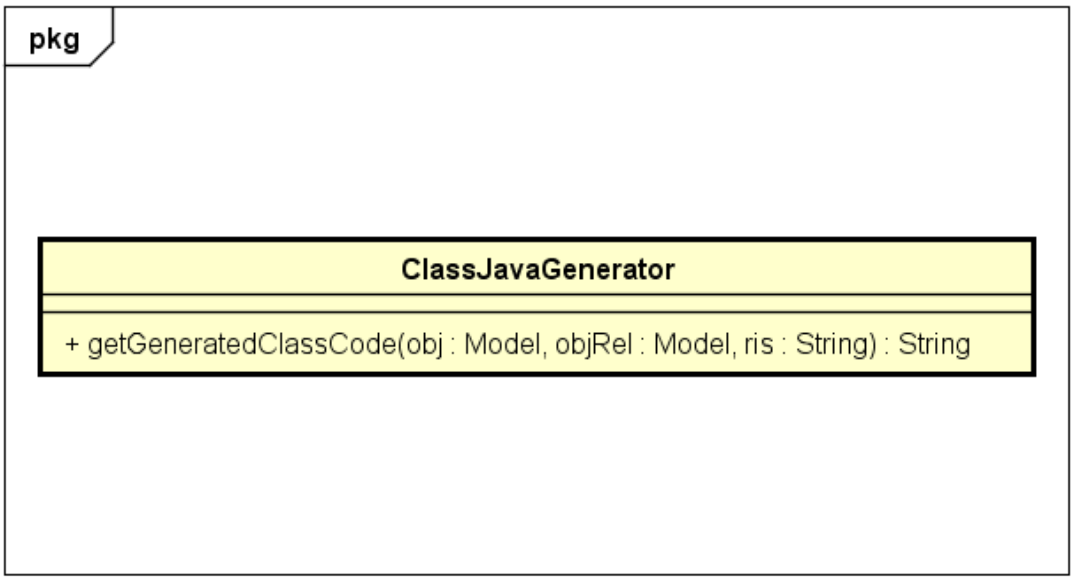

Figura 50: Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::ClassJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di una classe partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da
  `Back-end::ApplicationTier::Generator::JavaGenerator`
  `::ClassDiaJavaGenerator`
  `::ClassDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente una classe.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassDiaJavaGenerator`: utilizza `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassJavaGenerator` per la generazione di codice Java del file Model relativo ad una classe;
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::AttributeJavaGenerator`: necessaria perché la generazione del codice di una classe comprende la generazione del codice di un attributo appartenente alla classe stessa partendo dal Model;

  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::MethodJavaGenerator`: necessaria perché la generazione del codice di una classe comprende la generazione del codice di un metodo appartenente alla classe stessa partendo dal Model;

- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedClassCode(obj: Model, objRel: Model, ris: String): String` si occupa di generare il codice di una classe con i suoi relativi metodi e attributi.
    Parametri:
    * `obj: Model`
      contiene il Model che rappresenta le informazioni all'interno di una classe.
    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma delle classi creato dall'utente. .
    * `ris: Model`
      rappresenta il risultato del codice generato in forma di stringa .

4.1.7.2.3 AttributeJavaGenerator

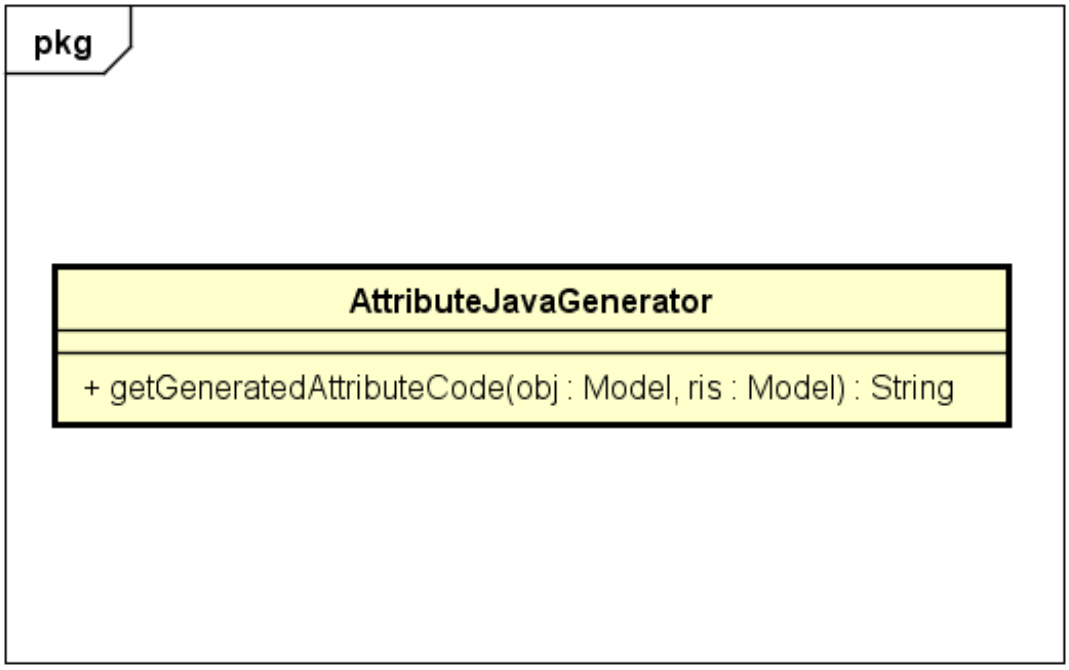

Figura 51: Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::AttributeJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un attributo partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un attributo.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassJavaGenerator`: utilizza `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::AttributeJavaGenerator` perché la generazione del codice di una classe comprende la generazione del codice di un attributo appartenente alla classe stessa partendo dal Model;
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::MethodJavaGenerator`: utilizza `Back-end`

`::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::AttributeJavaGenerator` perché la generazione del codice di un metodo comprende la generazione del codice di un attributo appartenente al metodo stesso partendo dal Model;

- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedAttributeCode(obj: Model, ris: Model): String`
    si occupa di generare il codice degli attributi di una classe.
    Parametri:
    * `obj: Model`
      contiene il Model che rappresenta le informazioni all'interno di una classe.
    * `ris: Model`
      rappresenta il risultato del codice generato in forma di stringa .

4.1.7.2.4 MethodJavaGenerator

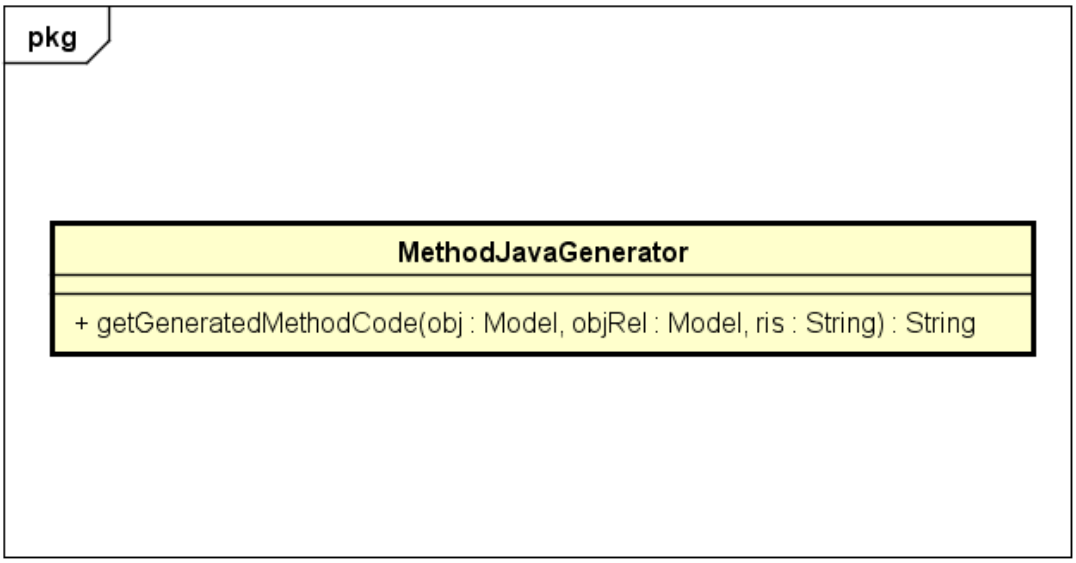

Figura 52: Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::MethodJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un metodo partendo dal file Model.

- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator ::ClassDiaJavaGenerator::ClassJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un metodo. Utilizza `Back-end:: ApplicationTier::Generator::JavaGenerator::AcitvityDiaJavaGenerator ::ClassDiaJavaGenerator` per la generazione del codice del diagramma delle attività associato al metodo.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator:: ClassDiaJavaGenerator::ClassJavaGenerator`: utilizza `Back-end ::ApplicationTier::Generator::JavaGenerator:: ClassDiaJavaGenerator::MethodJavaGenerator` perché la generazione del codice di una classe comprende la generazione del codice di un metodo appartenente alla classe stessa partendo dal Model;
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator:: ClassDiaJavaGenerator::AttributeJavaGenerator`: necessaria perché la generazione del codice di un metodo comprende la generazione del codice di un attributo appartenente al metodo stesso partendo dal Model;
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator ::ClassDiaJavaGenerator::ClassDiaJavaGenerator`: necessaria perché la generazione del codice di un metodo fa parte sia della generazione del codice di un diagramma delle classi che della generazione del codice di un diagramma delle attività.
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedMethodCode(obj: Model, objRel: Model, ris: String): String` si occupa di generare il codice dei metodi di una classe.
    Parametri:
    - `obj: Model`
      contiene il Model che rappresenta le informazioni all'interno di una classe.
    - `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma delle classi creato dall'utente.
    - `ris: Model`
      rappresenta il risultato del codice generato in forma di stringa .

4.1.7.2.5 RelationJavaGenerator

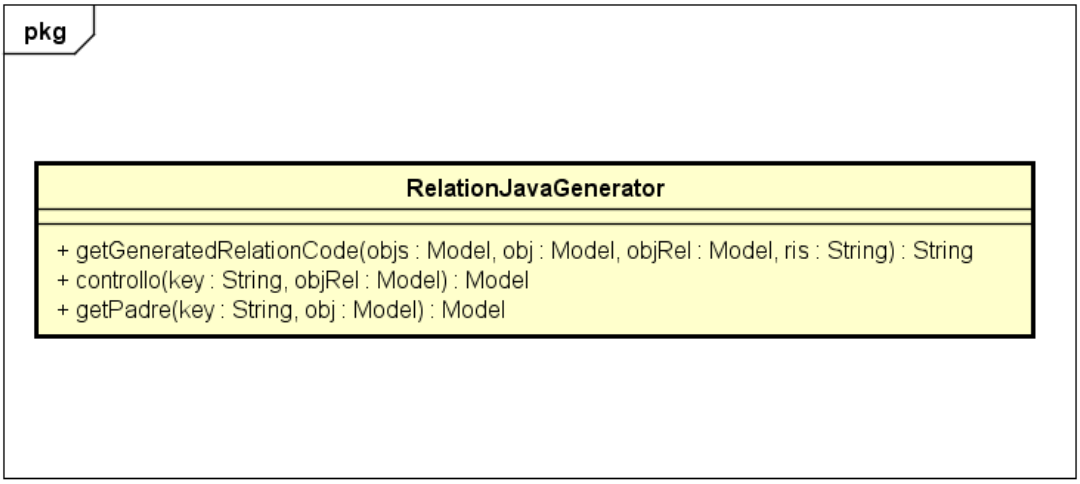

Figura 53: Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::RelationJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di una relazione partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente una relazione tra classi.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassDiaJavaGenerator`: utilizza `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::RelationJavaGenerator` perché la generazione del codice di una relazione fa parte della generazione del codice di un diagramma delle classi.
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedRelationCode(objs: Model, obj: Model, objRel: Model, ris: String): String`
    si occupa di generare il codice delle relazioni tra le classi.
    Parametri:

    * `objs: Model`
      contiene il Model che rappresenta le informazioni all'interno di una classe.

    * `obj: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle classi e nei diagrammi delle attività creati dall'utente.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma delle classi creato dall'utente.

    * `ris: Model`
      rappresenta il risultato del codice generato in forma di stringa .

- `+ getPadre(key: String, obj: Model): Model`
  si occupa di ritornare il padre di una classe sottotipo.
  Parametri:

    * `key: String`
      contiene la key della classe padre da ricercare.

    * `obj: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle classi creato dall'utente.

- `+ controllo(key: String, objRel: Model): Model`
  si occupa di ritornare un Model che contiene un booleano true, se la classe con key uguale a key (parametro formale) è sottotipo di un'altra classe, e le informazioni della classe padre. Nel caso in cui la classe considerata non sia sottotipo ritorna un Model contenente il booleano a false e le informazioni sulla classe padre vuote.
  Parametri:

    * `key: String`
      contiene la key della classe da considerare.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma delle classi creato dall'utente.

4.1.7.2.6 CommentJavaGenerator

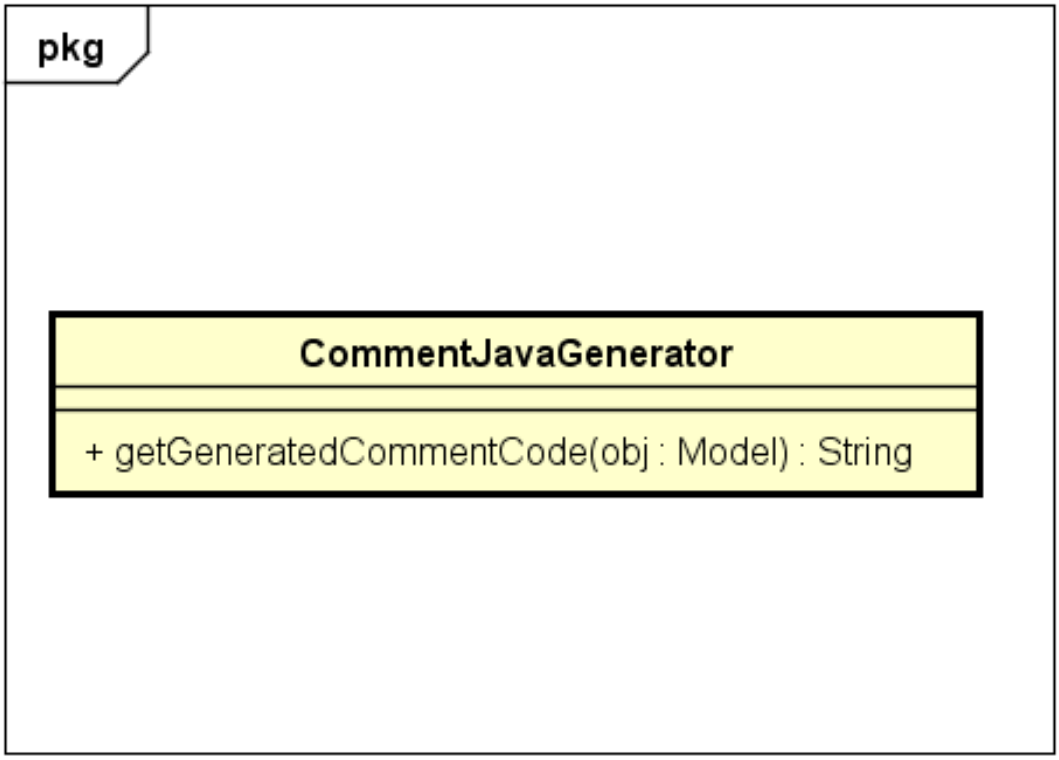

Figura 54: Back-end::ApplicationTier::Generator::JavaGenerator::
ClassDiaJavaGenerator::CommentJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un commento partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un commento.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::ClassDiaJavaGenerator`: utilizza `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::CommentJavaGenerator` perché la generazione del codice di un commento fa parte della generazione del codice di un diagramma delle classi.

- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedCommentCode(obj:Model): String`
    si occupa di generare il codice dei commenti nel diagramma delle classi.
    Parametri:
    * `obj: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle classi creato dall'utente.

4.1.8 Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator

4.1.8.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene le classi relative alla generazione di codice Java partendo dal file Model ricavato dai diagrammi delle attività.

4.1.8.2 Classi

4.1.8.2.1 ActivityDiaJavaGenerator

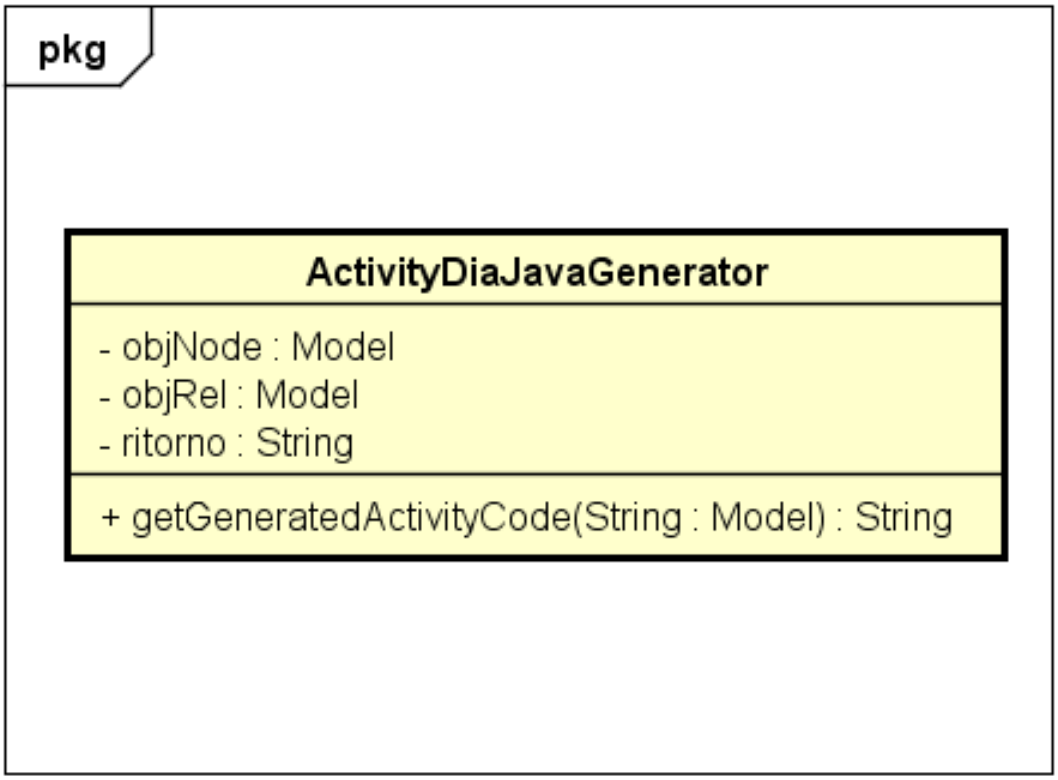

Figura 55: Back-end::ApplicationTier::Generator::JavaGenerator::
ActivityDiaJavaGenerator::ActivityDiaJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione di codice Java di un diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ClassDiaJavaGenerator::MethodJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un diagramma delle attività corrispondente al corpo di un metodo di una classe definita. Viene inoltre utilizzata per la generazione del codice Java relativo al corpo dei cicli e degli if/else in quanto essi vengono trattati come dei sotto-diagrammi delle attività.
- **Relazioni con altre classi**:

  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator ::ClassDiaJavaGenerator::MethodJavaGenerator`: utilizza `Back-end ::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator ::ActivityDiaJavaGenerator` per generare il codice Java corrispondente al corpo di un metodo di una classe definita;
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::CycleJavaGenerator`: utilizza `Back-end ::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator ::ActivityDiaJavaGenerator` per generare il codice Java corrispondente al corpo di un blocco ciclo all'interno di un metodo di una classe definita;
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::IfElseJavaGenerator`: utilizza `Back-end ::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator ::ActivityDiaJavaGenerator` per generare il codice Java corrispondente al corpo di un blocco if/else all'interno di un metodo di una classe definita;
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::InstructionJavaGenerator`: necessaria per semplificare la generazione di codice in quanto si ha un contratto comune a tutte le classi.

- **Attributi**:
  - `- objNode: Model`
    contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.
  - `- objRel: Model`
    contiene il Model che rappresenta tutte le relazioni presenti nel diagramma delle attività di un metodo creato dall'utente.
  - `- ritorno: String`
    rappresenta il risultato del codice generato in forma di stringa .
- **Metodi**:
  - `+ getGeneratedActivityCode(String: Model): String`
    si occupa di scrivere il codice del corpo di un metodo di una classe.
    Parametri:
    * `String: Model`
      contiene il Model che rappresenta l'insieme dei blocchi del diagramma delle attività di un metodo.

4.1.8.2.2 InstructionJavaGenerator

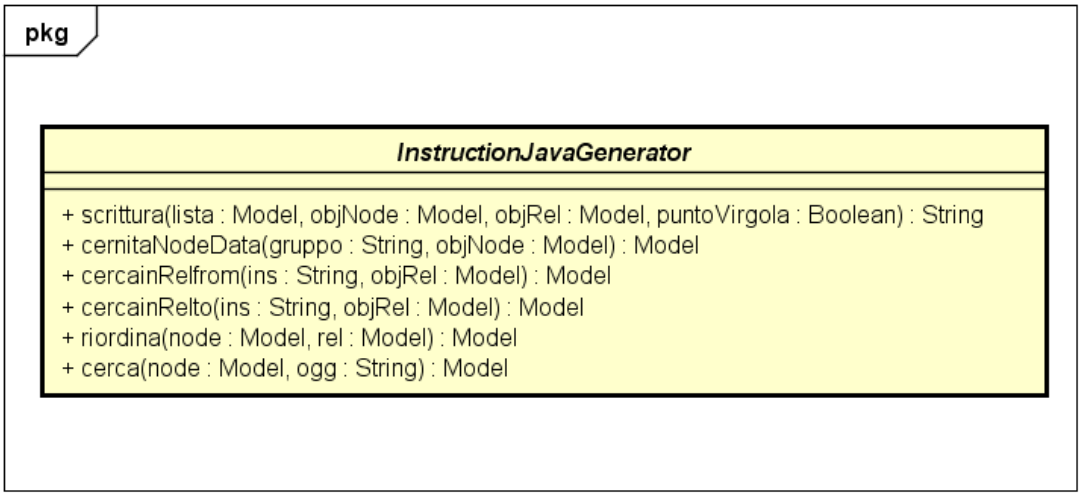

Figura 56: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::InstructionJavaGenerator

- **Descrizione**:
  Questa classe astratta rappresenta un contratto comune a tutte le classi che generano codice Java relativo a blocchi del diagramma delle attività.
- **Utilizzo**:
  Viene utilizzata per rappresentare un oggetto base comune a tutte le classi che rappresentano istruzioni all'interno di un diagramma delle attività.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator`: utilizza `Back-end ::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator ::InstructionJavaGenerator` per la generazione di codice a partire dal file Model.
- **Sottoclassi**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::VariableJavaGenerator`
  - `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::MethodCallJavaGnerator`
  - `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::JollyJavaGenerator`
  - `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::StepJavaGenerator`

  - `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::CycleJavaGenerator`
  - `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::OperatorJavaGenerator`
  - `Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::IfElseJavaGenerator`

- **Attributi**: Assenti.
- **Metodi**:
  - `+ scrittura(lista: Model, objNode: Model, objRel: Model, puntoVirgola: Boolean): String`
    si occupa di scrivere il codice di un blocco presente nel diagramma delle attività di un metodo.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta un blocco del diagramma delle attività.
    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.
    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.
    * `puntovirgola: Boolean`
      rappresenta una variabile booleana utilizzata per determinare la possibile scrittura del punto e virgola.
  - `+ cernitaNodeData(gruppo: String, objNode: Model): Model`
    si occupa di ritornare un Model contenente i blocchi appartenenti ad un determinato livello del diagramma delle attività di un metodo.
    Parametri:
    * `gruppo: String`
      rappresenta il livello richiesto.
    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.
  - `+ riordina(node: Model, rel: Model): Model`
    si occupa di riordinare il parametro node contenente un insieme di blocchi tenendo conto delle relazioni del diagramma delle attività di un metodo.
    Parametri:

    * `node: Model`
      contiene una serie di blocchi di un diagramma delle attività uniti tra loro.

    * `rel: Model` contiene le relazioni tra i blocchi presenti nel diagramma delle attività considerato.

- `+ cercainRelto(ins: String, objRel: Model): Model`
  si occupa della ricerca di una determinata relazione nel campo "to".
  Parametri:

    * `ins: String`
      rappresenta la chiave che deve essere presente nella relazione ricercata.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

- `+ cercainRelfrom(ins:String, objRel:Model): Model`
  si occupa della ricerca di una determinata relazione nel campo "from".
  Parametri:

    * `ins: String`
      rappresenta la chiave che deve essere presente nella relazione ricercata.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività creato dall'utente.

- `+ cerca(node: Model, ogg: String): Model`
  si occupa della ricerca di un blocco del diagramma delle attività che ha il campo "key" uguale al parametro formale ogg.
  Parametri:

    * `node: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

    * `ogg: String`
      rappresenta la key che il blocco ricercato deve avere.

4.1.8.2.3 VariableJavaGenerator

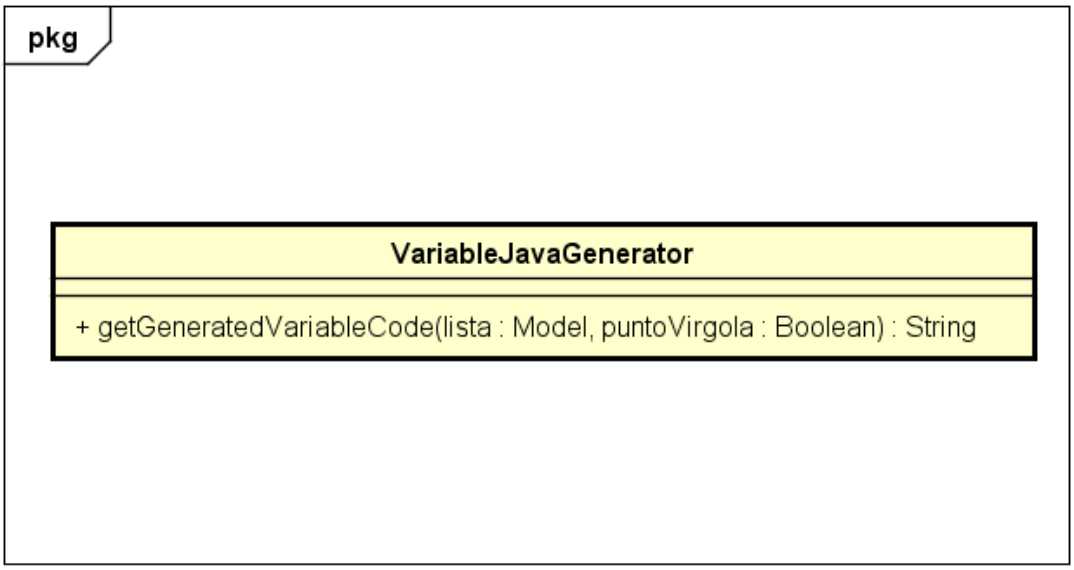

Figura 57: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::VariableJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un blocco variabile del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un blocco variabile del diagramma delle attività.
- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::InstructionJavaGenerator`
- **Sottoclassi**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::VariableConnectJavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:

– `+ getGeneratedVariableCode(lista: Model, puntoVirgola: Boolean): String`
si occupa di generare il codice di un blocco variabile del diagramma delle attività.
Parametri:

* `lista: Model`
contiene il Model che rappresenta le informazioni di un blocco variabile del diagramma delle attività.

* `puntoVirgola: Boolean`
rappresenta una variabile booleana utilizzata per determinare la possibile scrittura del punto e virgola.

4.1.8.2.4 VariableConnectJavaGenerator

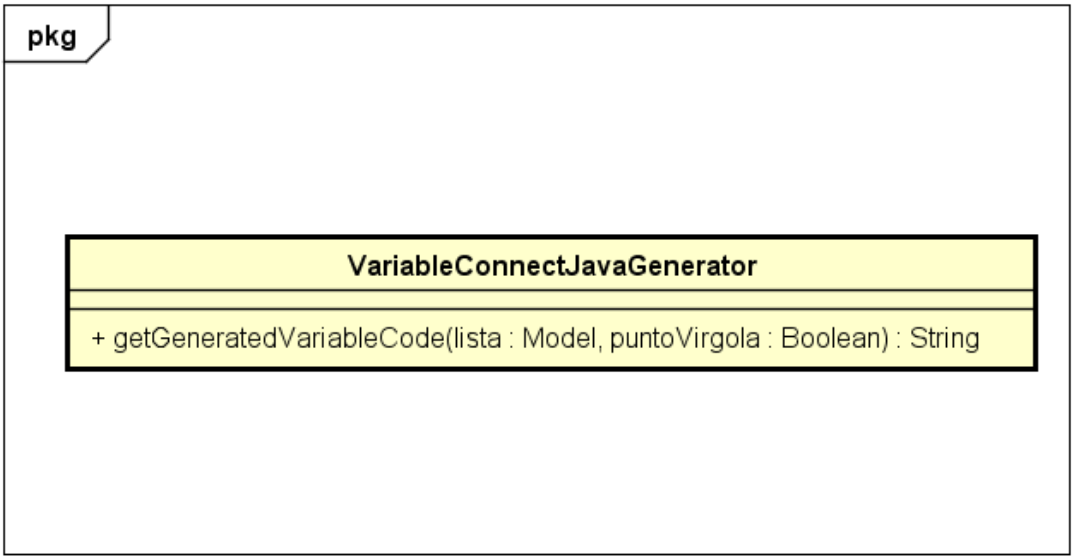

Figura 58: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::VariableConnectJavaGenerator

- **Descrizione**:
Questa classe si occupa della generazione del codice Java di un blocco connessione variabile esistente del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un blocco connessione variabile esistente del diagramma delle attività.

- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::VariableJavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedVariableCode(lista: Model, puntoVirgola: Boolean): String` si occupa di generare il codice di un blocco connessione variabile del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni di un blocco connessione variabile del diagramma delle attività.
    * `puntoVirgola: Boolean`
      rappresenta una variabile booleana utilizzata per determinare la possibile scrittura del punto e virgola.

4.1.8.2.5 MethodCallJavaGenerator

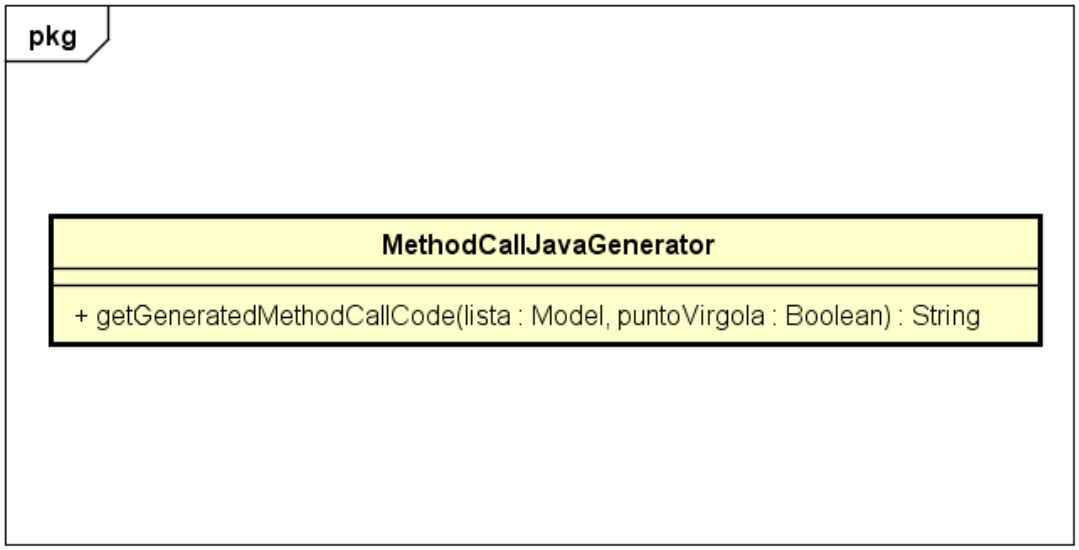

Figura 59: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::MethodCallJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un blocco chiamata del metodo del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un blocco chiamata del metodo del diagramma delle attività.
- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::InstructionavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedMethodCallCode(lista: Model, puntoVirgola: Boolean): String` si occupa di generare il codice di un blocco chiamata metodo del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni di un blocco chiamata metodo del diagramma delle attività.
    * `puntoVirgola: Boolean`
      rappresenta una variabile booleana utilizzata per determinare la possibile scrittura del punto e virgola.

4.1.8.2.6 JollyJavaGenerator

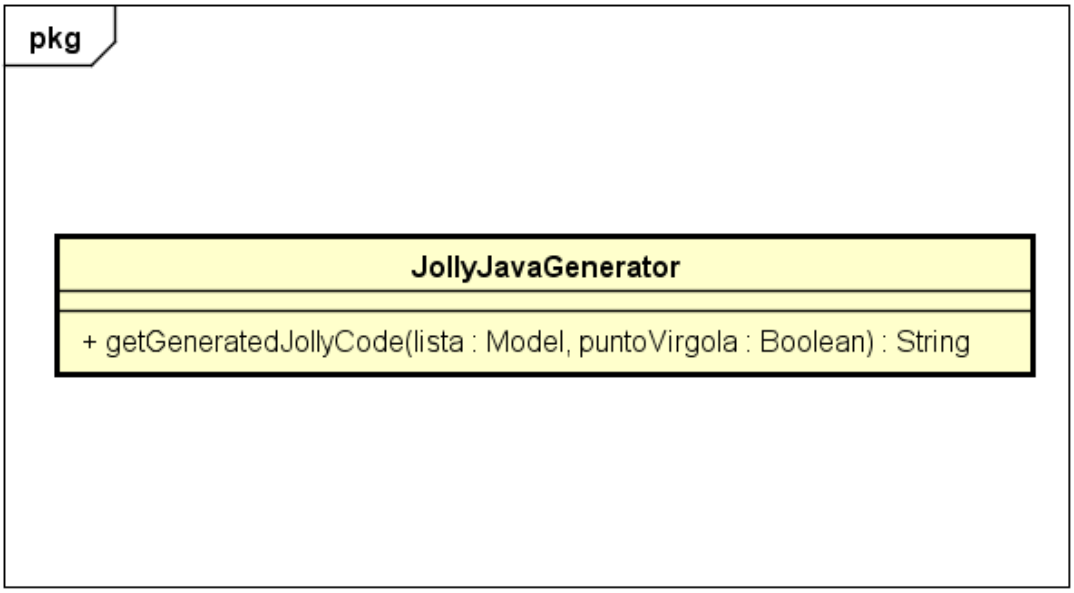

Figura 60: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::JollyJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un blocco jolly del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un blocco jolly del diagramma delle attività.
- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::InstructionavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedJollyCode(lista: Model, puntoVirgola: Boolean): String` si occupa di generare il codice di un blocco jolly del diagramma delle attività.
    Parametri:

* `lista: Model`
  contiene il Model che rappresenta le informazioni di un blocco Jolly del diagramma delle attività.
* `puntoVirgola: Boolean`
  rappresenta una variabile booleana utilizzata per determinare la possibile scrittura del punto e virgola.

4.1.8.2.7 StepJavaGenerator

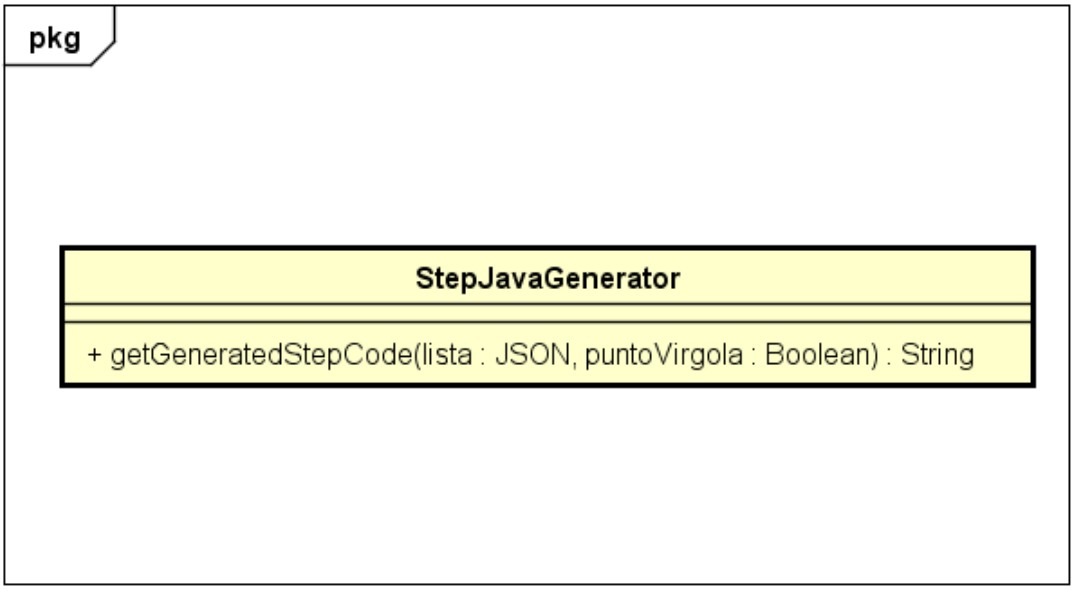

Figura 61: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::StepJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un blocco avanzamento del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un blocco avanzamento del diagramma delle attività.
- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::InstructionavaGenerator`

- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedStepCode(lista: Model, puntoVirgola: Boolean): String` si occupa di generare il codice di un blocco avanzamento del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni di un blocco avanzamento del diagramma delle attività.
    * `puntoVirgola: Boolean`
      rappresenta una variabile booleana utilizzata per determinare la possibile scrittura del punto e virgola.

4.1.8.2.8 CycleJavaGenerator

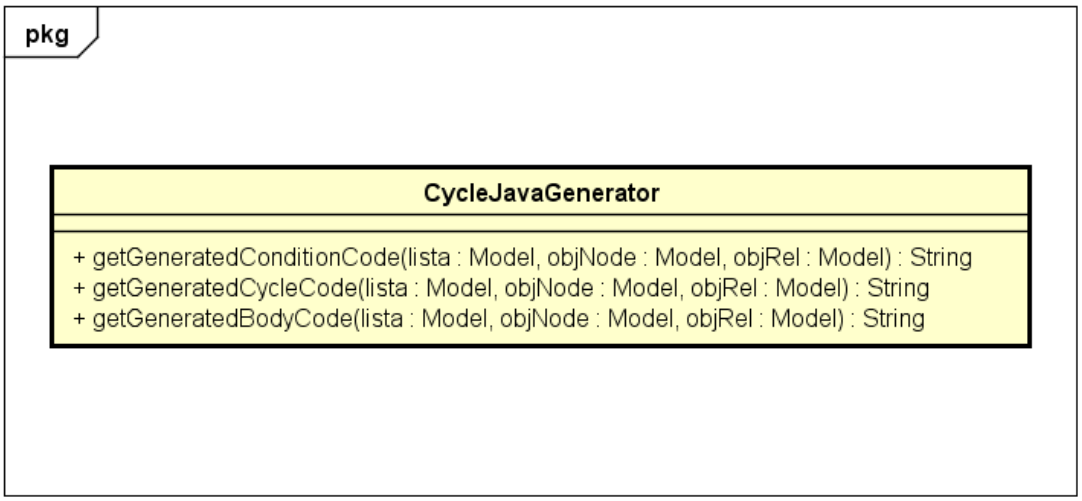

Figura 62: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::CycleJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un blocco ciclo del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del

codice Java relativo alla porzione di file Model contenente un blocco ciclo del diagramma delle attività.

- **Relazioni con altre classi**
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator`: necessaria perché rappresenta la classe che permette di ottenere il codice di tutto il diagramma delle attività associato a quel metodo;
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::OperatorJavaGenerator`: necessaria perché un blocco ciclo può contenere un blocco operatore.
- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::InstructionavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedCycleCode(lista: Model, objNode: Model, objRel: Model): String`
    si occupa di generare il codice di un blocco ciclo del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni di un blocco ciclo del diagramma delle attività.
    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.
    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.
  - `+ getGeneratedConditionCode(lista: Model, objNode: Model, objRel: Model): String`
    si occupa di generare il codice della condizione di un blocco ciclo del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni della condizione di un blocco ciclo del diagramma delle attività.

* `objNode: Model`
  contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

* `objRel: Model`
  contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

– `+ getGeneratedBodyCode(lista: Model, objNode: Model, objRel: Model): String`
si occupa di generare il codice del corpo di un blocco ciclo del diagramma delle attività.
Parametri:

* `lista: Model`
  contiene il Model rappresentante le informazioni del corpo di un blocco ciclo del diagramma delle attività.

* `objNode: Model`
  contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

* `objRel: Model`
  contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

4.1.8.2.9 OperatorJavaGenerator

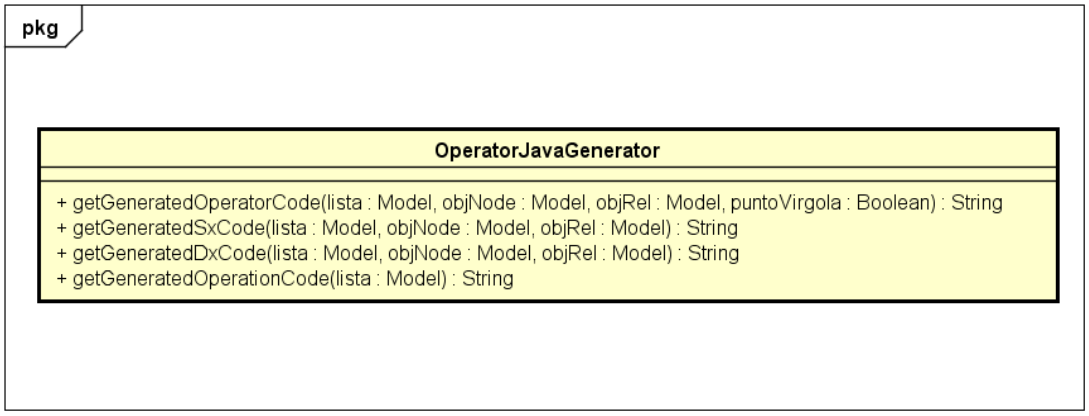

Figura 63: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::OperatorJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un blocco operatore del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un blocco operatore del diagramma delle attività.
- **Relazioni con altre classi**
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::CycleJavaGenerator`: utilizza `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::OperatorJavaGenerator` perché un blocco operatore può essere contenuto dentro un blocco ciclo;
  - IN: `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::IfElseJavaGenerator`: utilizza `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::OperatorJavaGenerator` perché un blocco operatore può essere contenuto dentro un blocco if/else.
- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::InstructionavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedOperatorCode(lista: Model, objNode: Model, objRel: Model, puntoVirgola: Boolean): String`
    si occupa di generare il codice di un blocco operatore del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni di un blocco operatore del diagramma delle attività.
    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

    * `puntoVirgola: Boolean`
      rappresenta una variabile booleana utilizzata per determinare la possibile scrittura del punto e virgola.

- `+ getGeneratedSxCode(lista: Model, objNode: Model, objRel: Model): String`
  si occupa di generare il codice dell'operatore sinistro di un blocco operatore del diagramma delle attività.
  Parametri:

    * `lista: Model`
      contiene il Model che rappresenta le informazioni dell'operatore sinistro del blocco operatore del diagramma delle attività.

    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

- `+ getGeneratedDxCode(lista: Model, objNode: Model, objRel: Model): String`
  si occupa di generare il codice dell'operatore destro di un blocco operatore del diagramma delle attività.
  Parametri:

    * `lista: Model`
      contiene il Model che rappresenta le informazioni dell'operatore destro del blocco operatore del diagramma delle attività.

    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

- `+ getGeneratedOperationCode(lista: Model): String`
  si occupa di generare il codice del blocco operazione di un blocco operatore del diagramma delle attività.
  Parametri:

* `lista: Model`
  contiene il Model che rappresenta le informazioni di un blocco operazione del blocco operatore del diagramma delle attività.

4.1.8.2.10 IfElseJavaGenerator

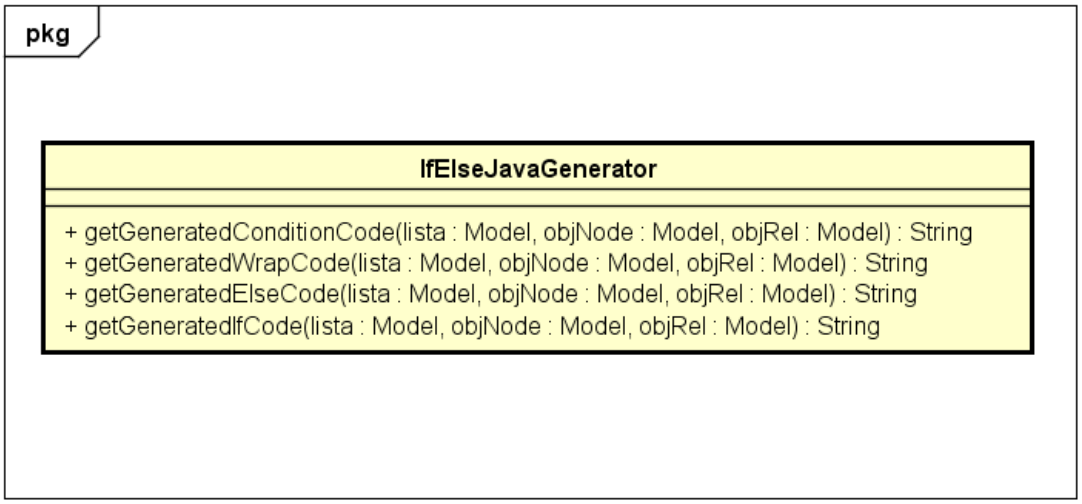

Figura 64: Back-end::ApplicationTier::Generator::JavaGenerator::
::ActivityDiaJavaGenerator::IfElseJavaGenerator

- **Descrizione**:
  Questa classe si occupa della generazione del codice Java di un blocco if/else del diagramma delle attività partendo dal file Model.
- **Utilizzo**:
  Viene utilizzata da `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator` per la generazione del codice Java relativo alla porzione di file Model contenente un blocco if/else del diagramma delle attività.
- **Relazioni con altre classi**
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator`: necessaria perché rappresenta la classe che permette di ottenere il codice di tutto il diagramma delle attività associato a quel metodo;
  - OUT: `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::OperatorJavaGenerator`: necessaria perchè un blocco if/else può contenere un blocco operatore.

- **Classi ereditate**:
  - `Back-end::ApplicationTier::Generator::JavaGenerator::ActivityDiaJavaGenerator::InstructionavaGenerator`
- **Attributi**: Assenti.
- **Metodi**:
  - `+ getGeneratedWrapCode(lista: Model, objNode: Model, objRel: Model): String` si occupa di generare il codice di un blocco if/else del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni del blocco if/else del diagramma delle attività.
    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.
    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.
  - `+ getGeneratedIfCode(lista: Model, objNode: Model, objRel: Model): String` si occupa di generare il codice del corpo dell'if di un blocco if/else del diagramma delle attività.
    Parametri:
    * `lista: Model`
      contiene il Model che rappresenta le informazioni del corpo dell' if di un blocco if/else del diagramma delle attività.
    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.
    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.
  - `+ getGeneratedConditionCode(lista: Model, objNode: Model, objRel: Model): String` si occupa di generare il codice della condizione di un blocco if/else del diagramma delle attività.
    Parametri:

    * `lista: Model`
      contiene il Model che rappresenta le informazioni della condizione del blocco if/else del diagramma delle attività.

    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

  – `+ getGeneratedElseCode(lista: Model, objNode: Model, objRel: Model): String`
    si occupa di generare il codice del corpo dell'else di un blocco if/else del diagramma delle attività.
    Parametri:

    * `lista: Model`
      contiene il Model che rappresenta le informazioni del corpo dell'else di un blocco if/else del diagramma delle attività.

    * `objNode: Model`
      contiene il Model che rappresenta tutti gli oggetti presenti nel diagramma delle attività di un metodo creato dall'utente.

    * `objRel: Model`
      contiene il Model che rappresenta tutte le relazioni presenti nel diagramma dell'attività di un metodo creato dall'utente.

4.1.9 Back-end::ApplicationTier::Error

4.1.9.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene la classe che gestisce la visualizzazione degli errori.

4.1.9.2 Classi

4.1.9.2.1 ErrorApplication

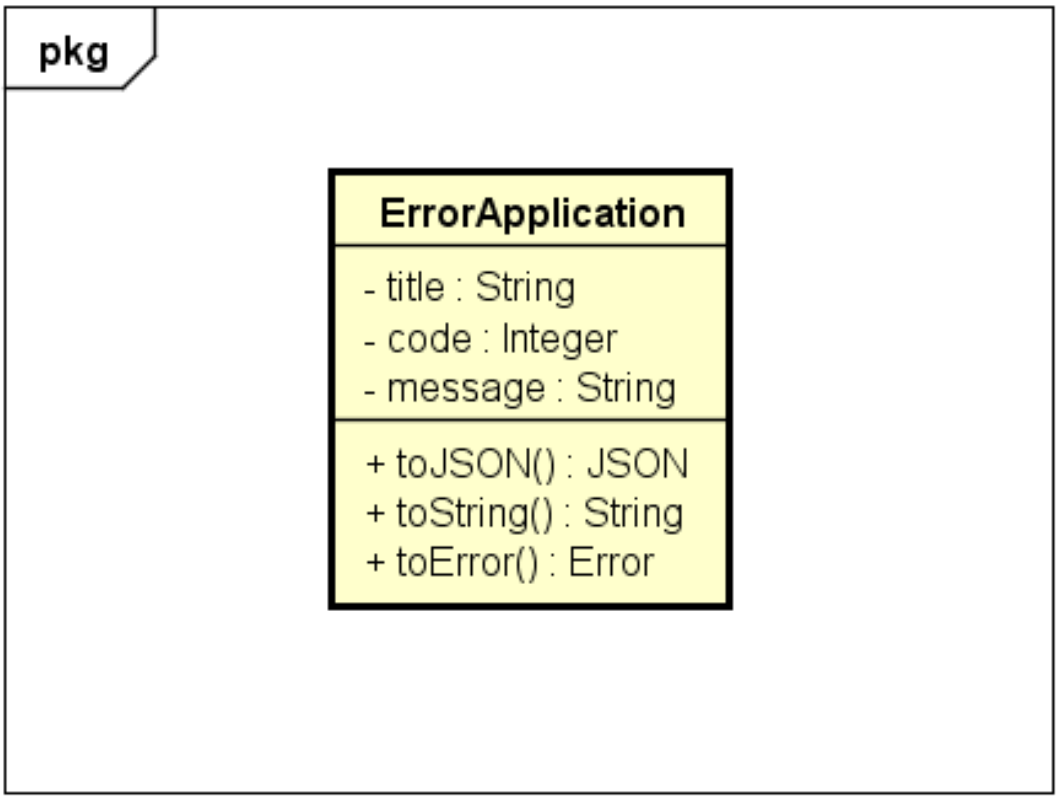

Figura 65: Back-end::ApplicationTier::Error::ErrorApplication

- **Descrizione**:
  Questa classe rappresenta un errore che può verificarsi nel `Back-end` .
- **Utilizzo**:
  Viene utilizzata da tutte le classi presenti all'interno del package `Back-end` per rappresentare un errore generato, identificandolo tramite codice, nome e descrizione.
- **Attributi**:
  - `- title: String`
    rappresenta il titolo dell'errore.
  - `- code: Integer`
    rappresenta il codice dell'errore.
  - `- message: String`
    rappresenta il messaggio corrispondente all'errore .
- **Metodi**:

- `+ toJSON(): JSON`
  si occupa di tornare l'errore in formato JSON.
- `+ toString(): String`
  si occupa di effettuare una concatenazione dei campi dati dell'errore in formato String e la ritorna.
- `+ toError(): Error`
  si occupa di convertire l'errore al tipo Error utilizzato da *Node.js*$_G$ ritornandolo.

4.1.10 Back-end::DataTier

4.1.10.1 Informazioni sul package

- **Descrizione**:
  Questo package contiene la classe che gestisce il recupero dei dati di libreria interfacciandosi con un database MongoDB, tramite la libreria Mongoose. Costituisce la parte Presentation dell'architettura Three-tier del back-end.
- **Framework esterni**:
  - MongoDB
  - Mongoose

4.1.10.2 Classi

4.1.10.2.1 Template

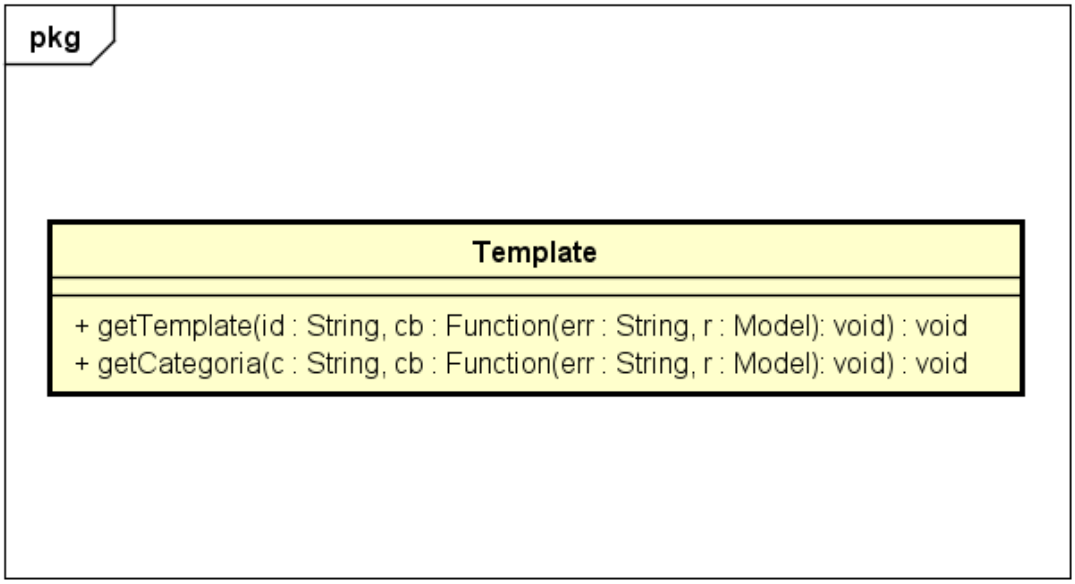

Figura 66: Back-end::DataTier::Template

- **Descrizione**:
  Questa classe gestisce la richiesta di un template JSON rappresentante un oggetto GoJs di tipo Model da parte del client richiedendolo al Database.
- **Utilizzo**:
  Viene utilizzato per gestire la richiesta di un template JSON rappresentante un oggetto GoJs di tipo Model da parte di `Back-end::ApplicationTier::ApplicationController` richiedendolo al database MongoDB utilizzando la libreria Mongoose.
- **Relazioni con altre classi**:
  - IN: `Back-end::ApplicationTier::ApplicationController`: utilizza `Back-end::DataTier::Template` per gestire la richiesta di un template JSON rappresentante un oggetto GoJs di tipo Model.
- **Attributi**: Assenti.
- **Metodi**:

- `+ getTemplate(id: String, cb: Function(err: String, r: Model): void): void`
  si occupa di prelevare dal database un template con un determinato id.
  Parametri:
  * `id: String`
    rappresenta il campo id del template che si vuole prelevare dal database.
  * `cb: Function(err: String, r: Model): void`
    rappresenta la callback che ritornerà il risultato alla funzione chiamante tramite l'oggetto rispota r.
- `+ getCategoria(c: String, cb: Function(err: String, r: Model): void)): void`
  si occupa di prelevare dal database i template di una determinata categoria.
  Parametri:
  * `c: String`
    rappresenta il campo categoria dei template che si vogliono prelevare dal database.
  * `cb: Function(err: String, r: Model): void`
    rappresenta la callback che ritornerà il risultato alla funzione chiamante tramite l'oggetto risposta r.

5 Diagrammi di sequenza

In questa sezione sono riportati i diagrammi di sequenza per le interazioni tra classi più complesse presenti all'interno del prodotto.

5.1 Generazione Codice

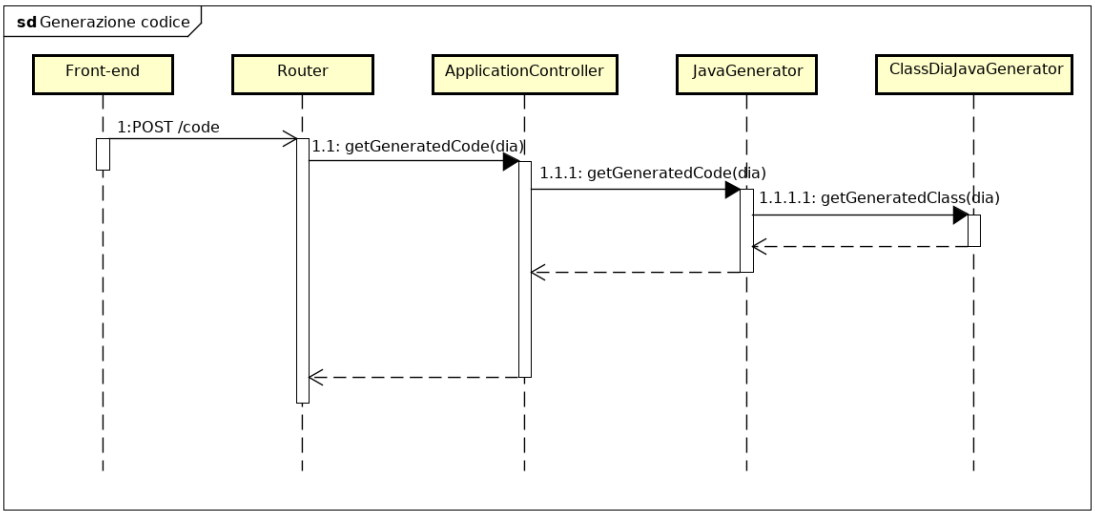

Figura 67: Diagramma di Sequenza Generazione Codice

A seguito di una richiesta HTML di generazione codice da parte del front-end il back-end invoca il metodo di generazione del codice che analizza il diagramma delle classi generando il codice opportuno.

5.2 Generazione Codice Classe

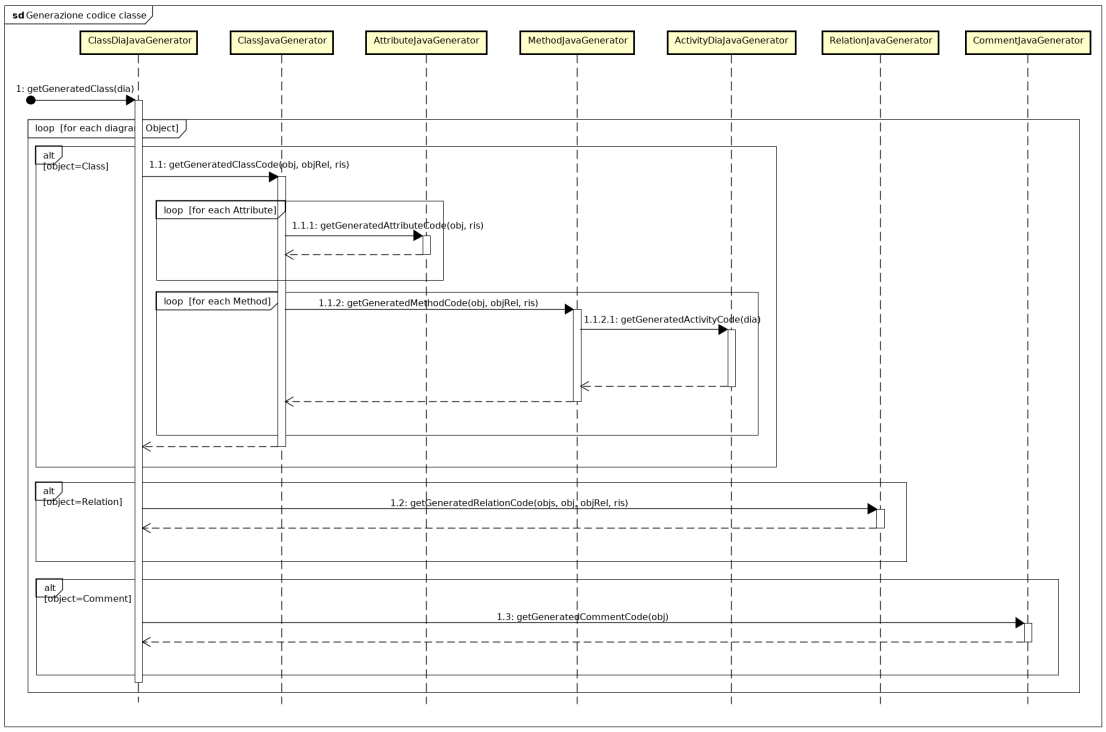

Figura 68: Diagramma di Sequenza Generazione Codice Classe

A seguito di una invocazione del metodo di generazione del codice di un diagramma delle classi viene invocato il metodo che attraversa il diagramma generando il codice di ciascuno degli oggetti che incontra.

Oggetto incontrato	Azione
Classe	• Generazione del codice per ciascun attributo; • Generazione del codice per ciascun metodo (invocando la generazione del codice per il diagramma delle attività associato).
Relazione	Generazione del codice per la relazione
Commento	Generazione del codice per il commento

Tabella 2: Diagramma di sequenza - Generazione codice classe

5.3 Generazione Codice Metodo

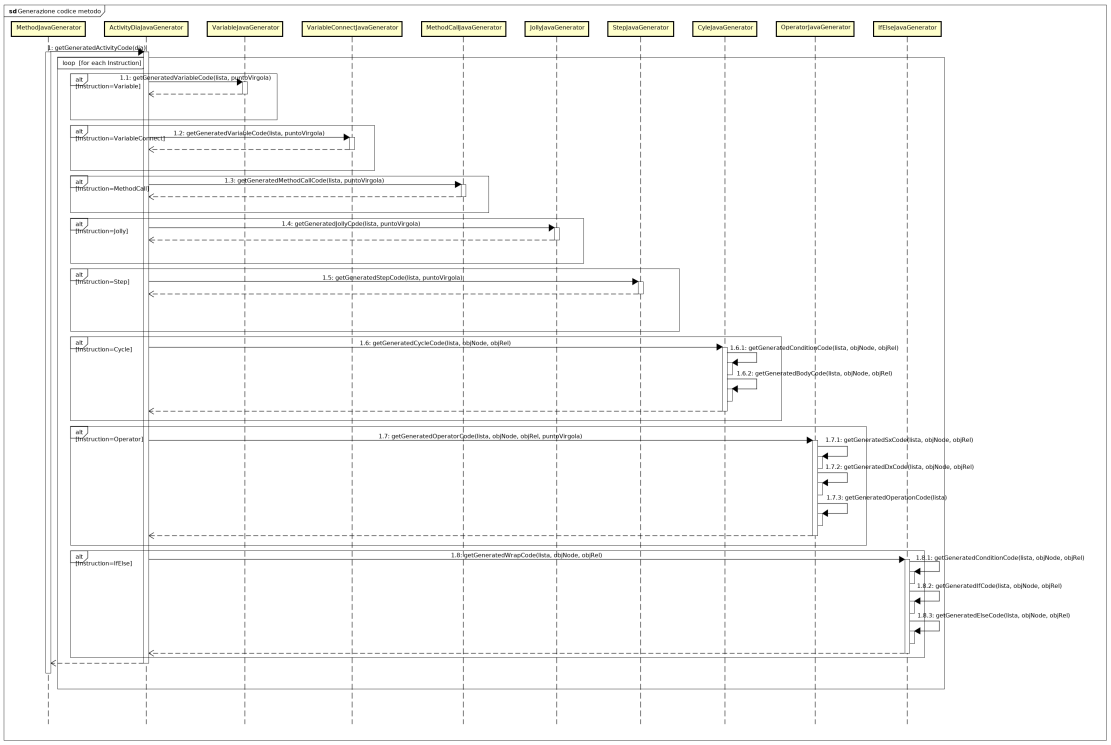

Figura 69: Diagramma di Sequenza Generazione Codice Metodo

A seguito di una invocazione del metodo di generazione del codice del corpo di un metodo viene invocato il metodo che attraversa il diagramma delle attività corrispondente generando il codice di ciascuno degli oggetti che incontra.

Oggetto incontrato	Azione
Variabile	Generazione del codice per il blocco variabile
Connessione variabile	Generazione del codice per il blocco variabile esistente
Chiamata metodo	Generazione del codice per il blocco chiamata ad un metodo
Jolly	Generazione del codice per il blocco jolly
Avanzamento	Generazione del codice per il blocco avanzamento
Ciclo	• Generazione del codice per la condizione; • Generazione del codice per il corpo.
Operator	• Generazione del codice per l'elemento di sinistra; • Generazione del codice per l'elemento di destra; • Generazione del codice per l'operazione.
If/Else	• Generazione del codice per la condizione; • Generazione del codice per il blocco if; • Generazione del codice per il blocco else.

Tabella 3: Diagramma di sequenza - Generazione codice metodo

5.4 Ottenimento template

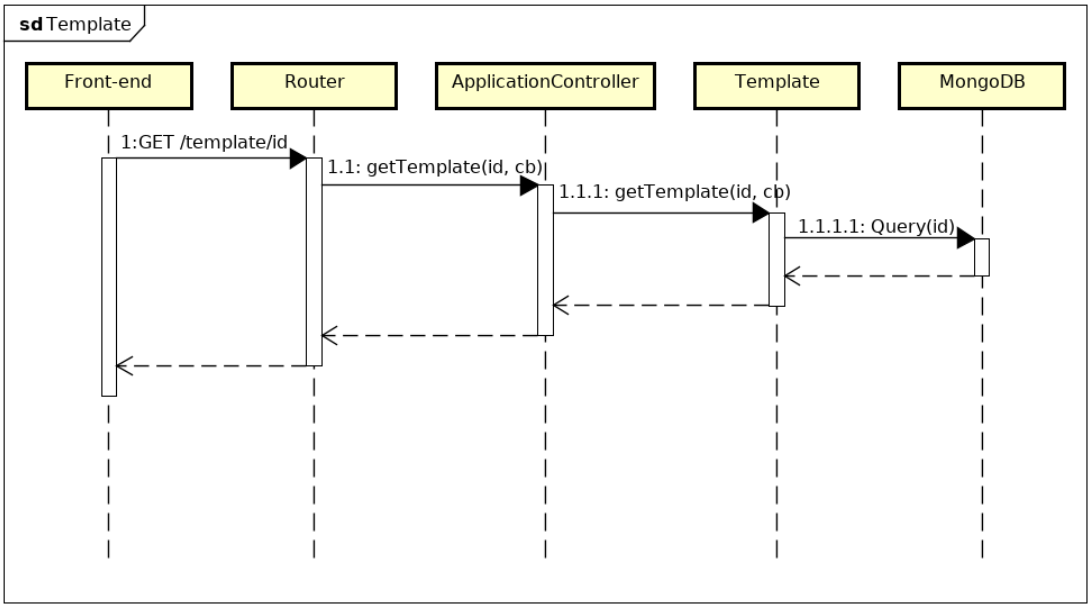

Figura 70: Diagramma di Sequenza Ottenimento template

A seguito di una richiesta HTML di ottenimento template (corredata di id del template) da parte del front-end il back-end invoca il metodo di ritorno del template. Questo metodo interroga MongoDB e ritorna il file JSON contenente il template di id richiesto.

5.5 Richiesta Generazione Codice

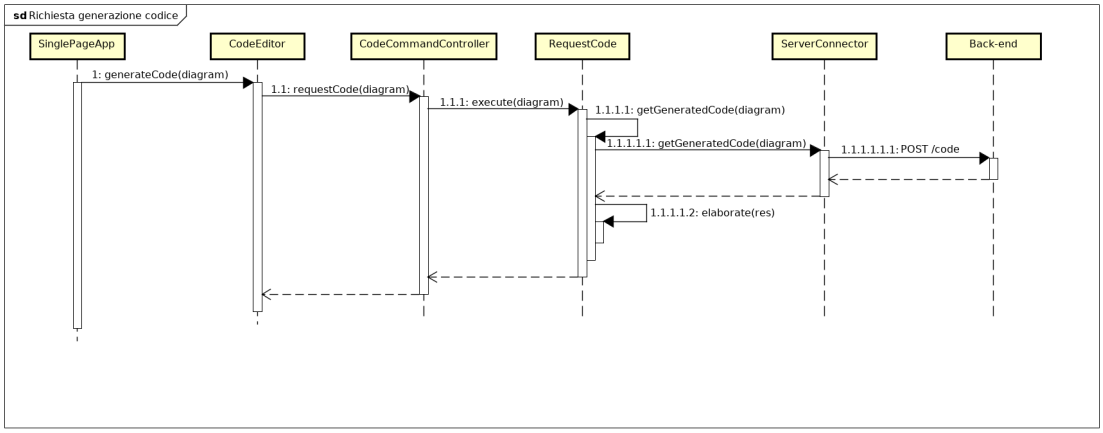

Figura 71: Diagramma di Sequenza Richiesta Generazione Codice

A seguito di una richiesta di generazione del codice per il popolamento dell'editor del codice la classe `Front-end::View::CodeEditor` invoca il proprio controller. Quest'ultimo esegue il comando di generazione codice (design pattern Command). La classe che rappresenta il comando di generazione codice (`Front-end::Model::Services::ServerConnector`) si occupa di effettuare la chiamata POST relativa alla generazione del codice e restituire la risposta del server. Infine la classe comando elabora la risposta ritornandola al generatore attraverso il controller.

6 Tracciamento

6.1 Requisiti-Classi

Requisito	Classi
R1O0	Front-end::View::SinglePageApp
R1O2	Front-end::View::SinglePageApp
R1O2.1	Front-end::View::SinglePageApp
R1O3	Front-end::View::SinglePageApp
R1O3.2	Front-end::View::SinglePageApp
R1O3.2.1	Back-end::ApplicationTier::Error:: ErrorApplication
R1O4	Front-end::View::DiagramEditor Front-end::View::ClassDiagramEditor Front-end::View::ClassFactory Front-end::View::DiagramPalette Front-end::View::ClassDiagramPalette Front-end::View::SinglePageApp Front-end::ViewModel::EditorObjController Front-end::ViewModel::PaletteObjController Front-end::ViewModel::PaletteCommandController
R1O4.1	Front-end::Model::Objects::BaseDiaObj Front-end::View::DiagramEditor Front-end::View::DiagramPalette Front-end::Model::Objects::ClassObjects::ClassDiaObj Front-end::View::ClassDiagramPalette Front-end::Model::Objects::ClassObjects::ClassFactory Front-end::View::ClassDiagramEditor Front-end::View::DiagramFactory Front-end::View::ClassFactory
R1O4.3	Front-end::Model::Objects::ClassObjects::Class
R1O4.4	Front-end::View::ClassDiagramEditor
R1O4.4.1	Front-end::View::ClassDiagramEditor
R1O4.4.2	Front-end::View::ClassDiagramEditor
R1D4.4.3	Front-end::View::ClassDiagramEditor
R1O4.4.5	Front-end::View::ClassDiagramEditor
R1O4.4.5.1	Front-end::View::ClassDiagramEditor
R1O4.4.5.2	Front-end::View::ClassDiagramEditor
R1O4.4.5.3	Front-end::View::ClassDiagramEditor
R1O4.4.5.4	Front-end::View::ClassDiagramEditor
R1O4.4.6	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.7	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8	Front-end::View::ClassDiagramEditor

	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.1	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.2	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.3	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.4	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.5	Front-end::View::ClassDiagramEditor Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.5.1	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.5.2	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.5.3	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.8.6	Front-end::Model::Objects::ClassObjects::Class
R1O4.4.9	Front-end::View::ClassDiagramEditor Front-end::Model::Objects::ClassObjects::Class
R1O4.4.10	Front-end::View::ClassDiagramEditor Front-end::Model::Objects::ClassObjects::Class
R1O4.5	Front-end::View::ClassDiagramEditor
R1O4.6	Front-end::View::ClassDiagramEditor Front-end::Model::Objects::ClassObjects::Relation
R1O4.6.1	Front-end::Model::Objects::ClassObjects::Association
R1O4.6.2	Front-end::Model::Objects::ClassObjects::Relation
R1O4.6.3	Front-end::Model::Objects::ClassObjects::Composition
R1O4.6.4	Front-end::Model::Objects::ClassObjects::Aggregation
R1O4.6.5	Front-end::Model::Objects::ClassObjects::Generalization
R1O4.7	Front-end::View::ClassDiagramEditor Front-end::Model::Objects::ClassObjects::Relation
R1O4.7.1	Front-end::Model::Objects::ClassObjects::Relation
R1O4.7.2	Front-end::Model::Objects::ClassObjects::Dependency Front-end::Model::Objects::ClassObjects::Association Front-end::Model::Objects::ClassObjects::Aggregation Front-end::Model::Objects::ClassObjects::Composition Front-end::Model::Objects::ClassObjects::Generalization
R1O4.7.3	Front-end::Model::Objects::ClassObjects::Relation
R1O4.8	Front-end::View::ClassDiagramEditor Front-end::Model::Objects::ClassObjects::Relation
R1D4.9	Front-end::View::ClassDiagramEditor Front-end::Model::Objects::ClassObjects::Comment
R1D4.10	Front-end::Model::Objects::ClassObjects::Comment
R1D4.11	Front-end::Model::Objects::ClassObjects::Comment
R1D4.12	Front-end::Model::Objects::ClassObjects::ClassDiaObj
R1D4.13	Front-end::Model::Objects::ClassObjects::ClassDiaObj
R1D4.17	Front-end::Model::Objects::ClassObjects::Class

R1O5	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj Front-end::View::ActivityDiagramEditor Front-end::View::ActivityFactory Front-end::View::DiagramPalette Front-end::View::ActivityDiagramPalette Front-end::ViewModel::EditorObjController Front-end::ViewModel::PaletteObjController Front-end::ViewModel::PaletteCommandController Front-end::View::SinglePageApp
R1O5.1	Front-end::View::ActivityDiagramPalette
R1O5.2	Front-end::View::ActivityDiagramEditor
R1O5.2.1	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1O5.2.1.1	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::VariableObj
R1O5.2.1.1.1	Front-end::Model::Objects::ActivityObjects::VariableObj
R1O5.2.1.2	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::ExistingVariableObj Front-end::Model::Objects::ActivityObjects::MethodCallObj
R1O5.2.1.3	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::CycleObj
R1O5.2.1.4	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::IfElseObj
R1O5.2.1.5	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::OperatorObj
R1O5.2.1.6	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::StepObj
R1O5.2.1.6.1	Front-end::Model::Objects::ActivityObjects::StepObj
R1O5.2.1.6.2	Front-end::Model::Objects::ActivityObjects::StepObj
R1O5.2.1.7	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::JollyObj
R1O5.2.2	Front-end::View::ActivityDiagramEditor
R1O5.2.2.1	Front-end::View::ActivityDiagramEditor
R1O5.2.2.1.1	Front-end::View::ActivityDiagramEditor
R1O5.2.2.1.2	Front-end::View::ActivityDiagramEditor
R1O5.2.2.2	Front-end::View::ActivityDiagramEditor
R1O5.2.2.3	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::CycleObj
R1O5.2.2.3.1	Front-end::Model::Objects::ActivityObjects::CycleObj

R1O5.2.2.3.2	Front-end::Model::Objects::ActivityObjects::CycleObj
R1O5.2.2.4	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::IfElseObj
R1O5.2.2.4.1	Front-end::Model::Objects::ActivityObjects::IfElseObj
R1O5.2.2.4.2	Front-end::Model::Objects::ActivityObjects::IfElseObj
R1O5.2.2.4.3	Front-end::Model::Objects::ActivityObjects::IfElseObj
R1O5.2.2.5	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::OperatorObj
R1O5.2.2.5.1	Front-end::Model::Objects::ActivityObjects::OperatorObj
R1O5.2.2.5.2	Front-end::Model::Objects::ActivityObjects::OperatorObj
R1O5.2.2.5.3	Front-end::Model::Objects::ActivityObjects::OperatorObj
R1O5.2.2.5.4	Front-end::Model::Objects::ActivityObjects::OperatorObj
R1O5.2.2.6	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::StepObj
R1O5.2.2.6.1	Front-end::Model::Objects::ActivityObjects::StepObj
R1O5.2.2.6.2	Front-end::Model::Objects::ActivityObjects::StepObj
R1O5.2.2.7	Front-end::View::ActivityDiagramEditor Front-end::Model::Objects::ActivityObjects::JollyObj
R1O5.2.3	Front-end::View::ActivityDiagramEditor
R1O5.2.3.1	Front-end::View::ActivityDiagramEditor
R1O5.2.3.2	Front-end::View::ActivityDiagramEditor
R1O5.2.3.3	Front-end::View::ActivityDiagramEditor
R1O5.2.3.4	Front-end::View::ActivityDiagramEditor
R1O5.2.3.5	Front-end::View::ActivityDiagramEditor
R1O5.2.3.6	Front-end::View::ActivityDiagramEditor
R1O5.2.3.7	Front-end::View::ActivityDiagramEditor
R1D5.2.4	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.4.1	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.4.2	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.4.3	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.4.4	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.4.5	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.5	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj

R1D5.2.5.1	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.5.2	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.5.3	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.5.4	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.5.5	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1D5.2.6	Front-end::Model::Objects::ActivityObjects::ActivityDiaObj
R1O5.3	Front-end::View::ActivityDiagramEditor
R1O5.4 R1O6	Front-end::Model::Commands::GetTemplate Front-end::View::CodeEditor Front-end::Model::Services::ServerConnector Front-end::ViewModel::CodeCommandController Front-end::Model::Commands::Command Front-end::Model::Commands::RequestCode
R1O6.1	Back-end::PresentationTier::Middleware::ErrorHandler Back-end::PresentationTier::Middleware::NotFoundHandler
R1O6.2	Front-end::View::SinglePageApp Front-end::ViewModel::CodeCommandController
R106.3	Front-end::View::SinglePageApp Front-end::Model::Commands::Command
R107	Front-end::View::SinglePageApp Front-end::Model::Commands::Command
R1O7.1	Front-end::View::SinglePageApp
R1O7.1.1	Front-end::View::SinglePageApp Back-end::ApplicationTier::Error:: ErrorApplication
R1O9	Front-end::Model::Commands::Command Back-end::PresentationTier::Middleware::MiddlewareLoader Back-end::PresentationTier::Middleware::Router Back-end::PresentationTier::Controller::IndexGiver Back-end::ApplicationTier::ApplicationController Back-end::ApplicationTier::Generator::BaseGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::JavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ClassDiaJavaGenerator

	Back-end::ApplicationTier::Generator::JavaGenerator ::ClassJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::AttributeJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::MethodJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::RelationJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::CommentJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::InstructionJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::VariableJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::VariableConnectJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::MethodCallJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::JollyJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::StepJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::CycleJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::OperatorJavaGenerator Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::IfElseJavaGenerator Back-end::DataTier::Template

Tabella 4: Tracciamento Requisiti-Classi

6.2 Classi-Requisiti

Classi	Requisiti
Front-end::View::SinglePageApp	R1O0 R1O2 R1O2.1 R1O3 R1O3.2 R1O4 R1O5 R1O6.2 R1O6.3 R1O7 R1O7.1 R1O7.1.1
Front-end::View::CodeEditor	R1O6
Front-end::View::DiagramEditor	R1O4 R1O4.1
Front-end::View::ClassDiagramEditor	R1O4 R1O4.1
Front-end::View::ClassDiagramEditor	R1O4.4 R1O4.4.1 R1O4.4.2 R1D4.4.3 R1O4.4.5 R1O4.4.5.1 R1O4.4.5.2 R1O4.4.5.3 R1O4.4.5.4 R1O4.4.8 R1O4.4.8.5 R1O4.4.9 R1O4.4.10 R1O4.5 R1O4.6 R1O4.7 R1O4.8 R1D4.9
Front-end::View::ActivityDiagramEditor	R1O5 R1O5.2 R1O5.2.1.1 R1O5.2.1.2 R1O5.2.1.3

	R1O5.2.1.4 R1O5.2.1.5 R1O5.2.1.6 R1O5.2.1.7 R1O5.2.2 R1O5.2.2.1 R1O5.2.2.1.1 R1O5.2.2.1.2 R1O5.2.2.2 R1O5.2.2.3 R1O5.2.2.4 R1O5.2.2.5 R1O5.2.2.6 R1O5.2.2.7 R1O5.2.3 R1O5.2.3.1 R1O5.2.3.2 R1O5.2.3.3 R1O5.2.3.4 R1O5.2.3.5 R1O5.2.3.6 R1O5.2.3.7 R1O5.3
Front-end::View::DiagramPalette	R1O4 R1O4.1 R1O5
Front-end::View::ClassDiagramPalette	R1O4 R1O4.1
Front-end::View::ActivityDiagramPalette	R1O5 R1O5.1
Front-end::View::DiagramFactory	R1O4.1
Front-end::View::ClassFactory	R1O4.1
Front-end::View::ActivityFactory	R1O5
Front-end::ViewModel::EditorObjController	R1O4 R1O5
Front-end::ViewModel::PaletteObjController	R1O4 R1O5
Front-end::ViewModel::CodeCommandController	R1O6 R1O6.2
Front-end::ViewModel::PaletteCommandController	R1O4 R1O5
Front-end::Model::Objects::BaseDiaObj	R1O4.1
Front-end::Model::Objects::ClassObjects::ClassDiaObj	R1O4.1

	R1D4.12 R1D4.13
Front-end::Model::Objects::ClassObjects::Class	R1O4.3 R1O4.4 R1O4.4.1 R1O4.4.2 R1O4.4.3 R1O4.4.4 R1O4.4.5 R1O4.4.5.1 R1O4.4.5.2 R1O4.4.5.3 R1O4.4.5.4 R1O4.4.6 R1O4.4.7 R1O4.4.8 R1O4.4.8.1 R1O4.4.8.2 R1O4.4.8.3 R1O4.4.8.4 R1O4.4.8.5 R1O4.4.8.5.1 R1O4.4.8.5.2 R1O4.4.8.5.3 R1O4.4.8.6 R1O4.4.9 R1O4.4.10 R1O4.16 R1D4.17
Front-end::Model::Objects::ClassObjects::Relation	R1O4.6 R1O4.6.2 R1O4.7 R1O4.7.1 R1O4.7.2 R1O4.7.3 R1O4.7.4 R1O4.8
Front-end::Model::Objects::ClassObjects::Dependency	R1O4.7.2
Front-end::Model::Objects::ClassObjects::Association	R1O4.6.1 R1O4.7.2
Front-end::Model::Objects::ClassObjects::Aggregation	R1O4.6.4 R1O4.7.2
Front-end::Model::Objects::ClassObjects::Composition	R1O4.6.3

	R1O4.7.2
Front-end::Model::Objects::ClassObjects::Generalization	R1O4.6.5 R1O4.7.2
Front-end::Model::Objects::ClassObjects::Comment	R1D4.9 R1D4.10 R1D4.11
Front-end::Model::Objects::ActivityObjects::ActivityDiaObj	R1O5 R1O5.2.1 R1D5.2.4 R1D5.2.4.1 R1D5.2.4.2 R1D5.2.4.3 R1D5.2.4.4 R1D5.2.4.5 R1D5.2.5 R1D5.2.5.1 R1D5.2.5.2 R1D5.2.5.3 R1D5.2.5.4 R1D5.2.5.5 R1D5.2.6
Front-end::Model::Objects::ActivityObjects::VariableObj	R1O5.2.1.1.1
Front-end::Model::Objects::ActivityObjects::ExistingVariableObj	R1O5.2.1.2
Front-end::Model::Objects::ActivityObjects::MethodCallObj	R1O5.2.1.2 R1O5.2.2.2.1 R1O5.2.2.2.2
Front-end::Model::Objects::ActivityObjects::CycleObj	R1O5.2.1.3 R1O5.2.2.3 R1O5.2.2.3.1 R1O5.2.2.3.2
Front-end::Model::Objects::ActivityObjects::IfElseObj	R1O5.2.1.4 R1O5.2.2.4.1 R1O5.2.2.4.2 R1O5.2.2.4.3
Front-end::Model::Objects::ActivityObjects::OperatorObj	R1O5.2.1.5 R1O5.2.2.5.1 R1O5.2.2.5.2 R1O5.2.2.5.3 R1O5.2.2.5.4
Front-end::Model::Objects::ActivityObjects::StepObj	R1O5.2.1.6

	R1O5.2.1.6.1 R1O5.2.1.6.2 R1O5.2.2.6 R1O5.2.2.6.1 R1O5.2.2.6.2
Front-end::Model::Objects::ActivityObjects::JollyObj	R1O5.2.1.7 R1O5.2.2.7
Front-end::Model::Commands::Command	R1O6 R1O6.3 R1O7 R1O9
Front-end::Model::Commands::RequestCode	R1O6
Front-end::Model::Commands::GetTemplate	R1O5.4
Front-end::Model::Services::ServerConnector	R1O6
Back-end::PresentationTier::Middleware::MiddlewareLoader	R1O9
Back-end::PresentationTier::Middleware::ErrorHandler	R1O6.1
Back-end::PresentationTier::Middleware::Router	R1O9
Back-end::PresentationTier::Middleware::NotFoundHandler	R1O6.1
Back-end::PresentationTier::Controller::IndexGiver	R1O9
Back-end::ApplicationTier::ApplicationController	R1O9
Back-end::ApplicationTier::Generator::BaseGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::JavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ClassDiaJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ClassJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::AttributeJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::MethodJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::RelationJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::CommentJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::ActivityDiaJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::InstructionJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::VariableJavaGenerator	R1O9

Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::VariableConnectJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::MethodCallJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::JollyJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::StepJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::CycleJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::OperatorJavaGenerator	R1O9
Back-end::ApplicationTier::Generator::JavaGenerator ::ActivityDiaJavaGenerator::IfElseJavaGenerator	R1O9
Back-end::ApplicationTier::Error:: ErrorApplication	R1O3.2.1 R1O7.1.1
Back-end::DataTier::Template	R1O9

Tabella 5: Tracciamento Classi-Requisiti