



Manuale del manutentore

Informazioni sul documento

Versione	2.0.0
Data di Creazione	2017-04-01
Data ultima modifica	2017-08-21
Stato	Approvato
Redazione	Jordan Gottardo
Verifica	Giovanni Prete
Approvazione	Marco Pasqualini
Uso	Esterno
Lista di distribuzione	Professor Tullio Vardanega Professor Riccardo Cardin <i>Zephyrus</i> <i>RiskApp</i>
Email di riferimento	zephyrus.swe@gmail.com

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
2.0.0	2017-08-21	Marco Pasqualini	<i>Responsabile</i>	Approvazione documento
1.1.0	2017-08-20	Giovanni Prete	<i>Verificatore</i>	Verifica documento
1.0.3	2017-08-19	Jordan Gottardo	<i>Programmatore</i>	Modificata sezione 6.1 Analisi di rischio inserendo l'attuale descrizione dell'implementazione tramite mock
1.0.2	2017-08-19	Jordan Gottardo	<i>Programmatore</i>	Rimossa sezione 6.1 Scenari di danno
1.0.1	2017-08-19	Jordan Gottardo	<i>Programmatore</i>	Aggiunto Axios sezione 2.15
1.0.0	2017-04-09	Giovanni Prete	<i>Responsabile</i>	Approvazione documento
0.3.0	2017-04-08	Leonardo Brutesco	<i>Verificatore</i>	Verifica documento
0.2.1	2017-04-06	Giulia Petenazzi	<i>Programmatore</i>	Correzione errori sezione 5
0.2.0	2017-04-05	Leonardo Brutesco	<i>Verificatore</i>	Verifica documento
0.1.3	2017-04-05	Giulia Petenazzi	<i>Programmatore</i>	Stesura sezione 6 - Estensione delle funzionalità e del codice e sezione 7 - Estensione del codice
0.1.2	2017-04-04	Giovanni Damo	<i>Programmatore</i>	Stesura sezione 5 - Componenti
0.1.1	2017-04-02	Giovanni Damo	<i>Programmatore</i>	Aggiunte versioni browser sezione 3
0.1.0	2017-04-02	Marco Pasqualini	<i>Verificatore</i>	Verifica documento
0.0.2	2017-04-01	Giovanni Damo	<i>Programmatore</i>	Stesura sezioni 1 - Introduzione, 2 - Tecnologie utilizzate e 3 - Configurazione ambiente di lavoro
0.0.1	2017-04-01	Giovanni Damo	<i>Programmatore</i>	Creazione template e indici

Indice

1 Introduzione	1
1.1 Scopo del documento	1
1.2 Scopo del prodotto	1
1.3 Glossario	1
1.4 Riferimenti	1
1.4.1 Riferimenti informativi	1
2 Tecnologie utilizzate	3
2.1 CSS3	3
2.2 HTML5	3
2.3 JavaScript ES6	4
2.4 JSON	4
2.5 JSX	5
2.6 Node.js	5
2.7 OpenLayers	5
2.8 Open Street Map	6
2.9 React	6
2.10 ReactColor	7
2.11 React-Redux	7
2.12 React Toolbox	7
2.13 Redux	8
2.14 SVG	8
2.15 Axios	8
3 Configurazione ambiente di lavoro	9
3.1 Requisiti di sistema	9
3.1.1 Dispositivi e browser supportati	9
3.2 IDE: WebStorm	9
3.3 Gestore dei pacchetti	10
3.3.1 Node.js	10
3.3.2 npm	10
3.4 Browser	11
3.4.1 Google Chrome	11
3.4.2 Mozilla Firefox	11
3.4.3 Safari	11
3.5 Download del progetto	12
3.6 Configurazione della VirtualBox	13
4 Architettura DeGeOP	14
4.1 Introduzione	14
4.2 Server	14
4.2.1 Chiamate REST	14
4.2.1.1 Customer	14
4.2.1.2 Graph	15
4.2.1.3 Asset	15
4.2.1.4 Node	15
4.2.1.5 Edge	15
4.3 Client	16

4.3.1	Design architetturale di DeGeOP	16
5	Componenti	17
5.1	DeGeOP	17
5.1.1	Informazioni sul package	17
5.2	DeGeOP::StorePkg	18
5.2.1	Informazioni sul package	18
5.3	DeGeOP::StorePkg::StoreContentsPkg	20
5.3.1	Informazioni sul package	20
5.3.2	Classi	20
5.3.2.1	Customer	20
5.3.2.2	Options	21
5.3.2.3	StoreDeGeOP	22
5.4	DeGeOP::StorePkg::ProcessPkg	23
5.4.1	Informazioni sul package	23
5.4.2	Classi	23
5.4.2.1	Asset	23
5.4.2.2	Edge	26
5.4.2.3	ExitNode	27
5.4.2.4	MachineNode	27
5.4.2.5	Node	29
5.4.2.6	Process	30
5.4.2.7	QueueNode	31
5.4.2.8	ResourceNode	32
5.4.2.9	SourceNode	32
5.5	DeGeOP::StorePkg::PolygonPkg	34
5.5.1	Informazioni sul package	34
5.5.2	Classi	34
5.5.2.1	ConcretePolygon	34
5.5.2.2	ConcretePolygonFactory	35
5.5.2.3	Coordinate	35
5.5.2.4	Polygon	36
5.5.2.5	PolygonFactory	36
5.6	DeGeOP::ReducerPkg	37
5.6.1	Informazioni sul package	37
5.6.2	Classi	37
5.6.2.1	AssetReducer	37
5.6.2.2	EdgeReducer	38
5.6.2.3	NodeReducer	38
5.6.2.4	OptionReducer	39
5.6.2.5	Reducer	39
5.7	DeGeOP::CallManagerPkg	41
5.7.1	Informazioni sul package	41
5.7.2	Classi	41
5.7.2.1	Request	41
5.7.2.2	Server	42
5.8	DeGeOP::ActionPkg	44
5.8.1	Informazioni sul package	44
5.8.2	Classi	45

5.8.2.1	EdgeAction	45
5.9	DeGeOP::ActionPkg::ActionElementsPkg	46
5.9.1	Informazioni sul package	46
5.9.2	Classi	46
5.9.2.1	Action	46
5.9.2.2	AssetAction	47
5.9.2.3	NodeAction	47
5.9.2.4	OptionAction	47
5.10	DeGeOP::ActionPkg::ActionCreatorsPkg	49
5.10.1	Informazioni sul package	49
5.10.2	Classi	49
5.10.2.1	AssetActionCreator	49
5.10.2.2	EdgeActionCreator	50
5.10.2.3	NodeActionCreator	50
5.10.2.4	OptionActionCreator	51
5.11	DeGeOP::ViewPkg	52
5.11.1	Informazioni sul package	52
5.12	DeGeOP::ViewPkg::DeGeOPViewPkg	53
5.12.1	Informazioni sul package	53
5.12.2	Classi	54
5.12.2.1	DeGeOPView	54
5.13	DeGeOP::ViewPkg::MapComponentsPkg	57
5.13.1	Informazioni sul package	57
5.13.2	Classi	57
5.13.2.1	ButtonWrapper	57
5.13.2.2	MapWrapper	58
5.13.2.3	MessageWrapper	59
5.13.2.4	PolygonOperationWrapper	60
5.14	DeGeOP::ViewPkg::SidebarPkg	61
5.14.1	Informazioni sul package	61
5.14.2	Classi	62
5.14.2.1	HomeSidebar	62
5.14.2.2	InsertAssetSidebar	62
5.14.2.3	InsertEdgeSidebar	63
5.14.2.4	InsertNodeSidebar	63
5.14.2.5	ViewAssetSidebar	64
5.14.2.6	ViewEdgeSidebar	64
5.14.2.7	ViewNodeSidebar	65
5.15	DeGeOP::ViewPkg::SidebarPkg::ContentPkg	66
5.15.1	Informazioni sul package	66
5.15.2	Classi	67
5.15.2.1	AbstractContent	67
5.15.2.2	InsertAssetContent	67
5.15.2.3	InsertEdgeContent	68
5.15.2.4	InsertNodeContent	68
5.15.2.5	ViewAssetContent	69
5.15.2.6	ViewEdgeContent	69
5.15.2.7	ViewNodeContent	70
5.16	DeGeOP::ViewPkg::SidebarPkg::ButtonsPkg	71

5.16.1	Informazioni sul package	72
5.16.2	Classi	72
5.16.2.1	AbstractButtons	72
5.16.2.2	ThreeButtons	72
5.16.2.3	TwoButtons	73
6	Estensione delle funzionalità	74
6.1	Analisi di rischio	74
6.2	Nuova tipologia di nodo	74
7	Estensione del codice	75
7.1	Creazione di una nuova categoria di sidebar	75
7.2	Aggiornamenti dei campi dati di asset e nodi	75
7.2.1	Asset	75
7.2.2	Nodo	75
A	Glossario	76
A.1	A	76
A.2	B	76
A.3	C	76
A.4	D	77
A.5	E	77
A.6	F	77
A.7	G	77
A.8	H	78
A.9	I	78
A.10	J	78
A.11	L	78
A.12	M	79
A.13	N	79
A.14	O	79
A.15	P	79
A.16	R	79
A.17	S	80
A.18	V	80
A.19	W	80

1 Introduzione

1.1 Scopo del documento

Il presente documento rappresenta il Manuale del Manutentore per il progetto *DeGeOP* sviluppato dal gruppo *Zephyrus* per il proponente *RiskApp*. All'interno di esso vengono descritte:

- le tecnologie utilizzate per lo sviluppo;
- gli strumenti utilizzati e consigliati;
- l'architettura del software con i relativi componenti;
- le funzionalità presenti.

Tutto ciò viene descritto per aiutare lo sviluppatore a comprendere a fondo l'applicazione per eventualmente apportarvi delle modifiche o estensioni, assumendo e richiedendo che lo sviluppatore conosca già le tecnologie utilizzate nel progetto e brevemente illustrate nella prossima sezione.

1.2 Scopo del prodotto

Lo scopo del prodotto consiste nella creazione di un'interfaccia web contenente una mappa geografica su cui potranno essere rappresentati:

- il processo produttivo aziendale;
- gli scenari di danno;
- i risultati dell'analisi dei rischi.

Il prodotto verrà utilizzato da agenti assicuratori per l'inserimento delle informazioni utili allo svolgimento dell'analisi dei rischi dell'assicurando.

L'interfaccia dovrà essere in grado di connettersi ai sistemi preesistenti di *RiskApp* per la memorizzazione e gestione dei dati inseriti. A causa del requisito di integrabilità, che è stato deciso di soddisfare, l'applicazione da sviluppare sarà parte integrante dell'attuale applicazione del proponente.

1.3 Glossario

Allo scopo di rendere più semplice e chiara la comprensione del manuale, nel presente documento viene incluso un breve glossario. Per evidenziare un termine presente in tale documento, esso verrà marcato con il pedice *G*. Solo la prima occorrenza del termine in ogni sezione sarà marcata per non appesantire la lettura del documento. Tutti i termini del glossario evidenziati sono link ipertestuali al termine corrispettivo presente nel glossario.

1.4 Riferimenti

1.4.1 Riferimenti informativi

- **Introduzione a CSS3:** https://www.w3schools.com/css/css3_intro.asp;
- **introduzione a HTML5:** https://www.w3schools.com/html/html5_intro.asp;
- **guida a ES6:** <http://es6-features.org/#Constants>;

- **introduzione a JSON:** https://www.w3schools.com/js/js_json_intro.asp;
- **introduzione a JSX:** <https://facebook.github.io/react/docs/introducing-jsx.html>;
- **Node.js:** <https://nodejs.org/it/>;
- **guida a OpenLayers:** <https://openlayersbook.github.io/>;
- **guida a Open Street Map:** http://wiki.openstreetmap.org/wiki/Beginners%27_guide;
- **guida a React:** <https://facebook.github.io/react/>;
- **React Color:** <https://casesandberg.github.io/react-color/>;
- **React-Redux:** <https://github.com/reactjs/react-redux>;
- **React Toolbox:** <http://react-toolbox.com/#/>;
- **Redux:** <http://redux.js.org/docs/basics/>;
- **SVG tutorial:** https://www.w3schools.com/graphics/svg_intro.asp

2 Tecnologie utilizzate

In questa sezione vengono elencate le tecnologie su cui si basa lo sviluppo del progetto. Per ognuna di esse verrà indicato l'ambito di utilizzo e una descrizione.

Tecnologia	Utilizzo
CSS3	Linguaggio per la formattazione delle pagine web
HTML5	Linguaggio per la costruzione di pagine web
JavaScript ES6	Linguaggio principale in cui è sviluppata l'applicazione
JSON	Formato dati utilizzato per lo scambio di informazioni
JSX	Estensione <i>JavaScript_G</i> per l'integrazione del codice <i>HTML_G</i>
Node.js	Ambiente operativo per utilizzare JavaScript lato server
OpenLayers	Libreria per la gestione della mappa e del grafo
Open Street Map	Libreria per la fornitura di mappe in formato vettoriale
React	Costruzione dell'interfaccia grafica
ReactColor	Libreria per gestire la paletta di colori
React-Redux	Libreria per interfacciare <i>React_G</i> con Redux
React Toolbox	Libreria che implementa la specifica di Material Design
Redux	Libreria per l'implementazione dell'architettura
SVG	Scalable Vector Graphics

Tabella 1: Tabella riassuntiva tecnologie utilizzate nel progetto

2.1 CSS3

Descrizione	È un linguaggio utilizzato per la presentazione di documenti HTML. Lo standard viene definito dal <i>W3C_G</i> .
Utilizzo	Viene utilizzato per definire il layout dell'applicazione.
Documentazione	https://www.w3schools.com/css/css3_intro.asp
ufficiale	

2.2 HTML5

Descrizione	È un <i>linguaggio di markup_G</i> utilizzato per definire la struttura delle pagine web. Lo standard viene definito dal W3C.
Utilizzo	Viene utilizzato per definire il layout dell'applicazione.
Documentazione	https://www.w3.org/TR/html5/
ufficiale	

2.3 JavaScript ES6

Descrizione	JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, utilizzato principalmente nella programmazione Web lato client. Le caratteristiche più importanti di questo linguaggio sono:
	<ul style="list-style-type: none">• eventi: quando l'utente interagisce con la pagina Web in vari modi, come ad esempio mouse e tastiera, viene generato un evento; JavaScript gestisce tali eventi, i quali possono avviare un'azione registrata in un gestore di eventi;• tipizzazione dinamica: il programmatore non è tenuto a specificare il tipo degli oggetto che utilizza;• paradigma a protipi: stile di programmazione orientato ad oggetti in cui l'ereditarietà è implementata tramite il riuso di oggetti esistenti, basandosi sul loro prototipo.
	In particolare, il <i>gruppoG</i> si baserà sull'utilizzo della specifica JavaScript ES6, che definisce significativi cambiamenti sintattici per la scrittura di applicazioni complesse in modo più semplice.
Utilizzo	JavaScript è il linguaggio base con cui si svilupperà l'applicazione <i>DeGeOP</i> . Di conseguenza è anche il linguaggio utilizzato maggiormente dalle librerie esterne da noi sfruttate.
Documentazione ufficiale	http://www.ecma-international.org/ecma-262/6.0/

2.4 JSON

Descrizione	Formato dati utilizzato per lo scambio di informazioni tra il client (ovvero il nostro prodotto) e il server (ovvero il prodotto di <i>RiskApp</i>).
Utilizzo	Viene utilizzato per lo scambio di dati tra l'applicazione <i>DeGeOPe</i> e il server di <i>RiskApp</i> .
Documentazione ufficiale	http://www.json.org

2.5 JSX

Descrizione	JSX è un linguaggio orientato agli oggetti staticamente tipizzato. È un'estensione di JavaScript. I file in linguaggio JSX vengono poi tradotti in JavaScript.
Utilizzo	Viene utilizzato come sintassi all'interno di React.
Documentazione ufficiale	https://facebook.github.io/react/docs/introducing-jsx.html

2.6 Node.js

Descrizione	Ambiente operativo per utilizzare JavaScript in ambito server.
Utilizzo	Viene utilizzato per far avviare la nostra applicazione.
Documentazione ufficiale	https://nodejs.org/it/docs/
Versione installata	6.9.1

2.7 OpenLayers

Descrizione	E' una libreria JavaScript per visualizzare mappe interattive nei browser web. OpenLayers offre API ai programmatori per poter accedere a diverse fonti d'informazioni cartografiche in Internet: mappe del progetto OpenStreetMap, mappe sotto licenze non-libere (Google Maps, Bing, Yahoo), Web Feature Service, ecc. E' coperto da licenza BSD.
Utilizzo	Viene utilizzato per gestire la mappa.
Documentazione ufficiale	http://openlayers.org/en/latest/doc/
Versione installata	4.0.1

2.8 Open Street Map

Descrizione	E' una libreria JavaScript per fornire informazioni geografiche in formato vettoriale.
Utilizzo	OpenStreetMap viene utilizzato in modo indiretto dalla libreria OpenLayers per fornire una vista vettoriale nella mappa dell'applicazione.
Documentazione ufficiale	https://www.openstreetmap.org/help

2.9 React

Descrizione	E' una libreria JavaScript <i>open source</i> mantenuta da Facebook e Instagram utile alla costruzione di interfacce grafiche. Per fare ciò, React utilizza componenti indipendenti e riusabili che ereditano dalla classe base astratta React.Component. Le componenti devono implementare il metodo <code>render()</code> che si occupa di rappresentare la <i>componente</i> sul browser. Le caratteristiche più importanti di questa libreria sono: <ul style="list-style-type: none"> • One-way-data-flow: meccanismo tramite il quale le proprietà (un insieme di valori immutabili passato al render di un componente) non possono essere direttamente modificate. Queste proprietà possono però essere modificate da una <i>callback</i>; • Virtual DOM: virtualizzazione operata da React per effettuare un re-rendering efficiente dei componenti. Consiste in: <ul style="list-style-type: none"> - replicare il DOM in memoria; - individuare le differenze tra il DOM reale e il DOM virtuale; - aggiornare le informazioni del DOM reale sulla base delle differenze precedentemente individuate. • utilizzo di JSX.
Utilizzo	React viene utilizzata per la costruzione dell'interfaccia grafica dell'applicazione.
Documentazione ufficiale	https://facebook.github.io/react/docs/hello-world.html
Versione installata	15.4.2

2.10 ReactColor

Descrizione	E' una libreria JavaScript per creare una palette di colori RGB.
Utilizzo	Viene utilizzata per la creazione di un color picker.
Documentazione ufficiale	https://casesandberg.github.io/react-color/
Versione installata	2.11.3

2.11 React-Redux

Descrizione	Libreria che facilita l'integrazione tra Redux e React.
Utilizzo	Le classi JavaScript vengono passate ad una funzione della libreria per ottenere una nuova classe che sfrutta React-Redux.
Documentazione ufficiale	http://redux.js.org/docs/basics/UsageWithReact.html
Versione installata	5.0.3

2.12 React Toolbox

Descrizione	E' una libreria JavaScript composta da un insieme di componenti React che implementano la specifica del Material Design di Google.
Utilizzo	React Toolbox viene utilizzata per implementare alcune le componenti grafiche secondo la specifica del Material Design.
Documentazione ufficiale	http://react-toolbox.com/#/components
Versione installata	2.0.0-beta.7

2.13 Redux

Descrizione	Libreria per l'implementazione dell'architettura che si occupa di gestire le interazioni tra la business logic e la presentazione. Per fare ciò:
	<ul style="list-style-type: none">• implementa un <i>design pattern</i> architettonico da usare il sostituzione a MVC, come descritto in ??;• offre delle API apposite per la gestione degli elementi del design pattern descritto al punto precedente.
Utilizzo	Redux viene utilizzato per implementare l'architettura di <i>DeGeOP</i> .
Documentazione ufficiale	http://redux.js.org
Versione installata	3.6.0

2.14 SVG

Descrizione	Standard per la scrittura di immagini in formato vettoriale.
Utilizzo	SVG viene utilizzato per aggiungere oggetti personalizzati alla mappa.
Documentazione ufficiale	https://www.w3.org/Graphics/SVG/
Versione installata	0.1.0

2.15 Axios

Descrizione	Libreria JavaScript per la gestione di richieste HTTP.
Utilizzo	Viene utilizzato per effettuare le chiamate verso il server di Riskapp.
Documentazione ufficiale	https://github.com/mzabriskie/axios
Versione installata	0.16.2

3 Configurazione ambiente di lavoro

In questa sezione verranno descritti gli strumenti utilizzati e consigliati per lo sviluppo dell'applicazione. Inoltre verranno descritte delle procedure per l'installazione e la configurazione degli stessi.

3.1 Requisiti di sistema

Essendo *DeGeOP* un'applicazione web, è necessaria una connessione ad internet per poterla utilizzare.

3.1.1 Dispositivi e browser supportati

Al momento della stesura di questo documento l'applicazione è supportata dai seguenti dispositivi:

- Window v10
- Linux Ubuntu10.04 e Debian 8.0
- MacOS v10.2
- Tablet - Asus P01MA con sistema operativo Android 6.0.1
- Tablet - iPad Air 2 con sistema operativo iOS 10.2

e dai seguenti browser:

- Chome v55
- Safari v10
- Firefox v5.3 per iOS 10.2
- Firefox v50

3.2 IDE: WebStorm

WebStormG è un IDE fornito dall'azienda JetBrains per lo sviluppo di applicazioni e siti web. Questo software permette di supportare lo sviluppatore nella realizzazione di prodotti web. Tra le principali funzionalità sono presenti:

- assistenza nella codifica dei principali framework web;
- auto-completamento per la sintassi di molti linguaggi (es. JavaScript, HTML, CSS);
- gestione di progetti complessi;
- integrazione con i più popolari tool per lo sviluppo web;
- integrazione di un sistema di debugging per il codice JavaScript;
- compatibilità con sistemi di build automatico.

Questo strumento è stato utilizzato durante l'attività di codifica e le principali motivazioni che hanno portato il gruppo alla sua scelta sono:

- strumento completo e professionale;
- licenza gratuita per studenti;
- integrabilità con strumenti per analisi statica e test;
- integrabilità con GitHub;
- cross-platform.

Indirizzo download <https://www.jetbrains.com/webstorm/download/>

Versione installata 2017.1 o superiore

3.3 Gestore dei pacchetti

3.3.1 Node.js

Node.js è una piattaforma event-driven per il motore JavaScript V8, disponibile sulle principali piattaforme, anche se maggiormente performante su sistemi operativi UNIX-like.

Indirizzo download <https://nodejs.org/en/download/>

Versione installata 6.9.5 o superiore

Consigliamo la versione stabile (LTS) di Node.js e non quella con le ultime features. Inoltre per verificare la versione attualmente installata, eseguire il seguente comando a terminale:

`node -v`

3.3.2 npm

npm (node package manager) è il gestore di pacchetti di default utilizzato da Node.js. Questo strumento consente l'installazione e la gestione di moduli esterni che forniscono funzionalità aggiuntive al sistema node di base, risolvendo varie dipendenze. Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- larga diffusione;
- più di 400.000 pacchetti installabili;
- ampia documentazione.

Indirizzo download <https://www.npmjs.com>

Versione installata 4.2.0 o superiore

Per lo sviluppo di questo progetto è stato creato e configurato un apposito file che richiama l'installazione di tutti i pacchetti necessari. Di seguito viene descritta la procedura per l'installazione di quest'ultimo:

1. aver installato Node.js, oppure installarlo seguendo le istruzioni descritte in [3.3.1](#);
2. aprire un terminale e posizionarsi all'interno della cartella del progetto, in modo da scaricare i pacchetti localmente . Se non presente, scaricarla tramite la procedura descritta nella sezione [3.5](#);
3. scrivere il comando

```
npm install
```

4. attendere il termine dell'installazione, che può durare anche qualche minuto in funzione della banda o della potenza del processore;

Per vedere la lista dei pacchetti installati e controllarne la versione scrivere nel terminale all'interno della cartella del progetto *DeGeOP*:

```
npm list
```

3.4 Browser

3.4.1 Google Chrome

L'indirizzo per il download è valido per i seguenti sistemi operativi: Windows 10/8.1/8/7 64/32-bit, Mac OS X 10.9+, Linux

Indirizzo download	https://www.google.it/chrome/browser/desktop/index.html
Versione installata	55 o superiore

3.4.2 Mozilla Firefox

L'indirizzo per il download è valido per i seguenti sistemi operativi: Windows 10/8.1/8/7 64/32-bit, Mac OS X 10.9+, Linux

Indirizzo download	https://www.mozilla.org/it/firefox/new/?scene=2
Versione installata	50 o superiore

3.4.3 Safari

Indirizzo download	IntegratoneiMac
Versione installata	10 o superiore

3.5 Download del progetto

Il progetto è attualmente ospitato presso una repository privata su GitHub e quindi necessita l'autorizzazione da parte del grippo Zephyrus. Una volta ottenuta, si può procedere con la procedura seguente:

- scaricare il progetto tramite il seguente link:

<https://github.com/JordanGottardo/DeGeOP>

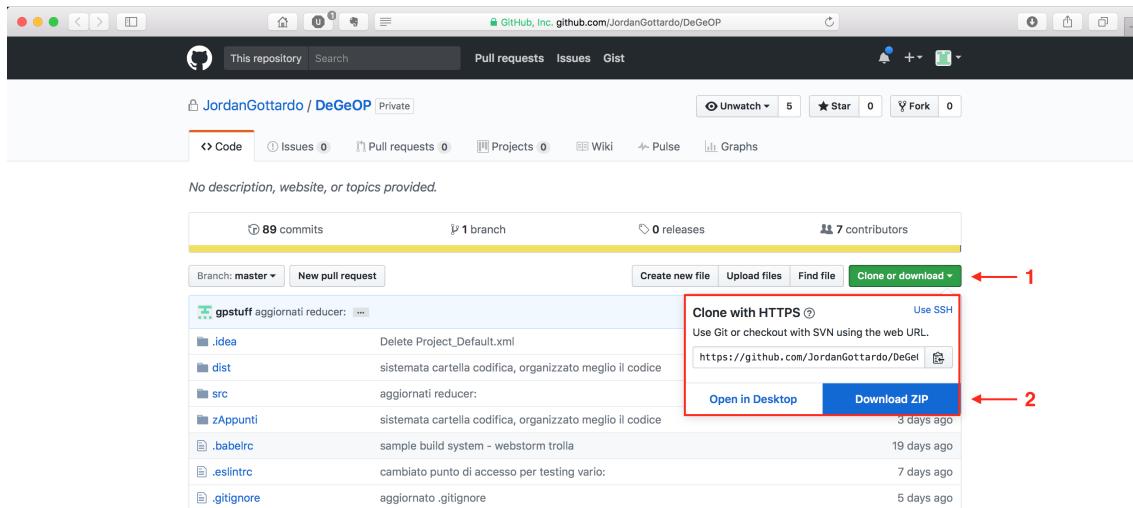


Figura 1: Download progetto DeGeOP dalla repository su GitHub

- decomprimere il file .zip scaricato;
- avviare WebStorm e aprire il progetto, seguendo il menu: **File -> Open** e selezionare la cartella estratta al punto 2;
- (opzionale) dopo aver installato tutti i pacchetti con procedura 3, è possibile configurare ESLint. Di seguito vengono descritti i passi per la configurazione su WebStorm:
 - aprire WebStorm;
 - aprire il menu **File -> Preferences**;
 - entrare nel menu **Language & Frameworks->JavaScript->Code Quality->ESLint**;
 - abilitare il tool essicurandosi di aver installato prima Node.js (vedi sezione 3.3.1);
 - selezionare nel campo ESLint package il file che si trova all'interno del progetto DeGeOp/node_modules/eslint;

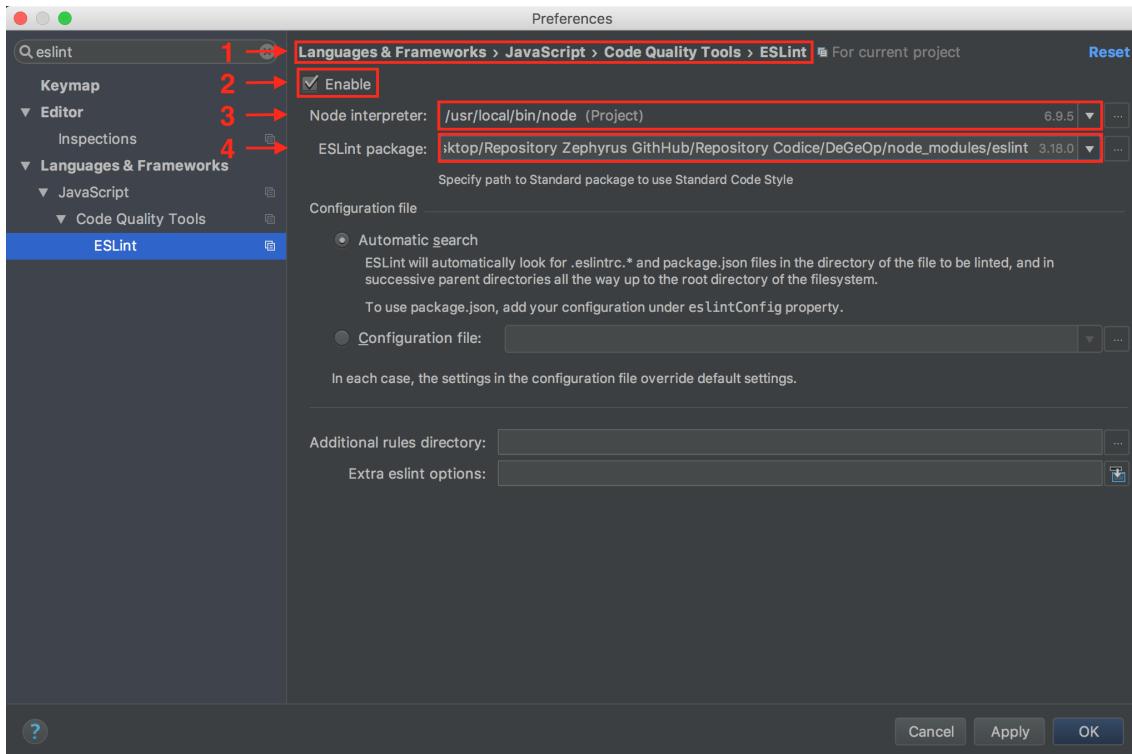


Figura 2: Configurazione ESLint DeGeOPsu WebStorm

3.6 Configurazione della VirtualBox

In accordo con il proponente abbiamo installato una macchina virtuale per effettuare il deploy e testare la nostra applicazione con il server di *RiskApp*. La procedura per configurare la macchina virtuale è la seguente:

1. scaricare e installare sul proprio computer Oracle VirtualBox dal sito ufficiale:

<https://www.virtualbox.org/wiki/Downloads>

2. scaricare il file immagine della macchina virtuale creata da *RiskApp*, dal seguente link, inserendo le apposite credenziali fornite dal proponente:

<https://www.satellite1.info/u/Zephyrus.ova;>

3. aprire l'applicazione VirtualBox sul proprio computer;
4. aprire il menu **File** e selezionare **Import Appliance**;
5. selezionare il file **Zephyrus.ova** scaricato nel punto 2 e cliccare su **Next**;
6. cliccare su **Import** e attendere la fine del processo;
7. terminata l'importazione fare click destro sulla nuova macchina virtuale presente nell'elenco a sinistra e selezionare **Settings**;
8. selezionare la sezione **Network**;
9. nella tab **Adapter 1** modificare l'opzione **Attached to:** in **NAT** e cliccare **OK**;
10. la macchina virtuale è ora pronta all'uso.

4 Architettura DeGeOP

4.1 Introduzione

Il prodotto *DeGeOP*, creato dal *gruppo_G* Zephyrus, sarà integrabile nell'attuale applicazione del proponente *RiskApp*.

Per esporre l'architettura dell'applicazione si procederà con approccio top-down, partendo cioè da una visione generale delle componenti che distinguono il sistema, per poi analizzare in dettaglio la conformazione di tali componenti.

Il sistema attuale di *RiskApp* presenta un'architettura client-server.

4.2 Server

DeGeOP si conserverà al server di *RiskApp* esclusivamente utilizzando le *API_G* REST fornite dal proponente stesso e riportate in questa sezione. Il gruppo non ha quindi accesso all'implementazione del server di *RiskApp*.

4.2.1 Chiamate REST

L'interfaccia REST proposta da *RiskApp* fornisce l'accesso alle seguenti entità:

- Customer
- Graph
- *Asset_G*
- Node
- Edge

Per ognuna di queste è possibile fare una chiamata REST usando i verbi http. (GET, OPTION, POST, PUT, UPDATE, DELETE)

Tutte queste entità sono identificate univocamente da uno uuid formato come una stringa di cifre esadecimale con la seguente struttura:

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Il verbo OPTION viene utilizzato all'interno del server di Riskapp per ottenere informazioni sulla struttura (nomi dei campi dati e relativi tipi di dato) delle entità ricevute a seguito di una chiamata GET o necessarie per le chiamate POST e PUT. Inoltre per determinati campi, la chiamata OPTION restituisce tutti i possibili valori, nel caso siano limitati al lato server, per il campo stesso.

4.2.1.1 Customer

- **descrizione:** l'entità customer contiene le informazioni del cliente;
- **chiamata:** /mitsuko/v01/customer/<uuid>/uuid/
 - **operazioni utilizzate:** GET.

4.2.1.2 Graph

- **descrizione:** l'entità grafo proposta da *RiskApp* contiene le informazioni relative al processo produttivo e ai suoi *nodi_G*;
- **chiamata:** /mitsuko/v01/graph/<uuid>/uuid/
 - **operazioni utilizzate:** GET.

4.2.1.3 Asset

- **descrizione:** l'entità asset rappresenta un asset del cliente con le relative informazioni;
- **chiamata:** /mitsuko/v01/customer/<uuid>/asset/new/
 - **operazioni utilizzate:** POST.
- **chiamata:** /mitsuko/v01/asset/<uuid>/uuid/
 - **operazioni utilizzate:** GET, PUT, OPTIONS, UPDATE, DELETE.

4.2.1.4 Node

- **descrizione:** l'entità node rappresenta un elemento di interesse strategico all'interno di un asset;
- **chiamata:** /mitsuko/v01/graph/<uuid>/node/new/
 - **operazioni utilizzate:** POST.
- **chiamata:** /mitsuko/v01/node/<uuid>/uuid/
 - **operazioni utilizzate:** GET, PUT, OPTIONS, UPDATE, DELETE.

4.2.1.5 Edge

- **descrizione:** l'entità edge rappresenta un collegamento fra due nodi;
- **chiamata:** /mitsuko/v01/graph/<uuid>/edge/new/
 - **operazioni utilizzate:** POST.
- **chiamata:** /mitsuko/v01/edge/<uuid>/uuid/
 - **operazioni utilizzate:** GET, PUT, OPTIONS, UPDATE, DELETE.

4.3 Client

Il proponente ha fornito l'ambiente di sviluppo contenente la loro attuale applicazione, denominata in figura come "RiskApp", su cui il gruppo potrà integrare DeGeOP. Il prodotto sarà sviluppato come una single-page accessibile cliccando sulla scheda di menu "Process and analysis".

4.3.1 Design architetturale di DeGeOP

E' stato scelto di utilizzare il design architetturale Redux (per maggiori informazioni su questo pattern, si veda l'appendice ??). Per una migliore gestione e uso di questo pattern si è deciso di rafforzare l'uso classico di Redux, incapsulando le varie componenti in una struttura a classi. Rispetto all'architettura base, ovvero Redux, sono quindi presenti altre componenti:

- **ActionCreators**: struttura il cui compito è creare le "azioni", ossia le operazioni che si occupano di interagire con lo *store_G*. Le actions come fornite da Redux permettono di definire azioni potenzialmente invalide o la cui esecuzione potrebbe portare ad errori. È quindi nata la necessità di incapsulare il concetto di azione e devolvere il compito della sua creazione ad una *componente_G* propria;
- **Reducer_G**: classe di utilità che accetta *action_G* generiche e le reindirizza alle giuste implementazioni per gestire l'azione in analisi. Anche in questo caso l'incapsulazione è stata adottata per fornire una migliore interfaccia di utilizzo dei reducer.

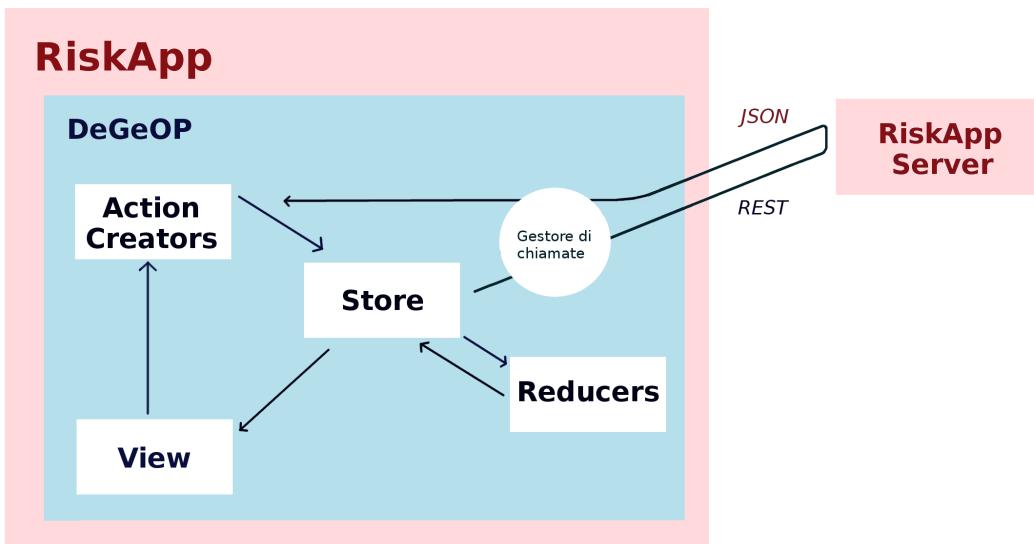


Figura 3: Architettura di base

5 Componenti

5.1 DeGeOP

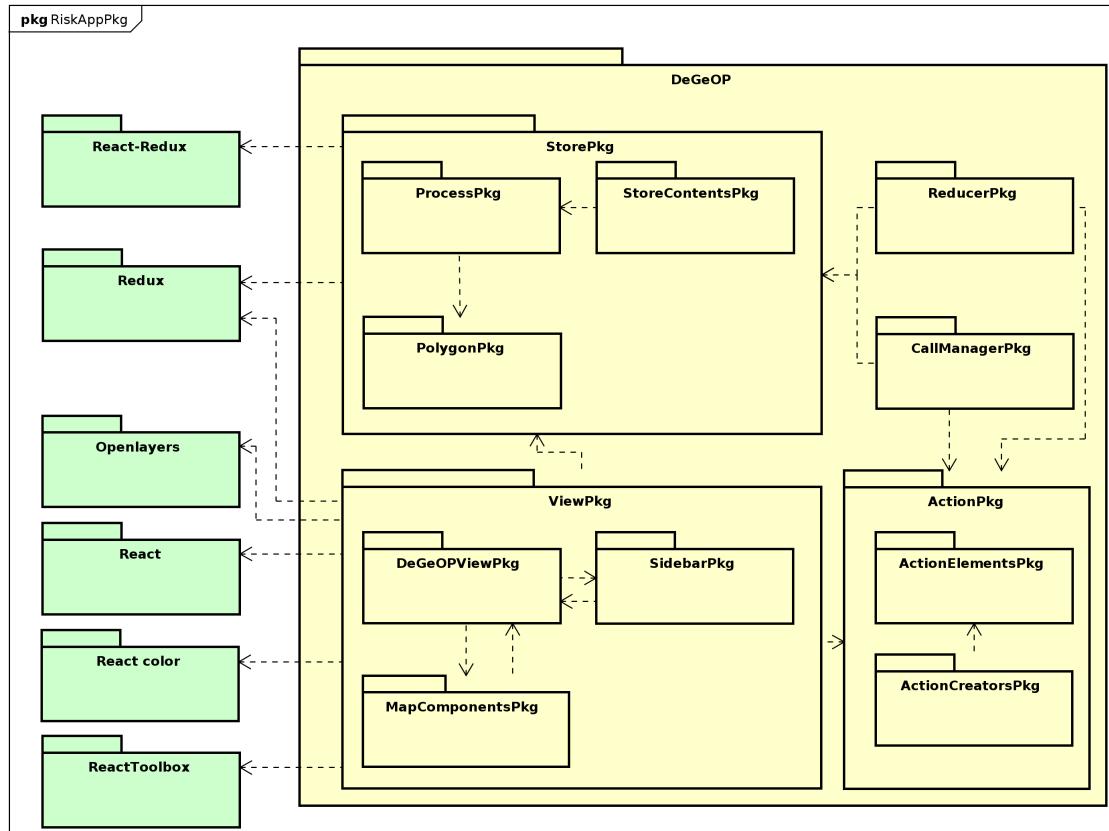


Figura 4: Schema componente DeGeOP

5.1.1 Informazioni sul package

- **descrizione:** racchiude tutte le componenti necessarie per il front-end del prodotto;
- **package contenuti:**
 - DeGeOP::[ActionPkg](#);
 - DeGeOP::[CallManagerPkg](#);
 - DeGeOP::[ReducerPkg](#);
 - DeGeOP::[StorePkg](#);
 - DeGeOP::[ViewPkg](#).

5.2 DeGeOP::StorePkg

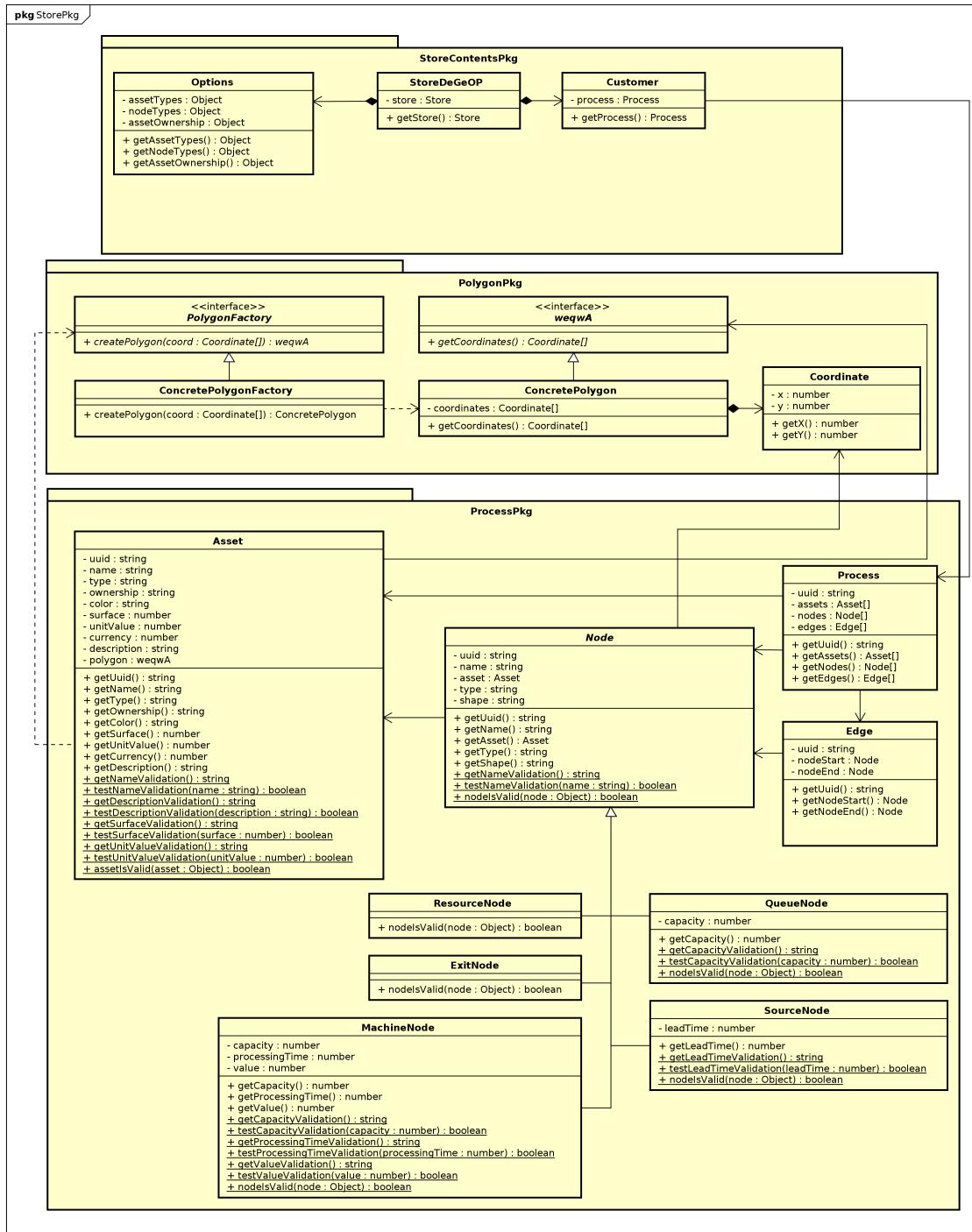


Figura 5: Schema componente DeGeOP::StorePkg

5.2.1 Informazioni sul package

- **descrizione**: racchiude le componenti utilizzate per la memorizzazione e rappresentazione dei dati;
- **padre**: [DeGeOP](#);

- **package contenuti:**

- StorePkg::[PolygonPkg](#);
- StorePkg::[ProcessPkg](#);
- StorePkg::[StoreContentsPkg](#).

- **interazioni con altri package:**

- IN CallManagerPkg: subscribe sullo store;
- IN ReducerPkg: applicazione di cambiamenti di stato;
- IN ViewPkg: subscribe sullo store;
- OUT React-Redux: utilizzo di Provider per evitare di passare lo store come proprietà alle componenti React;
- OUT Redux: creazione Store utilizzando il metodo createStore().

5.3 DeGeOP::StorePkg::StoreContentsPkg

5.3.1 Informazioni sul package

- **descrizione:** racchiude le componenti che implementano il concetto di store dell'architettura Redux;
- **padre:** [StorePkg](#);
- **interazioni con altri package:**
 - OUT AnalysisPkg: riferimento ad analisi di danno;
 - OUT ProcessPkg: riferimento a processo;
 - OUT React-Redux: utilizzo del Provider;
 - OUT Redux: creazione store.
- **classi contenute:**
 - Customer;
 - Options;
 - StoreDeGeOP.

5.3.2 Classi

5.3.2.1 Customer

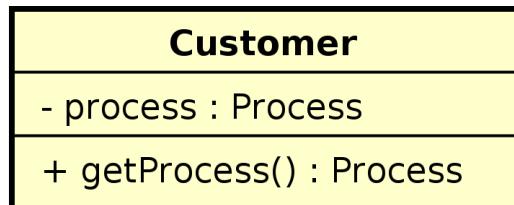


Figura 6: Diagramma classe Customer

- **descrizione:** rappresenta l'assicurando;
- **utilizzo:** viene utilizzato nello Store per memorizzare l'assicurando;
- **attributi:**
 - process : Process
 - * rappresenta il processo produttivo dell'assicurando.
- **metodi:**
 - +getProcess() : Process
 - il metodo restituisce il processo produttivo dell'assicurando

5.3.2.2 Options

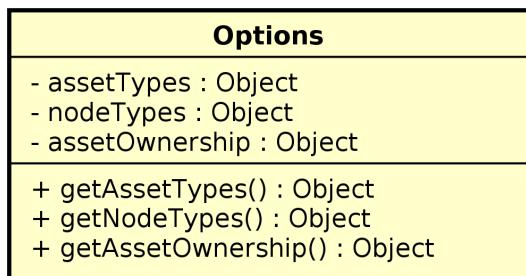


Figura 7: Diagramma classe Options

- **descrizione:** classe contenente le associazioni tra codice e valore dei campi dati degli altri elementi dello store;
- **utilizzo:** viene utilizzata per associare codice e valore dei campi dati degli altri elementi dello store;
- **attributi:**
 - -assetOwnership : Object
 - * associazione chiave valore delle tipologie di soggetti di appartenenza dell'asset.
 - -assetTypes : Object
 - * associazione chiave valore delle tipologie di materiale dell'asset.
 - -nodeTypes : Object
 - * associazione chiave valore delle tipologie di materiale del nodo.
- **metodi:**
 - +getAssetOwnership() : Object
 - il metodo ritorna l'oggetto assetOwnership
 - +getAssetTypes() : Object
 - il metodo ritorna l'oggetto assetTypes
 - +getNodeTypes() : Object
 - il metodo ritorna l'oggetto nodeTypes
- **relazioni con altre classi:**
 - IN OptionReducer;
 - IN StoreDeGeOP.

5.3.2.3 StoreDeGeOP



Figura 8: Diagramma classe StoreDeGeOP

- **descrizione:** rappresenta una classe che incapsula uno Store creato utilizzando Redux;
- **utilizzo:** viene utilizzato per memorizzare lo stato dell'applicazione. Le componenti che effettuano il subscribe sullo Store verranno notificate ad ogni cambiamento di stato dello Store;
- **attributi:**
 - -customer : Customer
 - * rappresenta il cliente che gestiamo.
 - -store : Object
 - * rappresenta l'oggetto Store creato utilizzando Redux.
- **metodi:**
 - +getStore() : Object
 - il metodo restituisce lo store creato utilizzando Redux
- **relazioni con altre classi:**
 - IN Reducer;
 - OUT Options.

5.4 DeGeOP::StorePkg::ProcessPkg

5.4.1 Informazioni sul package

- **descrizione:** racchiude le componenti necessarie alla rappresentazione del processo produttivo dell'assicurando;
- **padre:** [StorePkg](#);
- **interazioni con altri package:**
 - IN StoreContentsPkg: riferimento a processo;
 - OUT PolygonPkg: riferimento ad un poligono.
- **classi contenute:**
 - Asset;
 - Edge;
 - ExitNode;
 - MachineNode;
 - Node;
 - Process;
 - QueueNode;
 - ResourceNode;
 - SourceNode.

5.4.2 Classi

5.4.2.1 Asset



Figura 9: Diagramma classe Asset

- **descrizione:** rappresenta un fabbricato di interesse per il processo produttivo dell'assicurando;
- **utilizzo:** sono contenuti all'interno di Process;
- **attributi:**
 - -color : string
 - * indica il colore dell'asset.
 - -currency : number
 - * indica la valuta utilizzata.
 - -description : string
 - * rappresenta la descrizione dell'asset.
 - -name : string
 - * rappresenta il nome dell'asset.
 - -ownership : string
 - * rappresenta l'appartenenza dell'asset.
 - -surface : number
 - * rappresenta la dimensione, in mq, dell'asset.
 - -type : string
 - * rappresenta la tipologia dell'asset.
 - -unitValue : string
 - * indica il valore economico dell'asset.
 - -uuid : string
 - * rappresenta l'identificativo dell'asset (uuid).
- **metodi:**
 - +assetIsValid(asset) : boolean
 - il metodo testa che i parametri con cui l'asset sta per essere creato siano validi
 - * asset : Object
 - oggetto contenente i parametri di un Asset che dovrà essere validato.
 - +getColor() : string
 - il metodo ritorna il colore dell'asset
 - +getCurrency() : number
 - il metodo ritorna la valuta utilizzata per l'asset
 - +getDescription() : string
 - il metodo ritorna la descrizione dell'asset
 - +getDescriptionValidation() : string
 - il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 5000 caratteri; inizia e/o finisce con uno spazio; contiene caratteri speciali diversi dall'apostrofo
 - +getName() : string
 - ritorna il nome dell'asset

- `+getNameValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: più lunga di 50 caratteri; inizia e/o finisce con uno spazio; contiene caratteri speciali.
- `+getOwnership() : string`
il metodo ritorna l'appartenenza dell'asset
- `+getSurface() : number`
il metodo ritorna la dimensione dell'asset
- `+getSurfaceValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 5 cifre per la parte intera; più di 2 per la parte decimale;
- `+getType() : string`
il metodo ritorna la tipologia dell'asset
- `+getUnitValue() : string`
il metodo ritorna il valore economico dell'asset
- `+getUnitValueValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 5 cifre per la parte intera; più di 2 per la parte decimale
- `+getUuid() : string`
il metodo ritorna l'uuid dell'asset
- `+testDescriptionValidation(description) : boolean`
il metodo ritorna true solamente se il valore ricevuto in input, come parametro, è ritenuto valido.
 - * `description : string`
rappresenta la descrizione dell'asset.
- `+testNameValidation(name) : boolean`
il metodo ritorna true solamente se il valore ricevuto in input, come parametro, è ritenuto valido.
 - * `name : string`
rappresenta il nome dell'asset.
- `+testSurfaceValidation(surface) : boolean`
il metodo ritorna true solamente se il valore ricevuto in input, come parametro, è ritenuto valido.
 - * `surface : number`
rappresenta la dimensione, in mq, dell'asset.
- `+testUnitValueValidation(unitValue) : boolean`
il metodo ritorna true solamente se il valore ricevuto in input, come parametro, è ritenuto valido.
 - * `unitValue : number`
indica il valore economico dell'asset.

- **relazioni con altre classi:**

- IN AssetReducer.

5.4.2.2 Edge

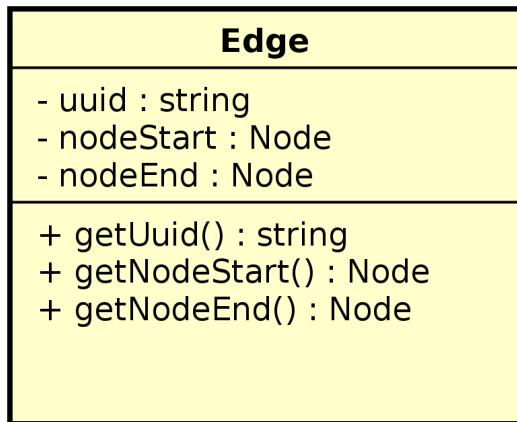


Figura 10: Diagramma classe Edge

- **descrizione:** rappresenta un arco che collega due nodi tra di loro; un arco indica che i nodi sono in correlazione tra di loro;
- **utilizzo:** è contenuto all'interno di Process;
- **attributi:**
 - -nodeEnd : Node
 - * rappresenta il nodo di arrivo.
 - -nodeStart : Node
 - * rappresenta il nodo di partenza.
 - -uuid : string
 - * rappresenta il codice identificativo dell'arco (uuid).
- **metodi:**
 - +getNodeEnd() : Node
 - il metodo restituisce il nodo di arrivo
 - +getNodeStart() : Node
 - il metodo restituisce il nodo di partenza
 - +getUuid() : string
 - il metodo restituisce il codice identificativo dell'arco (uuid)
- **relazioni con altre classi:**
 - IN EdgeReducer.

5.4.2.3 ExitNode

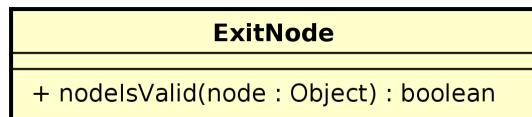


Figura 11: Diagramma classe ExitNode

- **descrizione:** rappresenta un nodo di tipo Uscita;
- **utilizzo:** è contenuto all'interno di Process;
- **metodi:**
 - `+nodeisValid() : boolean`
il metodo testa che i parametri con cui il nodo sta per essere creato siano validi

5.4.2.4 MachineNode

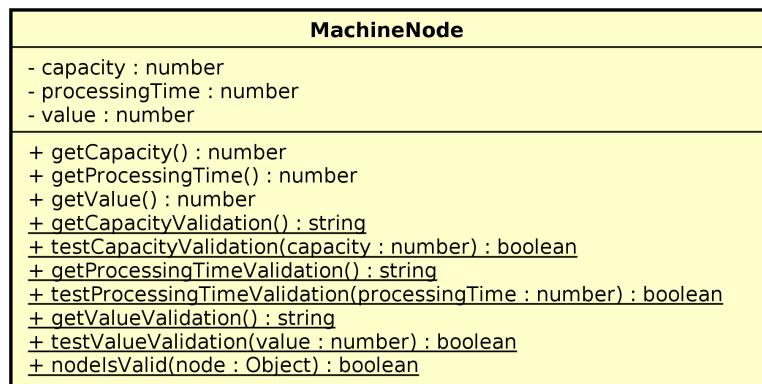


Figura 12: Diagramma classe MachineNode

- **descrizione:** rappresenta un nodo di tipo Macchina;
- **utilizzo:** è contenuto all'interno di Process;
- **attributi:**
 - `-capacity : number`
 - * rappresenta la capacità del nodo.
 - `-processingTime : number`
 - * rappresenta il tempo di processo del nodo.
 - `-value : number`
 - * rappresenta il valore del nodo.
- **metodi:**

- `+getCapacity() : number`
il metodo ritorna la capacità del nodo
- `+getCapacityValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 5 cifre per la parte intera; più di 2 per la parte decimale;
- `+getProcessingTime() : number`
il metodo ritorna il tempo di processo del nodo
- `+getProcessingTimeValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 5 cifre per la parte intera; più di 2 per la parte decimale;
- `+getValue() : number`
il metodo ritorna il valore del nodo
- `+getValueValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 20 cifre per la parte intera; più di 2 per la parte decimale
- `+nodeisValid(node) : boolean`
il metodo testa se i parametri con cui il nodo sta per essere creato sono validi
 - * `node : Object`
oggetto che rappresenta i parametri con cui il nodo sta per essere creato.
- `+testCapacityValidation() : boolean`
il metodo testa se la capacità del nodo valida
- `+testProcessingTimeValidation(processingTime) : boolean`
il metodo testa se il tempo di processo valida
 - * `processingTime : number`
rappresenta il tempo di processo del nodo.
- `+testValueValidation(value) : boolean`
il metodo testa se il valore del nodo valida
 - * `value : number`
rappresenta il valore del nodo.

5.4.2.5 Node

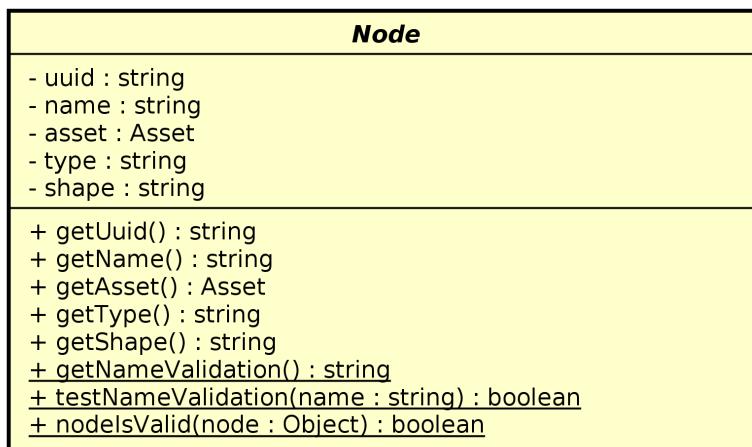


Figura 13: Diagramma classe Node

- **descrizione:** rappresenta un nodo contenuto all'interno di un Asset ;
- **utilizzo:** è contenuto all'interno di Process;
- **attributi:**
 - -asset : Asset
 - * rappresenta l'asset a cui il nodo appartiene.
 - -name : string
 - * rappresenta il nome del nodo.
 - -shape : string
 - * rappresenta la forma del nodo.
 - -type : string
 - * rappresenta la classe del nodo.
 - -uuid : string
 - * rappresenta il codice identificativo del nodo (uuid).
- **metodi:**
 - +getAsset() : Asset
 - il metodo ritorna l'asset di appartenenza del nodo
 - +getName() : string
 - il metodo ritorna il codice identificativo del nodo
 - +getNameValidation() : string
 - il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 50 caratteri; inizia e/o finisce con uno spazio; contiene caratteri speciali;
 - +getShape() : string
 - il metodo ritorna la forma del nodo

- `+getType() : string`
il metodo ritorna la classe del nodo
 - `+getUuid() : string`
il metodo ritorna il codice identificativo del nodo (uuid)
 - `+nodesisValid() : boolean`
il metodo testa che i parametri con cui il nodo sta per essere creato siano validi
 - `+testNameValidation(name) : boolean`
il metodo ritorna true solamente se il valore ricevuto in input, come parametro, è ritenuto valido.
 - * `name : string`
rappresenta il nome del nodo.
- **relazioni con altre classi:**
- IN NodeReducer.

5.4.2.6 Process

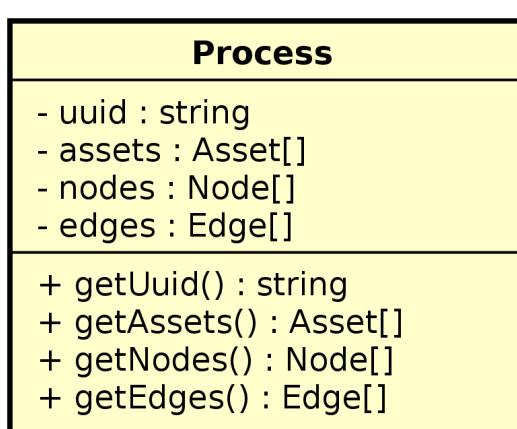


Figura 14: Diagramma classe Process

- **descrizione:** rappresenta un processo produttivo dell'azienda dell'assicurando;
- **utilizzo:** è memorizzato nello Store;
- **attributi:**
 - `-assets : Asset[]`
 - * rappresenta gli asset facenti parti del processo produttivo.
 - `-edges : Edge []`
 - * rappresenta gli archi che effettuano i collegamenti nel processo produttivo.
 - `-nodes : Node []`
 - * rappresenta i nodi facenti parti del processo produttivo.
 - `-uuid : string`
 - * rappresenta l'identificativo del processo (uuid).

- **metodi:**

- `+getAssets() : Asset[]`
il metodo ritorna gli asset facenti parti del processo produttivo
- `+getEdges() : Edge []`
il metodo ritorna gli archi che effettuano i collegamenti nel processo produttivo
- `+getNodes() : Node []`
il metodo ritorna i nodi facenti parti del processo produttivo
- `+getUuid() : string`
il metodo ritorna il codice identificativo (uuid) del processo

5.4.2.7 QueueNode

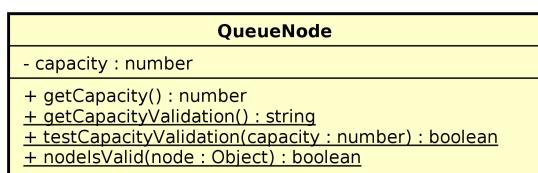


Figura 15: Diagramma classe QueueNode

- **descrizione:** rappresenta un nodo di tipo Coda;

- **utilizzo:** è contenuto all'interno di Process;

- **attributi:**

- `-capacity : number`
* rappresenta la capacità del nodo coda.

- **metodi:**

- `+getCapacity() : number`
il metodo ritorna la capacità del nodo
- `+getCapacityValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 5 cifre per la parte intera; più di 2 per la parte decimale;
- `+nodeisValid(node : Object)` : boolean
il metodo testa che i parametri con cui il nodo sta per essere creato siano validi
 - * `node : Object`
oggetto che rappresenta i parametri con cui il nodo sta per essere creato.
- `+testCapacityValidation(capacity : number) : boolean`
il metodo ritorna true solamente se il valore ricevuto in input, come parametro, è ritenuto valido.
 - * `capacity : number`
rappresenta la capacità del nodo.

5.4.2.8 ResourceNode

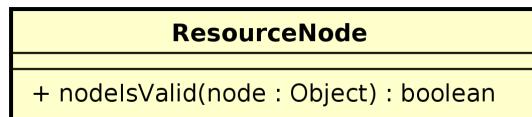


Figura 16: Diagramma classe ResourceNode

- **descrizione:** rappresenta un nodo di tipo Risorsa;
- **utilizzo:** è contenuto all'interno di Process;
- **metodi:**
 - `+nodeisValid() : boolean`
il metodo testa che i parametri con cui il nodo sta per essere creato siano validi

5.4.2.9 SourceNode

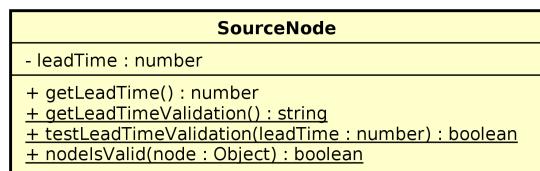


Figura 17: Diagramma classe SourceNode

- **descrizione:** rappresenta un nodo di tipo Sorgente;
- **utilizzo:** è contenuto all'interno di Process;
- **attributi:**
 - `-leadTime : Object`
* rappresenta il tempo di approvvigionamento del nodo.
- **metodi:**
 - `+getLeadTime() : number`
il metodo ritorna il tempo di approvvigionamento del nodo
 - `+getLeadTimeValidation() : string`
il metodo ritorna l'espressione regolare che indica una stringa non contenente le seguenti caratteristiche: vuota; più lunga di 5 cifre per la parte intera; più di 2 per la parte decimale;
 - `+nodeisValid(node) : boolean`
il metodo testa se i parametri con cui il nodo sta per essere creato sono validi
 - * `node : Object`
oggetto che rappresenta i parametri con cui il nodo sta per essere creato.

- `+testLeadTimeValidation(leadTime) : boolean`
 - il metodo testa se il tempo di approvvigionamento è valido
 - * `leadTime : number`
 - rappresenta il tempo di approvvigionamento.

5.5 DeGeOP::StorePkg::PolygonPkg

5.5.1 Informazioni sul package

- **descrizione:** racchiude le componenti necessarie alla rappresentazione dell'area degli asset e degli scenari di danno;
- **padre:** [StorePkg](#);
- **interazioni con altri package:**
 - IN AnalysisPkg: riferimento ad un poligono;
 - IN ProcessPkg: riferimento ad un poligono.
- **classi contenute:**
 - [ConcretePolygon](#);
 - [ConcretePolygonFactory](#);
 - [Coordinate](#);
 - [Polygon](#);
 - [PolygonFactory](#).

5.5.2 Classi

5.5.2.1 ConcretePolygon

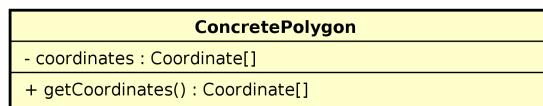


Figura 18: Diagramma classe ConcretePolygon

- **descrizione:** rappresenta un poligono;
- **utilizzo:** viene istanziata da [ConcretePolygonFactory](#); viene utilizzata in Asset e Scenario;
- **attributi:**
 - coordinates : Coordinate[]
 - * rappresenta le coordinate che vanno a delineare il poligono.
- **metodi:**
 - +getCoordinates() : Coordinate[]
 - il metodo restituisce le coordinate che vanno a delineare il poligono

5.5.2.2 ConcretePolygonFactory

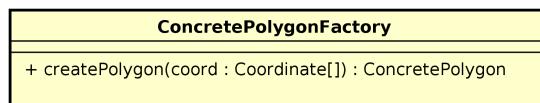


Figura 19: Diagramma classe ConcretePolygonFactory

- **descrizione:** gestisce la creazione concreta dei poligoni;
- **utilizzo:** implementazione di PolygonFactory; è la classe concreta da istanziare per gestire la creazione di un poligono;
- **metodi:**
 - `+createPolygon() : ConcretePolygon`
il metodo si occupa della costruzione del poligono

5.5.2.3 Coordinate

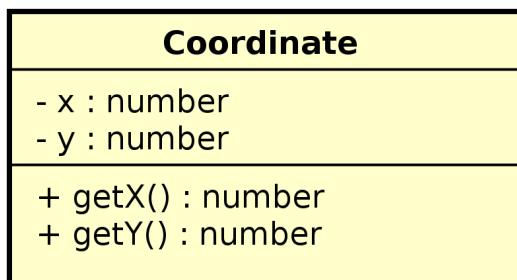


Figura 20: Diagramma classe Coordinate

- **descrizione:** rappresenta una coordinata geografica;
- **utilizzo:** è utilizzata all'interno di Polygon per delimitarne i suoi vertici;
- **attributi:**
 - `-x : number`
 - * rappresenta la latitudine di una coordinata geografica.
 - `-y : number`
 - * rappresenta la longitudine di una coordinata geografica.
- **metodi:**
 - `+getX() : number`
restituisce la latitudine della coordinata geografica
 - `+getY() : number`
restituisce la longitudine della coordinata geografica

5.5.2.4 Polygon

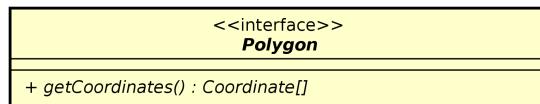


Figura 21: Diagramma classe Polygon

- **descrizione:** interfaccia che rappresenta il poligono;
- **utilizzo:** fornisce i metodi del poligono;
- **metodi:**
 - `+getCoordinates() : Coordinate[]`
il metodo restituisce un array di coordinate

5.5.2.5 PolygonFactory

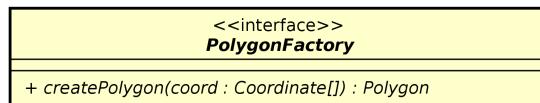


Figura 22: Diagramma classe PolygonFactory

- **descrizione:** interfaccia che si occupa della costruzione dei poligoni;
- **utilizzo:** viene usata dalle classi Scenario e Asset per la costruzione dei poligoni;
- **metodi:**
 - `+createPolygon(coord) : Polygon`
il metodo si occupa della costruzione del poligono
 - * `coord : Coordinate[]`
raccoglie le coordinate necessarie alla costruzione dei poligoni .

5.6 DeGeOP::ReducerPkg

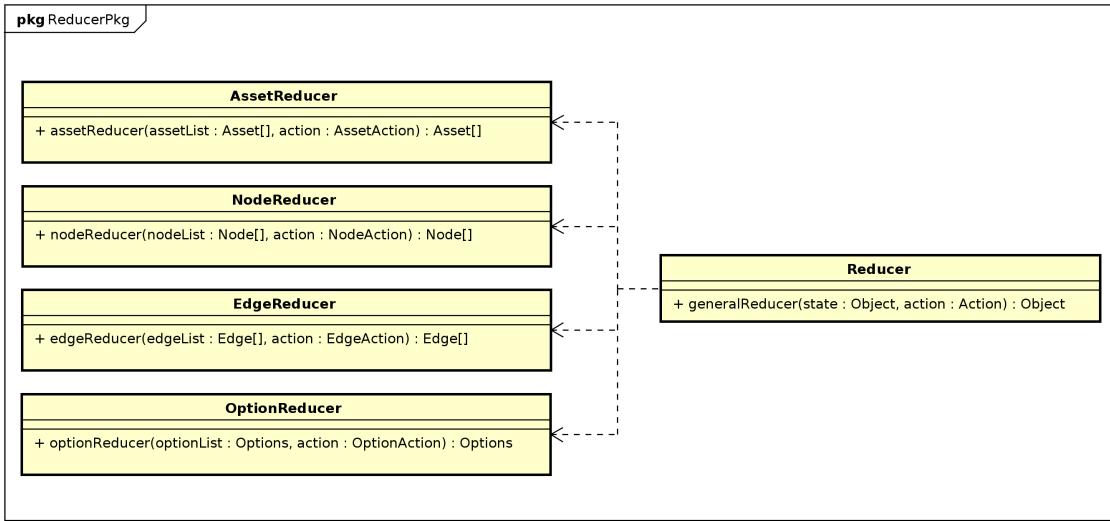


Figura 23: Schema componente DeGeOP::ReducerPkg

5.6.1 Informazioni sul package

- **descrizione:** racchiude le componenti necessarie all'implementazione dei reducer secondo l'architettura Redux;
- **padre:** [DeGeOP](#);
- **interazioni con altri package:**
 - OUT ActionPkg: utilizzo di azioni ;
 - OUT StorePkg: applicazione di cambiamenti di stato.
- **classi contenute:**
 - AssetReducer;
 - EdgeReducer;
 - NodeReducer;
 - OptionReducer;
 - Reducer.

5.6.2 Classi

5.6.2.1 AssetReducer

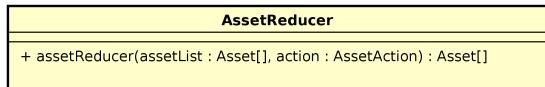


Figura 24: Diagramma classe AssetReducer

- **descrizione:** rappresenta il reducer dell'asset;

- **utilizzo:** il suo metodo gestisce le operazioni sullo store riguardanti l'asset;
- **metodi:**
 - `+assetReducer(action, assetList) : Asset[]`
il metodo gestisce le operazioni sullo store riguardanti l'asset e ritorna la nuova lista di asset ottenuta a seguito della modifica
 - * `action : AssetAction`
rappresenta un'azione che descrive i cambiamenti da effettuare sullo stato.
 - * `assetList : Asset[]`
rappresenta una lista di asset, che sono il vecchio stato.
- **relazioni con altre classi:**
 - OUT Asset.

5.6.2.2 EdgeReducer

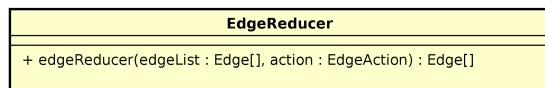


Figura 25: Diagramma classe EdgeReducer

- **descrizione:** rappresenta il reducer dell'arco;
- **utilizzo:** il suo metodo gestisce le operazioni sullo store riguardanti gli archi;
- **metodi:**
 - `+edgeReducer(action, edgeList) : Edge[]`
il metodo gestisce le operazioni sullo store riguardanti gli archi e ritorna la nuova lista di archi ottenuta a seguito della modifica
 - * `action : EdgeAction`
rappresenta un'azione che descrive i cambiamenti da effettuare sullo stato.
 - * `edgeList : Edge[]`
rappresenta una lista di archi, che sono il vecchio stato.
- **relazioni con altre classi:**
 - OUT Edge.

5.6.2.3 NodeReducer

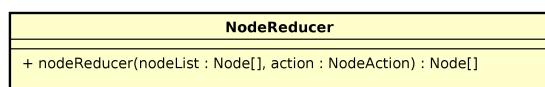


Figura 26: Diagramma classe NodeReducer

- **descrizione:** rappresenta il reducer del nodo;

- **utilizzo:** il suo metodo gestisce le operazioni sullo store riguardanti i nodi;
- **metodi:**
 - `+nodeReducer(action, nodeList) : Node []`
il metodo gestisce le operazioni sullo store riguardanti i nodi e ritorna la nuova lista di nodi ottenuta a seguito della modifica
 - * `action : EdgeAction`
rappresenta un'azione che descrive i cambiamenti da effettuare sullo stato.
 - * `nodeList : Node []`
rappresenta una lista di nodi, che sono il vecchio stato.
- **relazioni con altre classi:**
 - OUT Node.

5.6.2.4 OptionReducer

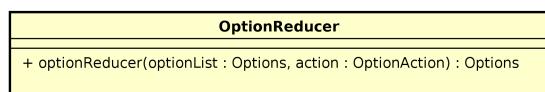


Figura 27: Diagramma classe OptionReducer

- **descrizione:** rappresenta il reducer delle option;
- **utilizzo:** il suo metodo gestisce le operazioni sullo store riguardanti l'asset;
- **metodi:**
 - `+optionReducer(action, optionList) : Options`
il metodo gestisce le operazioni sullo store riguardanti l'asset e ritorna la nuova lista di asset ottenuta a seguito della modifica
 - * `action : OptionAction`
rappresenta un'azione che descrive i cambiamenti da effettuare sullo stato.
 - * `optionList : Options`
rappresenta la lista delle option.
- **relazioni con altre classi:**
 - IN Reducer;
 - OUT Options.

5.6.2.5 Reducer

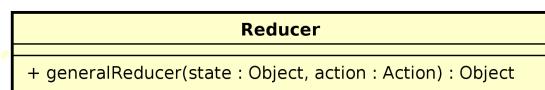


Figura 28: Diagramma classe Reducer

- **descrizione:** accetta una action generica in input e la reindirizza al giusto reducer;

- **utilizzo:** il suo metodo viene utilizzato per catturare un'azione e generare un nuovo stato sullo Store;
- **metodi:**
 - `+generalReducer() : Object`
Esegue l'azione sullo stato corrente dello store e ne ritorna il nuovo stato
- **relazioni con altre classi:**
 - OUT OptionReducer;
 - OUT StoreDeGeOP.

5.7 DeGeOP::CallManagerPkg

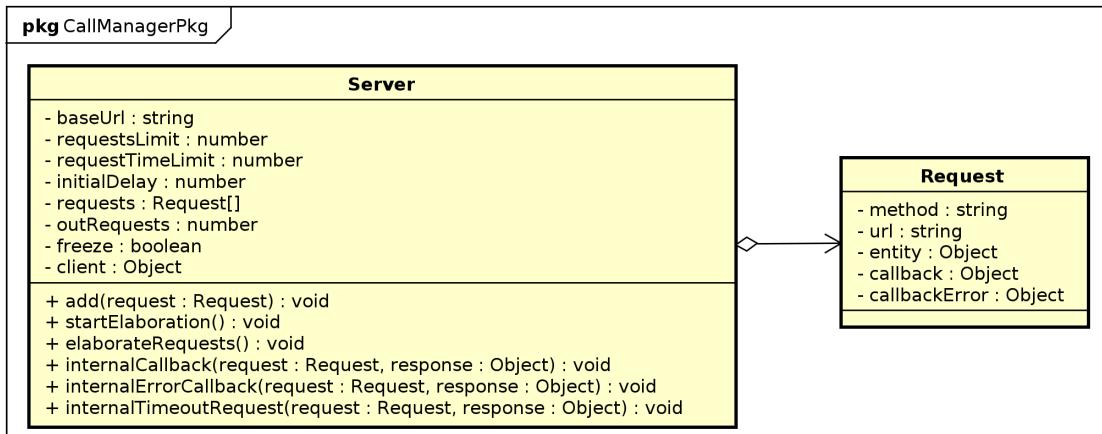


Figura 29: Schema componente DeGeOP::CallManagerPkg

5.7.1 Informazioni sul package

- **descrizione:** racchiude le componenti necessarie alla comunicazione dei dati verso il server;
- **padre:** [DeGeOP](#);
- **interazioni con altri package:**
 - OUT ActionPkg: dispatch di azioni;
 - OUT StorePkg: subscribe sullo store.
- **classi contenute:**
 - Request;
 - Server.

5.7.2 Classi

5.7.2.1 Request

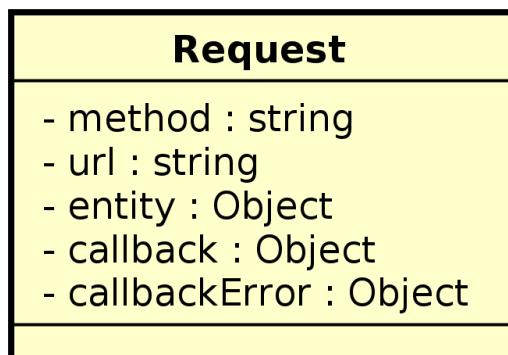


Figura 30: Diagramma classe Request

- **descrizione:** rappresenta i dati necessari per eseguire una richiesta di tipologia REST;
- **utilizzo:** è utilizzata all'interno di Server per gestire la coda delle richieste ;
- **attributi:**
 - callback : Object
 - * rappresenta la funzione che viene invocata se la richiesta è andata a buon fine.
 - callbackError : Object
 - * rappresenta la funzione che viene invocata se la richiesta non è andata a buon fine.
 - entity : Object
 - * rappresenta il payload della richiesta.
 - method : string
 - * rappresenta il verbo http con cui si esegue la richiesta.
 - url : string
 - * rappresenta l'url verso cui eseguire la richiesta.

5.7.2.2 Server

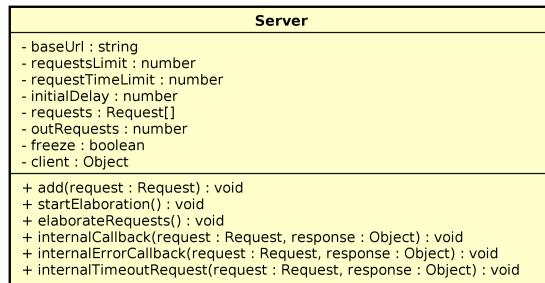


Figura 31: Diagramma classe Server

- **descrizione:** rappresenta il gestore delle chiamate REST;
- **utilizzo:** è utilizzato come buffer per gestire le richieste in uscita e le risposte ricevute;
- **attributi:**
 - baseUrl : string
 - * URL http verso cui devono essere inviate le richieste.
 - client : Object
 - * gestisce l'invio delle richieste e la ricezione delle risposte.
 - freeze : boolean
 - * viene utilizzata per bloccare il server in caso di collegamento mancante.
 - initialDelay : number
 - * rappresenta il tempo in millisecondi passato il quale una richiesta scaduta viene eseguita nuovamente.

- `-outRequests : number`
 - * rappresenta il numero delle richieste attualmente in attesa di risposta.
- `-requests : Request[]`
 - * lista delle richieste da evadere.
- `-requestsLimit : number`
 - * rappresenta il numero massimo di richieste eseguite ancora in attesa di risposta.
- `-timeLimitRequest : number`
 - * rappresenta il tempo in millisecondi dopo cui considerare la richiesta scaduta.

- **metodi:**

- `+add(request) : void`
 - aggiunge una richiesta alla coda del server
 - * `request : Request`
 - rappresenta la richiesta da aggiungere alla coda server.
- `+elaborateRequests() : void`
 - elabora la coda della richieste fino a quando ci sono elementi presenti
- `+internalCallback(request, response) : void`
 - esegue la chiamata alla funzione di callback fornita dalla richiesta nel caso in cui questa abbia ricevuto una risposta senza errori
 - * `request : Request`
 - rappresenta la richiesta che è stata inviata.
 - * `response : Object`
 - oggetto che rappresenta la risposta alla richiesta inviata.
- `+internalErrorCallback(request, response) : void`
 - esegue la chiamata alla funzione di callback fornita dalla richiesta nel caso in cui questa abbia ricevuto una risposta con errori
 - * `request : Request`
 - rappresenta la richiesta che è stata inviata.
 - * `response : Object`
 - oggetto che rappresenta la risposta alla richiesta inviata.
- `+internalTimeoutRequest(response) : void`
 - gestisce la richiesta nel caso in cui questa abbia non abbia ricevuto alcuna risposta entro il tempo di timeout
 - * `response : Object`
 - oggetto che rappresenta la risposta alla richiesta inviata.
- `+startElaboration() : void`
 - inizia l'esecuzione delle richieste in attesa di essere evase

5.8 DeGeOP::ActionPkg

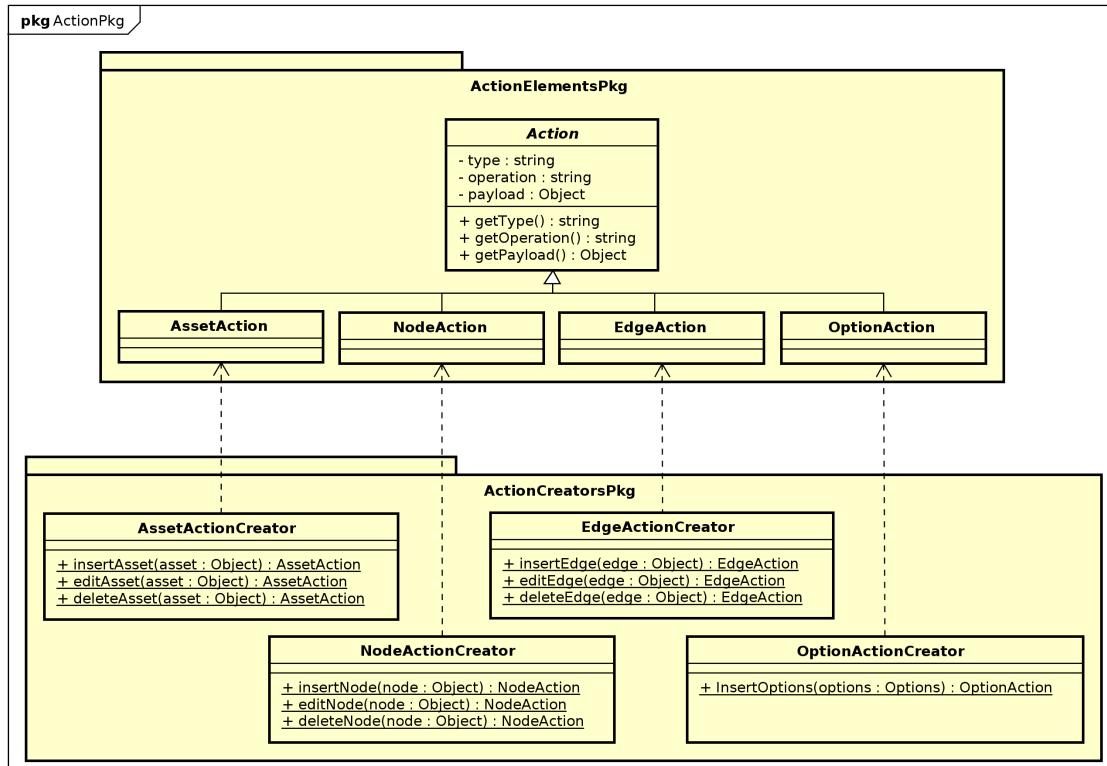


Figura 32: Schema componente DeGeOP::ActionPkg

5.8.1 Informazioni sul package

- **descrizione:** racchiude le componenti utilizzate per implementare le action dell'architettura Redux. Le action vengono create e ne viene fatto il dispatch verso lo store. Un reducer gestirà una action per produrre un cambiamento di stato sullo store;
- **padre:** [DeGeOP](#);
- **package contenuti:**
 - [ActionPkg::ActionCreatorsPkg](#);
 - [ActionPkg::ActionElementsPkg](#).
- **interazioni con altri package:**
 - IN CallManagerPkg: dispatch di azioni;
 - IN ReducerPkg: utilizzo di azioni ;
 - IN ViewPkg: dispatch di azioni.
- **classi contenute:**
 - [EdgeAction](#).

5.8.2 Classi

5.8.2.1 EdgeAction



Figura 33: Diagramma classe EdgeAction

- **descrizione:** rappresenta un'azione relativa agli archi;
- **utilizzo:** l'azione viene creata da un apposito ActionCreator per essere poi inviata ad un reducer.

5.9 DeGeOP::ActionPkg::ActionElementsPkg

5.9.1 Informazioni sul package

- **descrizione:** racchiude le componenti che rappresentano effettivamente le azioni;
- **padre:** [ActionPkg](#);
- **interazioni con altri package:**
 - IN ActionCreatorsPkg: creazione di azioni.
- **classi contenute:**
 - Action;
 - AssetAction;
 - NodeAction;
 - OptionAction.

5.9.2 Classi

5.9.2.1 Action

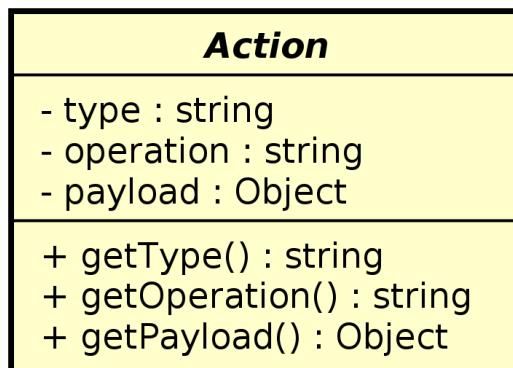


Figura 34: Diagramma classe Action

- **descrizione:** una classe astratta che rappresenta una generica azione di cui può essere fatto il dispatch;
- **utilizzo:** i suoi membri vengono usati dai reducer per completare una azione;
- **attributi:**
 - operation : string
 - * rappresenta l'operazione da eseguire.
 - payload : Object
 - * rappresenta l'oggetto che descrive il cambiamento apportato dall'azione.
 - type : string
 - * rappresenta la tipologia di elemento su cui eseguire l'azione.

- **metodi:**

- `+getOperation() : string`
il metodo ritorna l'operazione eseguita dall'azione
- `+getPayload() : Object`
il metodo ritorna il payload dell'oggetto
- `+getType() : string`
il metodo ritorna il tipo dell'azione

5.9.2.2 AssetAction



Figura 35: Diagramma classe AssetAction

- **descrizione:** rappresenta un'azione relativa agli asset;
- **utilizzo:** l'azione viene creata da un apposito ActionCreator per essere poi inviata ad un reducer.

5.9.2.3 NodeAction

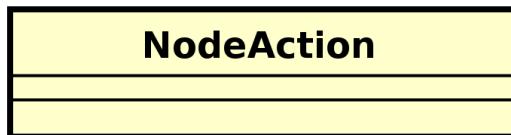


Figura 36: Diagramma classe NodeAction

- **descrizione:** rappresenta un'azione relativa ai nodi;
- **utilizzo:** l'azione viene creata da un apposito ActionCreator per essere poi inviata ad un reducer.

5.9.2.4 OptionAction



Figura 37: Diagramma classe OptionAction

- **descrizione:** rappresenta un'azione relativa all'oggetto option;
- **utilizzo:** l'azione viene creata da un apposito ActionCreator per essere poi inviata ad un reducer;
- **relazioni con altre classi:**
 - IN OptionActionCreator.

5.10 DeGeOP::ActionPkg::ActionCreatorsPkg

5.10.1 Informazioni sul package

- **descrizione:** racchiude le componenti che gestiscono la creazione delle azioni;
- **padre:** [ActionPkg](#);
- **interazioni con altri package:**
 - OUT ActionElementsPkg: creazione di azioni.
- **classi contenute:**
 - AssetActionCreator;
 - EdgeActionCreator;
 - NodeActionCreator;
 - OptionActionCreator.

5.10.2 Classi

5.10.2.1 AssetActionCreator

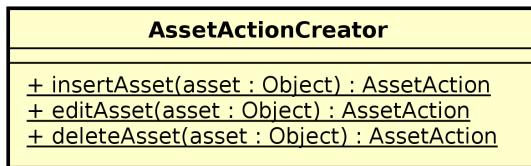


Figura 38: Diagramma classe AssetActionCreator

- **descrizione:** rappresenta la factory di azioni relative agli asset;
- **utilizzo:** i suoi metodi sono chiamati dalla View e dal CallManager per la creazione di azioni relative agli asset;
- **metodi:**
 - `+deleteAsset(asset) : AssetAction`
il metodo crea l'azione relativa all'eliminazione dell'asset ricevuto in input
* asset : Object
oggetto contenente i parametri di un Asset che dovrà essere eliminato.
 - `+editAsset(asset) : AssetAction`
il metodo crea l'azione relativa alla modifica di un asset
* asset : Object
oggetto contenente i parametri di un Asset che dovrà essere modificato nello store.
 - `+insertAsset(asset) : AssetAction`
il metodo crea l'azione relativa all'inserimento di un nuovo asset
* asset : Object
oggetto contenente i parametri di un Asset che dovrà essere inserito nello store.

5.10.2.2 EdgeActionCreator

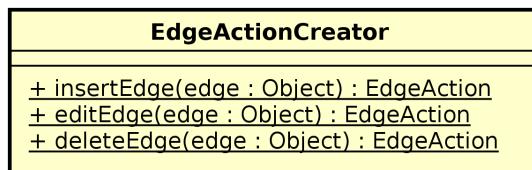


Figura 39: Diagramma classe EdgeActionCreator

- **descrizione:** rappresenta la factory di azioni relative agli archi;
- **utilizzo:** i suoi metodi sono chiamati dalla View e dal CallManager per la creazione di azioni relative agli archi;
- **metodi:**
 - `+deleteEdge(edge : Object)`
il metodo crea l'azione relativa all'eliminazione di un arco
 - * `edge : Object`
oggetto contenente i parametri di un arco che dovrà essere eliminato.
 - `+editEdge(edge : Object)`
il metodo crea l'azione relativa alla modifica di un arco
 - * `edge : Object`
oggetto contenente i parametri di un arco che dovrà essere modificato nello store.
 - `+insertEdge(edge : Object)`
il metodo crea l'azione relativa all'inserimento di un nuovo arco
 - * `edge : Object`
oggetto contenente i parametri di un arco che dovrà essere inserito nello store.

5.10.2.3 NodeActionCreator

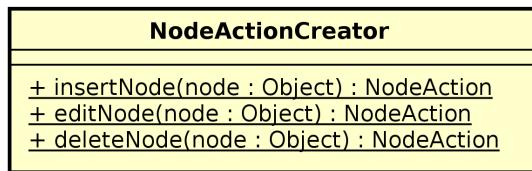


Figura 40: Diagramma classe NodeActionCreator

- **descrizione:** rappresenta la factory di azioni relative ai nodi;
- **utilizzo:** i suoi metodi sono chiamati dalla View e dal CallManager per la creazione di azioni relative ai nodi;
- **metodi:**

- `+deleteNode(node) : NodeAction`
il metodo crea l'azione relativa all'eliminazione di un nodo
 - * `node : Object`
oggetto contenente i parametri di un nodo che dovrà essere eliminato.
- `+editNode(node) : NodeAction`
il metodo crea l'azione relativa alla modifica di un nodo
 - * `node : Object`
oggetto contenente i parametri di un nodo che dovrà essere modificato nello store.
- `+insertNode(node) : NodeAction`
il metodo crea l'azione relativa all'inserimento di un nuovo nodo
 - * `node : Object`
oggetto contenente i parametri di un nodo che dovrà essere inserito nello store.

5.10.2.4 OptionActionCreator

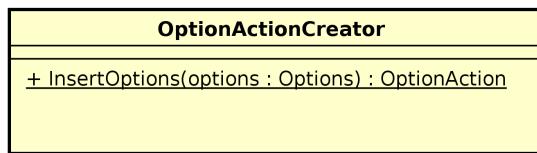


Figura 41: Diagramma classe OptionActionCreator

- **descrizione:** rappresenta la factory di azioni relative agli asset;
- **utilizzo:** i suoi metodi sono chiamati dalla View e dal CallManager per la creazione di azioni relative agli asset;
- **metodi:**
 - `+insertOptions() : OptionAction`
il metodo crea l'azione relativa all'inserimento dell'oggetto options nello store
- **relazioni con altre classi:**
 - OUT OptionAction.

5.11 DeGeOP::ViewPkg

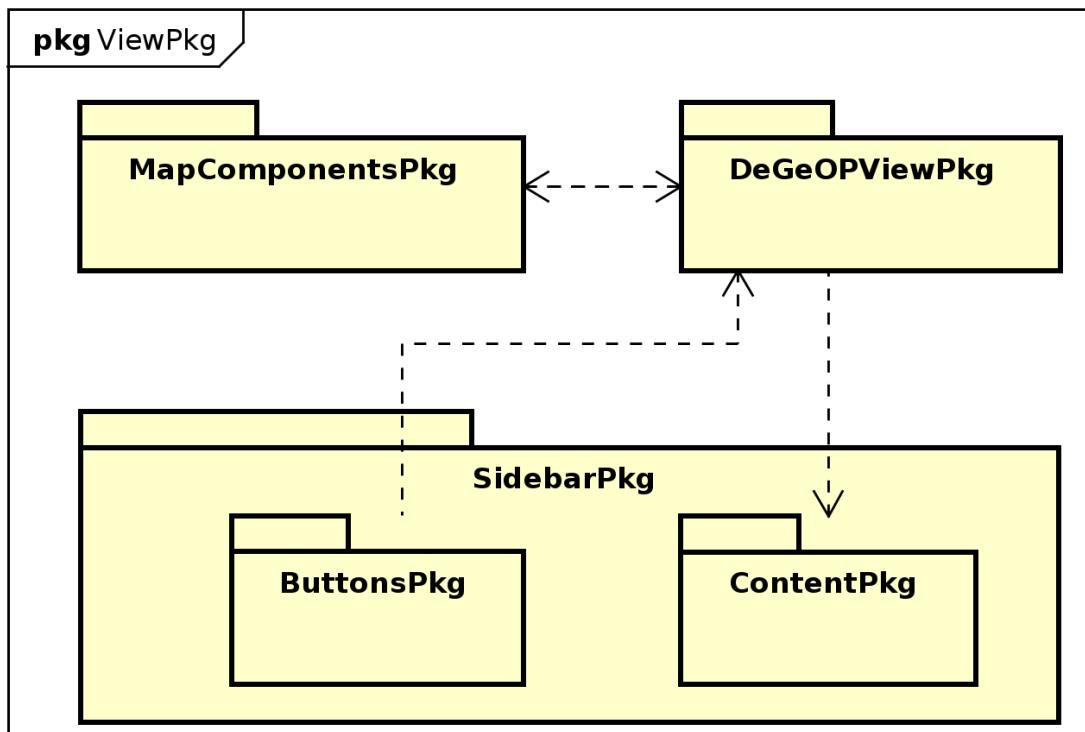


Figura 42: Schema componente DeGeOP::ViewPkg

5.11.1 Informazioni sul package

- **descrizione:** racchiude le componenti per la visualizzazione dell'interfaccia utente;
- **padre:** [DeGeOP](#);
- **package contenuti:**
 - [ViewPkg::DeGeOPViewPkg](#);
 - [ViewPkg::MapComponentsPkg](#);
 - [ViewPkg::SidebarPkg](#).
- **interazioni con altri package:**
 - OUT ActionPkg: dispatch di azioni;
 - OUT Alexa voice service: gestore vocale;
 - OUT Hammer: gestione gesture ;
 - OUT Openlayers: gestione mappa;
 - OUT React: utilizzo componenti react;
 - OUT ReactToolbox: utilizzo componenti material design;
 - OUT Redux: utilizzo metodo dispatch;
 - OUT StorePkg: subscribe sullo store.

5.12 DeGeOP::ViewPkg::DeGeOPViewPkg

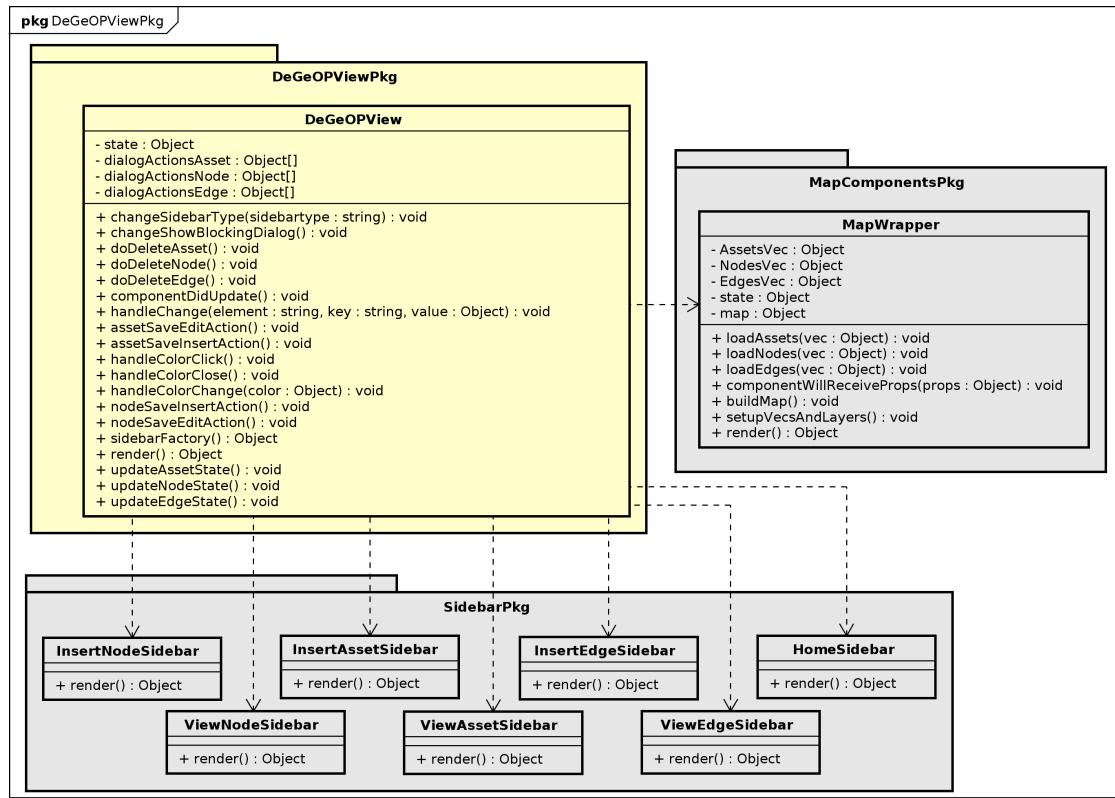


Figura 43: Schema componente DeGeOP::ViewPkg::DeGeOPViewPkg

5.12.1 Informazioni sul package

- **descrizione:** racchiude la componente principale della view;
- **padre:** [ViewPkg](#);
- **interazioni con altri package:**
 - IN MapComponentsPkg: utilizzo di componenti grafiche;
 - OUT MapComponentsPkg: utilizzo di componenti grafiche;
 - OUT SidebarPkg: utilizzo della sidebar.
- **classi contenute:**
 - DeGeOPView.

5.12.2 Classi

5.12.2.1 DeGeOPView

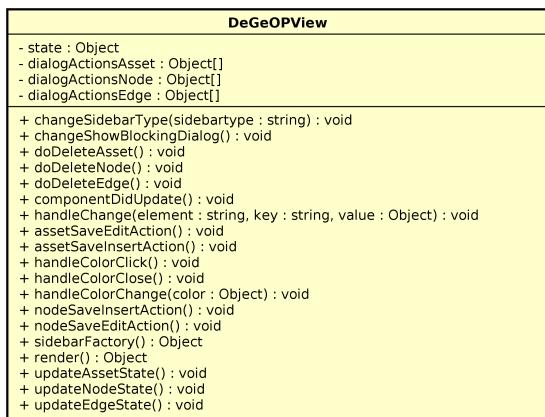


Figura 44: Diagramma classe DeGeOPView

- **descrizione:** rappresenta l'oggetto grafico radice, che comprende l'intera View del prodotto;
- **utilizzo:** i suoi metodi setter sono richiamati da ConcreteDeGeOPViewBuilder per impostare i suoi campi dati. La classe effettua il subscribe sullo Store per ricevere gli aggiornamenti ;
- **attributi:**
 - -dialogActionsAsset : Object[]
 - * contiene le funzioni richiamate durante la visualizzazione della schermata modale relativa agli asset.
 - -dialogActionsEdge : Object[]
 - * contiene le funzioni richiamate durante la visualizzazione della schermata modale relativa ai nodi.
 - -dialogActionsNode : Object[]
 - * contiene le funzioni richiamate durante la visualizzazione della schermata modale relativa ai nodi.
 - -state : Object
 - * rappresenta lo stato di DeGeOPView, ovvero i dati temporanei che l'utente inserisce prima che vengano inseriti nello store.
- **metodi:**
 - +assetSaveEditAction() : void
 - il metodo emette l'azione relativa alla modifica di un asset
 - +assetSaveInsertAction() : void
 - il metodo emette l'azione relativa all'inserimento di un asset
 - +changeShowBlockingDialog() : void
 - il metodo gestisce il cambiamento del booleano nello state di DeGeOPView che gestisce la visualizzazione della finestra modale

- `+changeSidebarType() : void`
il metodo gestisce il cambiamento della stringa dello state di DeGeOPView che gestisce il tipo di sidebar visualizzata al momento
- `+componentDidUpdate() : void`
il metodo viene richiamato quando lo state di DeGeOPView cambia; al suo interno vengono gestite le validazioni dei campi compilati dall'utente
- `+doDeleteAsset() : void`
il metodo emette l'azione di eliminazione dell'asset attualmente selezionato e ne fa il dispatch verso lo store
- `+doDeleteEdge() : void`
il metodo emette l'azione di eliminazione del nodo attualmente selezionato e ne fa il dispatch verso lo store
- `+doDeleteNode() : void`
il metodo emette l'azione di eliminazione del nodo attualmente selezionato e ne fa il dispatch verso lo store
- `+handleChange(element, key, value) : void`
il metodo gestisce il cambiamento di stato di DeGeOPView relativamente ai campi dati compilati dall'utente
 - * `element : string`
indica il campo dati dello stato di DeGeOPView da cambiare. Può assumere i valori: asset, node, edge, common.
 - * `key : string`
indica il campo dati dell'oggetto descritto da element che dovrà essere modificato.
 - * `value : Object`
indica il valore da inserire nell'oggetto descritto da element, nel campo dati descritto da key.
- `+handleColorChange(color) : void`
il metodo gestisce il cambiamento nello state di DeGeOPView relativo al colore selezionato dalla palette colori
 - * `color : Object`
rappresenta il nuovo colore selezionato dalla palette colori.
- `+handleColorClick() : void`
il metodo gestisce l'apertura e la chiusura della palette colori
- `+handleColorClose() : void`
il metodo gestisce la chiusura della palette colori
- `+nodeSaveEditAction() : void`
il metodo gestisce l'emissione dell'azione relativa alla modifica di un nodo
- `+nodeSaveInsertAction() : void`
il metodo gestisce l'emissione dell'azione relativa all'inserimento di un nodo
- `+render() : Object`
il metodo gestisce il render di DeGeOPView, composta da una sidebar e da un MapWrapper
- `+sidebarFactory() : Object`
il metodo gestisce la creazione di una sidebar da renderizzare in base alla stringa attualmente specificata nello state di DeGeOPView

- `+updateAssetState() : void`
il metodo imposta lo stato dell'asset in DeGeOPView con i dati dell'asset selezionato sulla mappa
- `+updateEdgeState() : void`
il metodo imposta lo stato dell'arco in DeGeOPView con i dati dell'arco selezionato sulla mappa
- `+updateNodeState() : void`
il metodo imposta lo stato del nodo in DeGeOPView con i dati del nodo selezionato sulla mappa

5.13 DeGeOP::ViewPkg::MapComponentsPkg

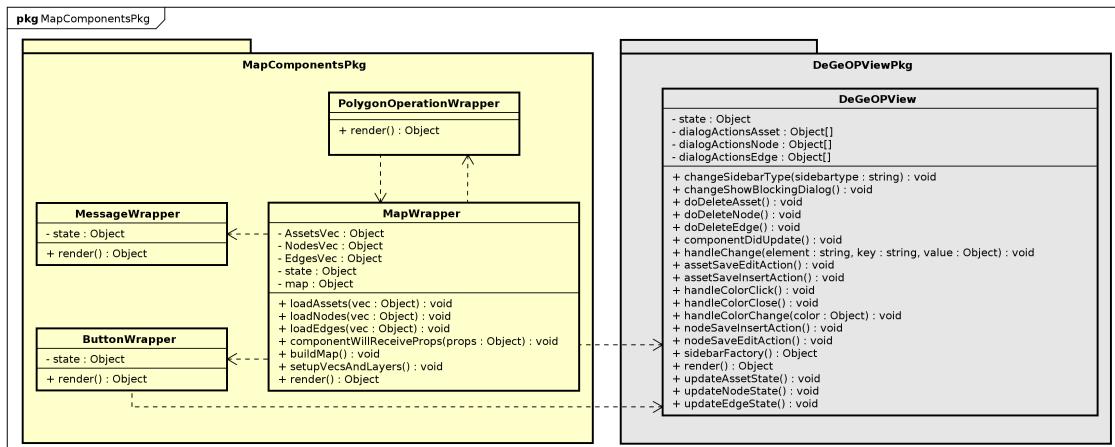


Figura 45: Schema componente DeGeOP::ViewPkg::MapComponentsPkg

5.13.1 Informazioni sul package

- descrizione:** racchiude le componenti relative alla mappa e ai pulsanti sopra di essa;
- padre:** [ViewPkg](#);
- interazioni con altri package:**
 - IN DeGeOPViewPkg: utilizzo di componenti grafiche;
 - OUT DeGeOPViewPkg: utilizzo di componenti grafiche;
 - OUT Openlayers: gestione mappa.
- classi contenute:**
 - ButtonWrapper;
 - MapWrapper;
 - MessageWrapper;
 - PolygonOperationWrapper.

5.13.2 Classi

5.13.2.1 ButtonWrapper

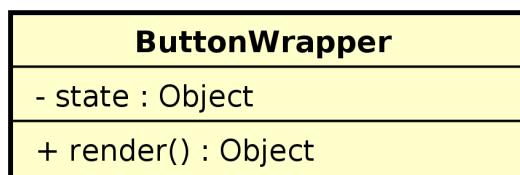


Figura 46: Diagramma classe ButtonWrapper

- descrizione:** rappresenta una classe wrapper per visualizzare una serie di pulsanti con cui è possibile eseguire varie operazioni;

- **utilizzo:** viene utilizzato per mostrare sulla mappa una serie di bottoni;
- **attributi:**
 - -state : Object
 - * rappresenta lo stato del ButtonWrapper.
- **relazioni con altre classi:**
 - IN MapWrapper.

5.13.2.2 MapWrapper

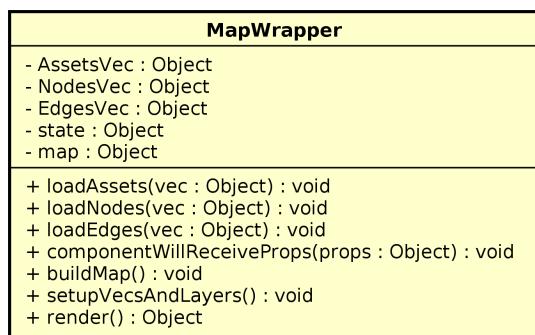


Figura 47: Diagramma classe MapWrapper

- **descrizione:** rappresenta una classe wrapper per visualizzare la mappa;
- **utilizzo:** viene utilizzata per visualizzare una mappa e permettere all'utente di interagire con essa;
- **attributi:**
 - -assetsVec : Object
 - * contiene gli oggetti geometrici relativi agli asset da disegnare sulla mappa.
 - -edgesVec : Object
 - * contiene gli oggetti geometrici relativi agli asset da disegnare sulla mappa.
 - -map : Object
 - * oggetto che contiene lo stato della mappa.
 - -nodesVec : Object
 - * contiene gli oggetti geometrici relativi ai nodi da disegnare sulla mappa.
 - -state : Object
 - * contiene lo stato della classe MapWrapper.
- **metodi:**
 - +buildMap() : void
 - il metodo costruisce la mappa

- `+componentWillReceiveProps(props) : void`
il metodo viene richiamato quando la componente MapWrapper sta per ricevere nuovi props
 - * `props : Object`
props da ricevere.
- `+loadAssets(vec) : void`
il metodo carica il vettore degli asset
 - * `vec : Object`
vettore degli asset da caricare.
- `+loadEdges(vec) : void`
il metodo carica il vettore degli archi
 - * `vec : Object`
vettore degli archi da caricare.
- `+loadNodes(vec) : void`
il metodo carica il vettore dei nodi
 - * `vec : Object`
vettore dei nodi da caricare.
- `+render() : Object`
renderizza il MapWrapper
- `+setupVecsAndLayers() : void`
il metodo prepara i vettori e i layers per la mappa

- **relazioni con altre classi:**

- IN PolygonOperationWrapper;
- OUT ButtonWrapper;
- OUT MessageWrapper;
- OUT PolygonOperationWrapper.

5.13.2.3 MessageWrapper

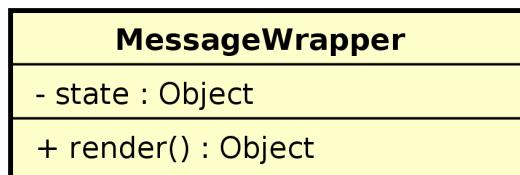


Figura 48: Diagramma classe MessageWrapper

- **descrizione:** rappresenta una classe wrapper per visualizzare un messaggio;
- **utilizzo:** viene utilizzato per mostrare messaggi di errore o di successo sulla mappa;
- **attributi:**
 - `-state : Object`

* rappresenta lo stato del MessageWrapper.

- **metodi:**

- +render() : Object
renderizza il MessageWrapper

- **relazioni con altre classi:**

- IN MapWrapper.

5.13.2.4 PolygonOperationWrapper

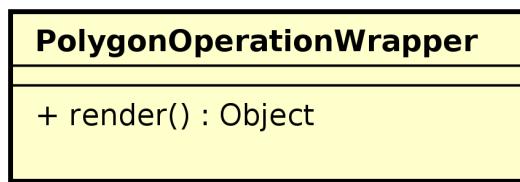


Figura 49: Diagramma classe PolygonOperationWrapper

- **descrizione:** rappresenta una classe wrapper per visualizzare un bottone con cui è possibile effettuare operazioni sul perimetro di un poligono sulla mappa;

- **utilizzo:** invocando i suoi metodi è possibile iniziare a disegnare il poligono su mappa oppure cancellare l'ultimo segmento disegnato;

- **metodi:**

- +render() : Object
renderizza il PolygonOperationWrapper

- **relazioni con altre classi:**

- IN MapWrapper;
 - OUT MapWrapper.

5.14 DeGeOP::ViewPkg::SidebarPkg

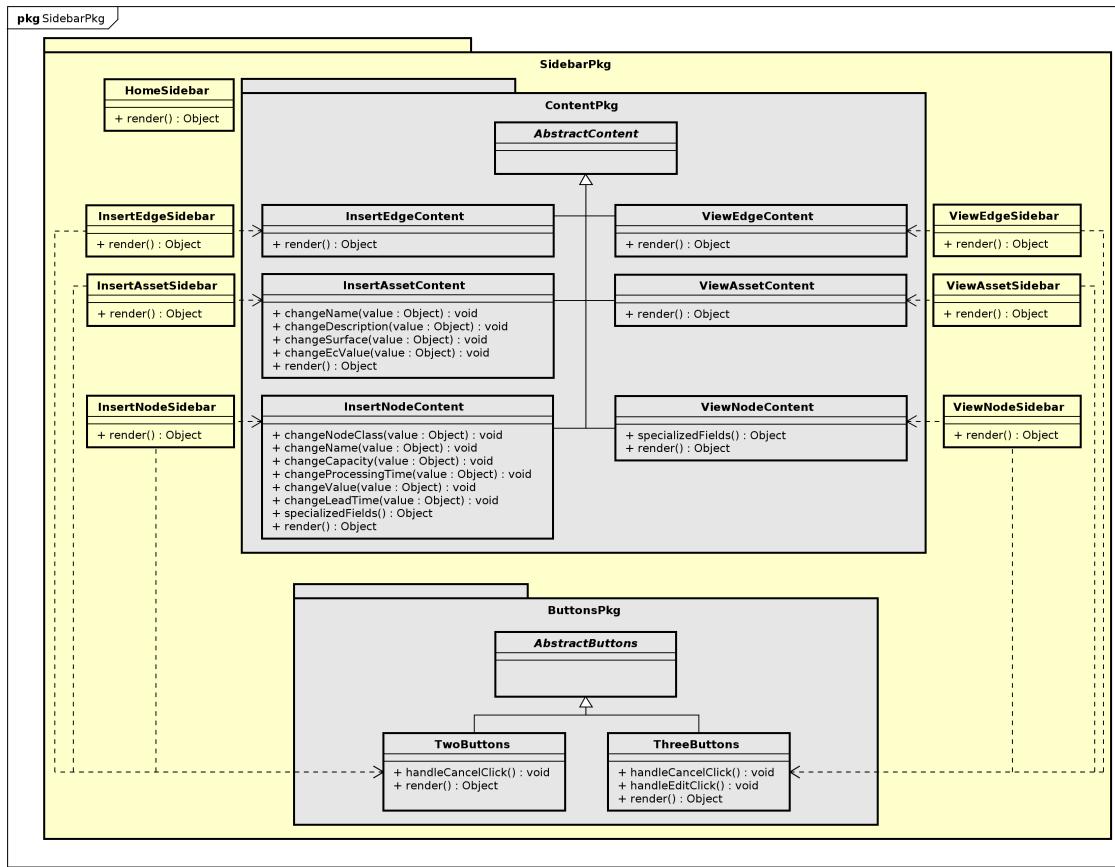


Figura 50: Schema componente DeGeOP::ViewPkg::SidebarPkg

5.14.1 Informazioni sul package

- **descrizione:** racchiude le componenti necessarie alla rappresentazione della sidebar;
- **padre:** [ViewPkg](#);
- **package contenuti:**
 - SidebarPkg::[ButtonsPkg](#);
 - SidebarPkg::[ContentPkg](#).
- **interazioni con altri package:**
 - IN DeGeOPViewPkg: utilizzo della sidebar.
- **classi contenute:**
 - HomeSidebar;
 - InsertAssetSidebar;
 - InsertEdgeSidebar;
 - InsertNodeSidebar;

- ViewAssetSidebar;
- ViewEdgeSidebar;
- ViewNodeSidebar.

5.14.2 Classi

5.14.2.1 HomeSidebar

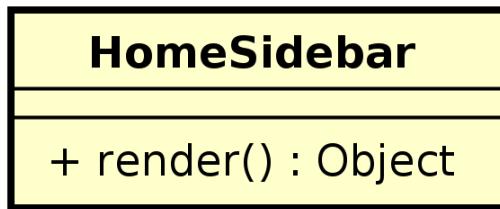


Figura 51: Diagramma classe HomeSidebar

- **descrizione:** rappresenta la sidebar di default ;
- **utilizzo:** renderizza una sidebar di default che dà il benvenuto all'utente;
- **metodi:**
 - +render() : Object
il metodo renderizza la sidebar di benvenuto

5.14.2.2 InsertAssetSidebar

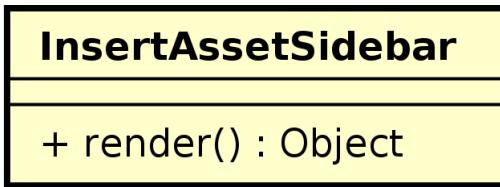


Figura 52: Diagramma classe InsertAssetSidebar

- **descrizione:** rappresenta una sidebar per l'inserimento e la modifica di un asset;
- **utilizzo:** renderizza una sidebar composta da contenuto in cui l'utente può compilare i dati e da buttoni che permettono di eseguire inserimenti e modifiche relative ad un asset;
- **metodi:**
 - +render() : Object
il metodo renderizza la sidebar di inserimento e modifica di un asset, composta da un InsertAssetContent e da un TwoButtons

5.14.2.3 InsertEdgeSidebar

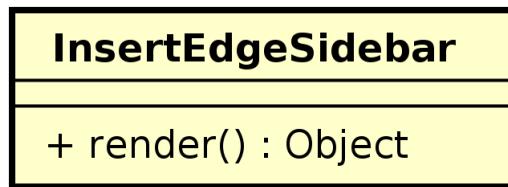


Figura 53: Diagramma classe InsertEdgeSidebar

- **descrizione:** rappresenta una sidebar per l'inserimento e la modifica di un arco;
- **utilizzo:** renderizza una sidebar composta da contenuto in cui l'utente può compilare i dati e da buttoni che permettono di eseguire inserimenti e modifiche relative ad un arco;
- **metodi:**
 - +render() : Object
il metodo renderizza la sidebar di inserimento e modifica di un arco, composta da un InsertEdgeContent e da un TwoButtons

5.14.2.4 InsertNodeSidebar

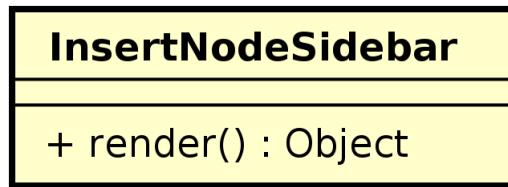


Figura 54: Diagramma classe InsertNodeSidebar

- **descrizione:** rappresenta una sidebar per l'inserimento e la modifica di un nodo;
- **utilizzo:** renderizza una sidebar composta da contenuto in cui l'utente può compilare i dati e da buttoni che permettono di eseguire inserimenti e modifiche relative ad un nodo;
- **metodi:**
 - +render() : Object
il metodo renderizza la sidebar di inserimento e modifica di un nodo, composta da un InsertNodeContent e da un TwoButtons

5.14.2.5 ViewAssetSidebar

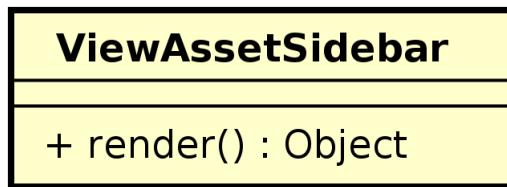


Figura 55: Diagramma classe ViewAssetSidebar

- **descrizione:** rappresenta una sidebar per la visualizzazione di un asset;
- **utilizzo:** renderizza una sidebar composta da contenuto in cui l'utente può visualizzare i dati e da buttoni che permettono di effettuare l'eliminazione dell'asset;
- **metodi:**
 - **+render() : Object**
il metodo renderizza una sidebar per la visualizzazione di un asset, composta da un ViewAssetContent e da un ThreeButtons

5.14.2.6 ViewEdgeSidebar

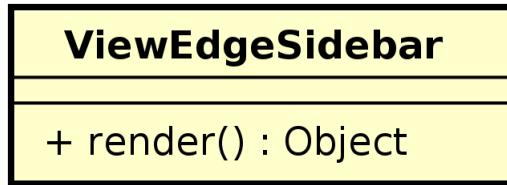


Figura 56: Diagramma classe ViewEdgeSidebar

- **descrizione:** rappresenta una sidebar per l'inserimento e la modifica di un arco;
- **utilizzo:** renderizza una sidebar composta da contenuto in cui l'utente può visualizzare i dati e da buttoni che permettono di effettuare l'eliminazione dell'arco;
- **metodi:**
 - **+render() : Object**
il metodo renderizza una sidebar per la visualizzazione di un arco, composta da un ViewEdgeContent e da un ThreeButtons

5.14.2.7 ViewNodeSidebar

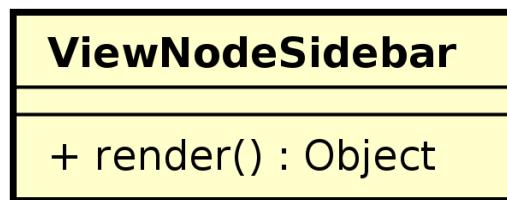


Figura 57: Diagramma classe ViewNodeSidebar

- **descrizione:** rappresenta una sidebar per la visualizzazione di un nodo;
- **utilizzo:** renderizza una sidebar composta da contenuto in cui l'utente può visualizzare i dati e da buttoni che permettono di effettuare l'eliminazione del nodo;
- **metodi:**
 - `+render() : Object`
il metodo renderizza una sidebar per la visualizzazione di un arco, composta da un ViewNodeContent e da un ThreeButtons

5.15 DeGeOP::ViewPkg::SidebarPkg::ContentPkg

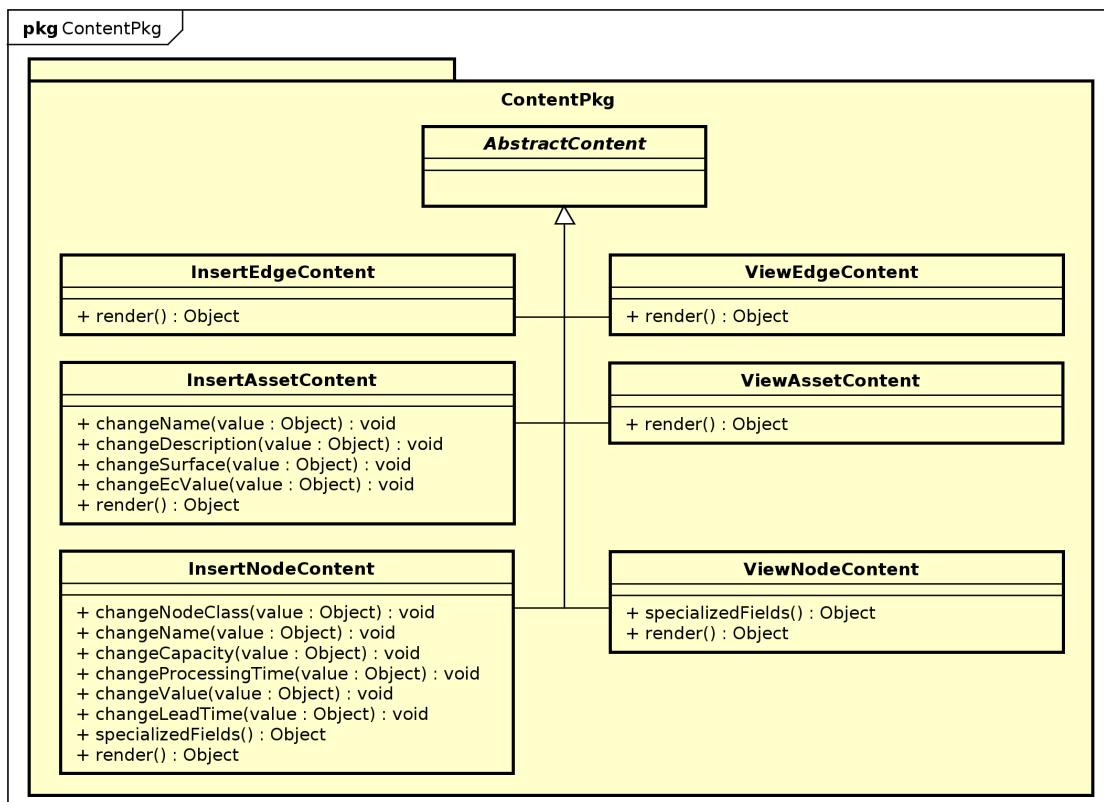


Figura 58: Schema componente DeGeOP::ViewPkg::SidebarPkg::ContentPkg

5.15.1 Informazioni sul package

- **descrizione:** racchiude le componenti che sono relative all'area informativa della Sidebar;
- **padre:** [SidebarPkg](#);
- **interazioni con altri package:**
 - IN FactorySidebarPkg: creazione contenuto della sidebar;
 - OUT React-color: visualizzazione palette colori.
- **classi contenute:**
 - `AbstractContent`;
 - `InsertAssetContent`;
 - `InsertEdgeContent`;
 - `InsertNodeContent`;
 - `ViewAssetContent`;
 - `ViewEdgeContent`;
 - `ViewNodeContent`.

5.15.2 Classi

5.15.2.1 AbstractContent



Figura 59: Diagramma classe AbstractContent

- **descrizione:** una classe d'interfaccia rappresentante il contenuto dell'area informativa nella sidebar;
- **utilizzo:** viene riferita da sidebar in quanto è una delle sue componenti.

5.15.2.2 InsertAssetContent

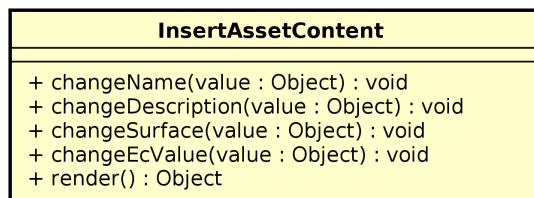


Figura 60: Diagramma classe InsertAssetContent

- **descrizione:** rappresenta il contenuto della sidebar relativa all'inserimento di un asset;
- **utilizzo:** viene creata da InsertAssetSidebarFactory ;
- **metodi:**
 - `+changeDescription() : void`
delega il cambiamento dell'input della descrizione alla componente di livello più alto
 - `+changeEcValue(value) : void`
delega il cambiamento dell'input del valore economico dell'asset alla componente di livello più alto
 - * `value : string`
valore attualmente contenuto nell'input del valore economico dell'asset.
 - `+changeName(value) : void`
delega il cambiamento dell'input del nome dell'asset alla componente di livello più alto
 - * `value : Object`
valore attualmente contenuto nell'input del nome dell'asset.
 - `+changeSurface() : void`
delega il cambiamento dell'input della superficie dell'asset alla componente di livello più alto

- `+render() : Object`
renderizza il contenuto della sidebar dell'asset in modalità inserimento e modifica

5.15.2.3 InsertEdgeContent

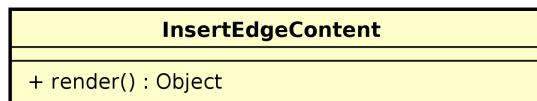


Figura 61: Diagramma classe InsertEdgeContent

- **descrizione:** rappresenta il contenuto della sidebar relativa all'inserimento di un arco;
- **utilizzo:** viene creata da `InsertEdgeSidebarFactory` ;
- **metodi:**
 - `+render() : Object`
renderizza il contenuto della sidebar dell'arco in modalità inserimento e modifica

5.15.2.4 InsertNodeContent

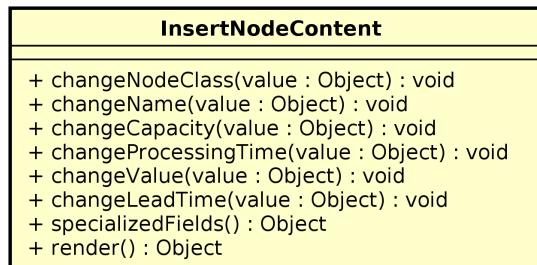


Figura 62: Diagramma classe InsertNodeContent

- **descrizione:** rappresenta il contenuto della sidebar relativa all'inserimento di un nodo;
- **utilizzo:** viene creata da `InsertNodeSidebarFactory` ;
- **metodi:**
 - `+changeCapacity() : void`
delega il cambiamento della capacità del nodo alla componente di più alto livello
 - `+changeLeadTime(value) : void`
delega il cambiamento del tempo di approvvigionamento del nodo alla componente di più alto livello
 - * `value : string`
valore che è contenuto nel dropdown della classe del nodo.
 - `+changeName(value) : void`
delega il cambiamento del nome del nodo alla componente di più alto livello

- * value : string
valore che è contenuto nel dropdown della classe del nodo.
- +changeNodeClass(value) : void
delega il cambiamento del dropdown del nome del nodo alla componente di più alto livello
 - * value : string
valore che è contenuto nel dropdown della classe del node.
- +changeProcessingTime(value) : void
delega il cambiamento del valore del nodo alla componente di più alto livello
 - * value : string
valore che è contenuto nel dropdown della classe del nodo.
- +render() : Object
renderizza il contenuto della sidebar del node in modalità inserimento e modifica
- +specializedFields() : Object
metodo che gestisce i campi dati da visualizzare a seconda della tipologia del nodo

5.15.2.5 ViewAssetContent



Figura 63: Diagramma classe ViewAssetContent

- **descrizione:** rappresenta il contenuto della sidebar relativa alla visualizzazione di un asset;
- **utilizzo:** viene creata da ViewAssetSidebarFactory ;
- **metodi:**
 - +render() : Object
renderizza la parte di sidebar relativa alla visualizzazione dei campi dati di un asset

5.15.2.6 ViewEdgeContent

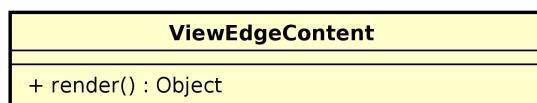


Figura 64: Diagramma classe ViewEdgeContent

- **descrizione:** rappresenta il contenuto della sidebar relativa alla visualizzazione di un arco;
- **utilizzo:** viene creata da ViewEdgeSidebarFactory ;
- **metodi:**

- `+render() : Object`
renderizza la parte di sidebar relativa alla visualizzazione dei campi dati di un arco
- `+specializedFields() : Object`
si occupa della visualizzazione dei campi dati a seconda della tipologia del nodo

5.15.2.7 ViewNodeContent

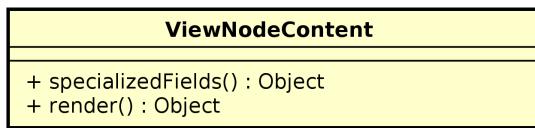


Figura 65: Diagramma classe ViewNodeContent

- **descrizione:** rappresenta il contenuto della sidebar relativa alla visualizzazione di un nodo;
- **utilizzo:** viene creata da `ViewNodeSidebarFactory` ;
- **metodi:**
 - `+render() : Object`
renderizza la parte di sidebar relativa alla visualizzazione dei campi dati di un nodo

5.16 DeGeOP::ViewPkg::SidebarPkg::ButtonsPkg

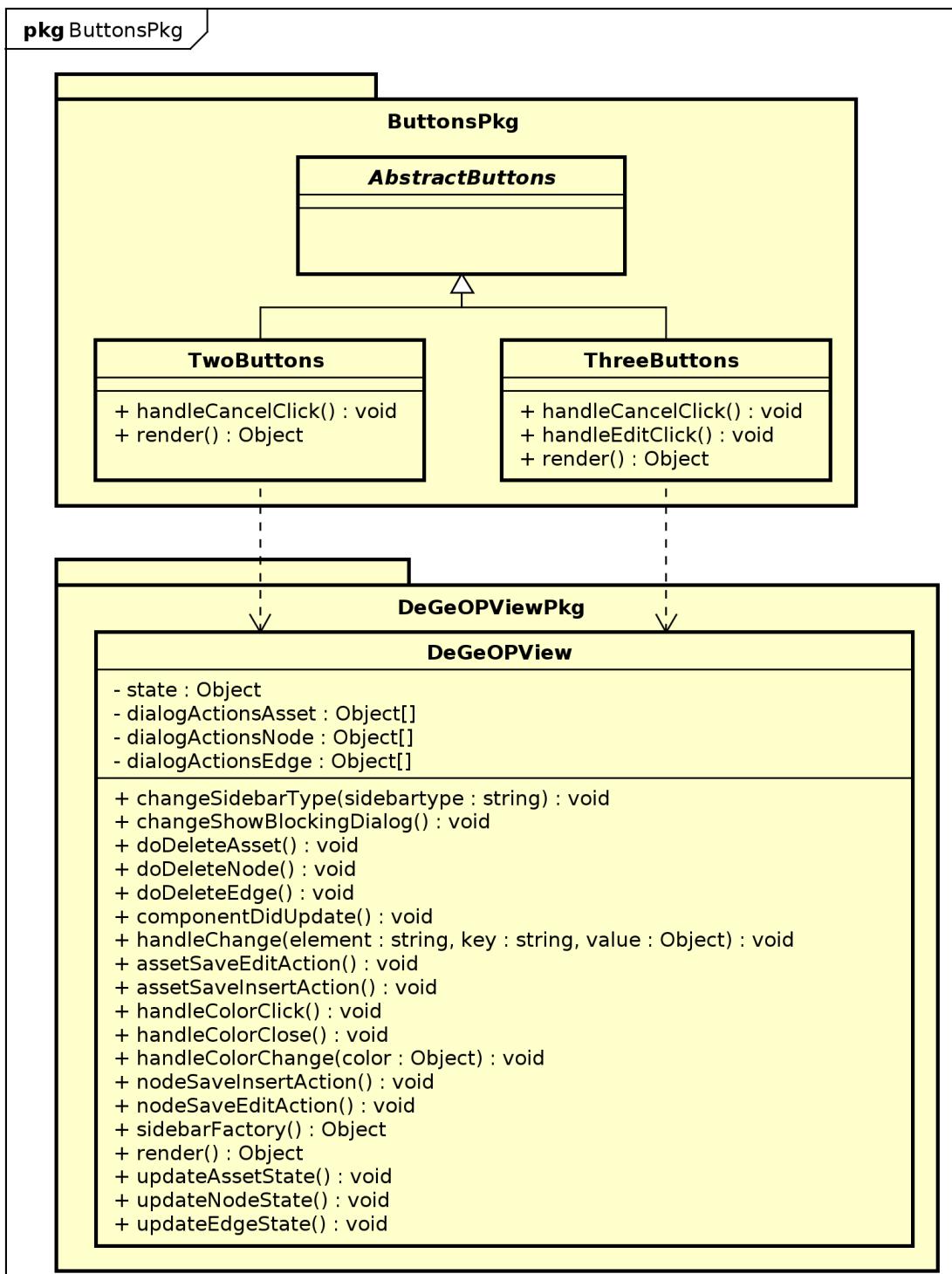


Figura 66: Schema componente DeGeOP::ViewPkg::SidebarPkg::ButtonsPkg

5.16.1 Informazioni sul package

- **descrizione:** racchiude le componenti che sono relative all'area con i bottoni della Sidebar;
- **padre:** [SidebarPkg](#);
- **classi contenute:**
 - AbstractButtons;
 - ThreeButtons;
 - TwoButtons.

5.16.2 Classi

5.16.2.1 AbstractButtons

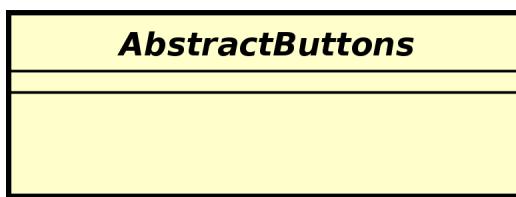


Figura 67: Diagramma classe AbstractButtons

- **descrizione:** una classe d'interfaccia rappresentante i bottoni inseriti nella sidebar;
- **utilizzo:** viene riferita da sidebar in quanto è una delle sue componenti.

5.16.2.2 ThreeButtons

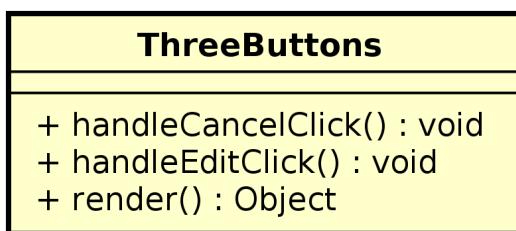


Figura 68: Diagramma classe ThreeButtons

- **descrizione:** classe che renderizza tre bottoni, uno di modifica, uno di eliminazione e uno di annullamento. Il bottone di annullamento provoca l'annullamento della selezione corrente. Il bottone di eliminazione provoca la comparsa di una finestra modale che chiede la conferma dell'eliminazione dell'elemento correntemente selezionato. Il bottone di modifica provoca l'avvio della modalità di modifica, con cui è possibile ridisegnare l'asset o cambiare i campi dati;
- **utilizzo:** viene utilizzata nelle sidebar di visualizzazione;

- **metodi:**

- `+handleCancelClick() : void`
il metodo gestisce il click sul bottone di annullamento, ripristinando la sidebar di default
- `+handleEditClick() : void`
il metodo imposta la sidebar di modifica dell'elemento attualmente selezionato
- `+render() : Object`
renderizza tre bottoni, uno di modifica, uno di annullamento e uno di eliminazione

5.16.2.3 TwoButtons

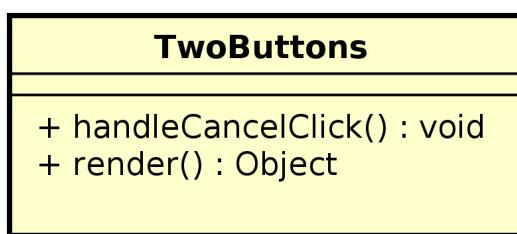


Figura 69: Diagramma classe TwoButtons

- **descrizione:** classe che renderizza due bottoni, uno di salvataggio e uno di annullamento. Il bottone di salvataggio è disabilitato fino a che tutti i campi della sidebar non sono stati compilati in modo corretto. Il bottone di annullamento provoca l'interruzione dell'inserimento dei dati;
- **utilizzo:** viene utilizzata nelle sidebar di inserimento;
- **metodi:**
 - `+handleCancelClick() : void`
il metodo gestisce il click sul bottone di annullamento, ripristinando la sidebar di default
 - `+render() : Object`
renderizza due bottoni, uno di annullamento e uno di salvataggio

6 Estensione delle funzionalità

Per ragioni di tempo e competenze alcune funzionalità non sono state implementate. Inoltre lo sviluppo è stato guidato dall'architettura preesistente di *RiskApp* e dalle API disponibili al momento della presa in consegna del progetto *DeGeOP*.

6.1 Analisi di rischio

La funzionalità di avvio ed eliminazione delle analisi di rischio sono state implementate utilizzando una componente che simula il server di Riskapp, dato che le API per le analisi non sono state fornite. Per implementare questa funzionalità è necessario:

- ottenere le API da *RiskApp* per interfacciarsi con il loro server;
- inserire in *StorePkg* le classi necessarie all'implementazione della parte di Store relativa alle analisi di rischio, come descritto dal documento di *Specifica Tecnica*;
- inserire in *ActionsPkg* le classi necessarie all'implementazione delle actions che operano sulle analisi di rischio, come descritto dal documento di *Specifica Tecnica*;
- inserire in *ReducerPkg* le funzioni necessarie all'implementazione dei reducer che operano sulle analisi di rischio, come descritto dal documento di *Specifica Tecnica*;
- inserire in *ViewPkg* le componenti React necessarie all'avvio ed all'eliminazione delle analisi di rischio, come descritto dal documento di *Specifica Tecnica*.
- inserire in *CallManagerPkg* le classi necessarie all'interfacciamento con il server *RiskApp*.

6.2 Nuova tipologia di nodo

I nodi possono essere di diverse tipologie. In caso si volesse inserire una nuova tipologia di nodo, le operazioni da svolgere per implementarli sono:

- ottenere le API da *RiskApp* per interfacciarsi con il loro server;
- inserire in *StorePkg* la classe necessaria all'implementazione della parte di Store relativa a quel nodo. Questa classe deve estendere la classe *Node*;
- inserire in *ActionsPkg* le classi necessarie all'implementazione delle actions che operano su quel nuovo tipo di nodo;
- inserire in *ReducerPkg* le funzioni necessarie all'implementazione dei reducer che operano su quel nuovo tipo di nodo;
- inserire in *ViewPkg* le componenti React necessarie all'avvio ed all'eliminazione delle analisi di rischio. In particolare nelle classi *ViewPkg::SidebarPkg::ContentPkg::InsertNodeContent* e *ViewPkg::SidebarPkg::ContentPkg::ViewNodeContent*, nella funzione *specializedField()* che si occupa della renderizzazione dei campi dati specifici, aggiungere la sezione che renderizza i campi dati della nuova tipologia di nodo;
- inserire in *CallManagerPkg* le classi necessarie all'interfacciamento con il server *RiskApp*.

7 Estensione del codice

7.1 Creazione di una nuova categoria di sidebar

Le sidebar presentano, come richiesto da React una struttura altamente modulare. Per aggiungere una nuova sidebar di seguito denominata per comodità *NewSidebar* si devono effettuare le seguenti operazioni:

- aggiungere `ViewPkg::DeGeOPView::DeGeOPView`, nella funzione `sidebarFactory()` il *conditional render* della nuova sidebar;
- aggiungere in `ViewPkg::DeGeOPView` la componente React *NewSidebar* che renderizza la sidebar. Essendo ogni sidebar divisa in due sezioni (contenuto e bottoni), per renderizzarla dovranno venire a sua volta richiamate le renderizzazioni:
 - della nuova componente React (da creare anch'essa) `ViewPkg::DeGeOPView::ContentPkg::NewSidebarContent`;
 - di una delle componenti a scelta tra `ViewPkg::DeGeOPView::ButtonsPkg::threeButtons`, `ViewPkg::DeGeOPView::ButtonsPkg::twoButtons`.

7.2 Aggiornamenti dei campi dati di asset e nodi

Visto che *RiskApp* utilizza il modello di sviluppo Agile, il loro server e i file JSON che esso ritorna, sono soggetti a continui e rapidi cambiamenti. Di seguito viene descritto come estendere il codice in caso venissero aggiunti nuovi campi dati per asset o nodi.

7.2.1 Asset

In caso venissero aggiunti nuovi campi dati per un asset seguire i seguenti passi:

- aggiungere in `StorePkg::ProcessPkg::Asset` il campo dati e le funzioni per gestire le sue eventuali validazioni;
- aggiungere nello `state` `ViewPkg::DeGeOPView::DeGeOPView` il campo dati e le sue validazioni e in `ViewPkg::SidebarPkg::ContentPkg::InsertAssetContent` e `ViewPkg::SidebarPkg::ContentPkg::ViewAssetContent` aggiungere il campo dati di interesse utilizzando le componenti di React-Toolbox, come descritto nella *Specifica Tecnica*.

7.2.2 Nodo

In caso venissero aggiunti nuovi campi dati per un nodo seguire i seguenti passi:

- aggiungere in `StorePkg::ProcessPkg::Node` nelle sue derivate il campo dati e le funzioni per gestire le sue eventuali validazioni;
- aggiungere nello `state` `ViewPkg::DeGeOPView::DeGeOPView` il campo dati e le sue validazioni e in `ViewPkg::SidebarPkg::ContentPkg::InsertNodeContent` e `ViewPkg::SidebarPkg::ContentPkg::ViewNodeContent` aggiungere il campo dati di interesse utilizzando le componenti di React-Toolbox, come descritto nella *Specifica Tecnica*.

A Glossario

A.1 A

Action

In Redux, oggetto che descrive un cambiamento di stato.

API

Application Programming Interface. Insieme di procedure utilizzabili per interfacciarsi con un programma o un sistema informatico in modo standard. Spesso si intendono le librerie software disponibili in un certo linguaggio di programmazione.

Arco

Collegamento orientato tra due nodi.

Asset

Fabbricato con importanza strategica per il processo produttivo di un'azienda. Un asset può contenere uno o più nodi.

Asus

Azienda produttrice di dispositivi tecnologici di varia tipologia.

A.2 B

Browser

Il web browser, o più semplicemente browser, è un'applicazione per il recupero, la presentazione e la navigazione di risorse web. Tali risorse (ad esempio pagine web, immagini, video) sono a disposizione sul World Wide Web su una rete locale o sullo stesso computer dove il browser è in esecuzione.

A.3 C

Chrome

Browser web sviluppato da Google.

Componente

1. React: elemento che fa parte della gerarchia del DOM. L'elemento eredita dalla classe base astratta React.Component.
2. Unità software dotata di una precisa identità e interfacce ben definite.

Cross-platform

Possibilità di poter usare lo stesso strumento software su diversi sistemi operativi.

CSS

Cascading Style Sheets. Linguaggio che permette di definire lo stile e la formattazione di una pagina [HTML_G](#). Permette di mantenere separate presentazione e contenuto.
La versione stabile più recente è CSS3.

A.4 D

Design pattern

Soluzione progettuale generale per la risoluzione di un problema ricorrente.
I design pattern orientati agli oggetti tipicamente mostrano relazioni ed interazioni tra classi o oggetti. Ad un livello più alto si trovano invece i pattern architetturali, con ambito più ampio.
Essi descrivono un pattern complessivo adottato dall'intero sistema.

Dispatch

In Redux, metodo per inviare un'azione allo store.

DOM

Document Object Model. Forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti, definita dal [W3C_G](#).

A.5 E

ESLint

Tool per l'analisi statica del codice [JavaScript_G](#).

A.6 F

Firefox

[Browser_G](#) web libero e multipiattaforma, mantenuto da Mozilla Foundation.

Framework

Insieme di classi cooperanti che forniscono lo scheletro di un'applicazione riusabile per uno specifico dominio applicativo. Delinea l'architettura delle applicazioni in cui viene usato.

A.7 G

Gesture

Combinazione di movimenti e click del dispositivo di puntamento (ad esempio il mouse) che vengono riconosciuti come comandi specifici.

Git

Sistema di controllo di versione_G distribuito e [open source_G](#), creato da Linus Torvalds nel 2005.
Vari progetti software usano Git per il controllo del versionamento, principalmente il kernel [Linux_G](#).

GitHub

Servizio Web per il controllo di versione basato su [Git_G](#). Offre diversi piani per [repository_G](#) privati sia a pagamento che gratuiti, molto utilizzati per lo sviluppo di progetti [open source_G](#).

Gruppo

Componenti che fanno parte del gruppo *Zephyrus*.

A.8 H**HTML**

HyperText Markup Language. Linguaggio usato per la definizione di pagine Web; la sua sintassi è stabilita dal [W3C_G](#). HTML5 è l'ultima versione stabile.

A.9 I**IDE**

Integrated Development Environment. Software utilizzato per la scrittura di codice sorgente. Spesso aiuta il programmatore segnalando errori di sintassi, oltre a tutta una serie di strumenti e funzionalità di supporto allo sviluppo.

iOS

Sistema operativo sviluppato da Apple Inc.

Ipad

Tablet prodotto dall'azienda Apple Inc.

A.10 J**JavaScript**

Linguaggio di scripting orientato agli oggetti e agli eventi. È utilizzato prevalentemente nella programmazione Web lato client per la creazione di effetti dinamici interattivi.

A.11 L**Libreria**

Insieme di funzioni e strutture dati predefinite e predisposte per lo sviluppo di software.

Linguaggio di markup

Insieme di regole che descrivono i meccanismi di rappresentazione (strutturali, semantici o presentazionali) di un testo che, utilizzando convenzioni standardizzate, sono utilizzabili su più supporti.

Linux

Linux è una famiglia di sistemi operativi di tipo Unix-like, rilasciati sotto varie distribuzioni, aventi la caratteristica comune di utilizzare come nucleo il kernel Linux. Ubuntu è la distribuzione di Linux più utilizzata.

A.12 M**MacOS**

Sistema operativo sviluppato da Apple.

A.13 N**Node.js**

Runtime *JavaScript_G* costruito sul motore V8 di Chrome.

Nodo

Oggetto che fa parte del processo produttivo aziendale. È contenuto all'interno di un asset.

A.14 O**Open source**

Software di cui i detentori dei diritti rendono pubblico il codice sorgente, permettendo ad altri programmatore di apportarvi modifiche. Questa possibilità è regolata tramite l'applicazione di apposite licenze d'uso.

OpenStreetMap

Servizio di mappe liberamente modificabili dell'intero pianeta.

A.15 P**Package**

Costrutto per organizzare classi logicamente correlate o che forniscono servizi simili, all'interno di sottogruppi ordinati. I package possono essere compressi permettendo la trasmissione di più classi in una sola volta. In *UML_G*, analogamente, è un raggruppamento arbitrario di elementi in una unità di livello più alto.

Proprietà

Input con i quali sono costruite le componenti React.

A.16 R**React**

Libreria *JavaScript_G* per la creazione di interfacce grafiche.

Reducer

In Redux, funzione che restituisce un nuovo stato dello store in seguito ad un'azione.

Render

Metodo richiesto da React.Component per la visualizzazione delle componenti sul DOM virtuale.

Repository

Ambiente di un sistema informativo in cui vengono gestiti i metadati attraverso tabelle relazionali. Il repository utilizzato dal *gruppo_G Zephyrus* è fornito dalla piattaforma *GitHub_G*.

REST

Stile architettonico che offre la possibilità di manipolare rappresentazioni testuali di risorse Web utilizzando un set predefinito di operazioni.

A.17 S**Safari**

browser_G web sviluppato dalla Apple.

Sistema di controllo di versione

Strumento che consente di tracciare le modifiche a cui viene sottoposto un insieme di file, consentendo di accedere alle vecchie versioni e di lavorare in più persone contemporaneamente.

Store

In Redux, il contenitore degli stati dell'applicazione.

A.18 V**VirtualBox**

Software open source per l'esecuzione di macchine virtuali.

A.19 W**W3C**

World Wide Web Consortium. Organizzazione non governativa internazionale che ha come scopo quello di sviluppare tutte le potenzialità del World Wide Web. Al fine di riuscire nel proprio intento, la principale attività svolta dal W3C consiste nello stabilire standard tecnici che riguardino sia i linguaggi di markup che i protocolli di comunicazione.

WebStorm

IDE_G che offre supporto allo sviluppo di applicazioni *JavaScript_G* sia client che server. Offre inoltre supporto a *NodeJS_G*, *HTML_G*, *CSS_G* e frameworks come AngularJS e a librerie JavaScript come React.

Windows

Sistema operativo sviluppato da Microsoft.