



Specifica Tecnica

Gruppo MINT – Progetto MaaS

Informazioni sul documento

Versione	1.0.0
Redazione	Navid Taha, Tommaso Zagni
Verifica	Enrico Canova, Fabiano Tavallini
Approvazione	Michael Ogbuachi
Uso	Interno
Distribuzione	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo MINT

Descrizione

Questo documento descrive la specifica tecnica e l'architettura utilizzata nella realizzazione del progetto MaaS.

Registro delle modifiche

Versione	Data	Collaboratori	Descrizione
1.0.0	16-05-2016	Navid Taha	Approvazione del documento.
0.1.1	15-05-2016	Michael Ogbuachi	Verifica del documento.
0.1.0	14-05-2016	Thomas Fuser	Completato Front-End.
0.0.9	13-05-2016	Enrico Canova	Completato Back-End.
0.0.8	07-05-2016	Michael Ogbuachi e Thomas Fuser	Appendici Pattern.
0.0.7	06-05-2016	Tommaso Zagni	Descrizione View.
0.0.6	05-05-2016	Tommaso Zagni	Inizio Front-End.
0.0.5	05-05-2016	Fabiano Tavallini	Inizio Back-End.
0.0.4	30-04-2016	Tommaso Zagni	Inizio Descrizione architetturale.
0.0.3	29-04-2016	Thomas Fuser ed Enrico Canova	Aggiunta sezione tecnologie utilizzate.
0.0.2	27-04-2016	Thomas Fuser ed Enrico Canova	Inizio stesura Introduzione.
0.0.1	25-04-2016	Navid Taha	Inizio stesura documento.

Indice

1 Introduzione	8
1.1 Scopo del documento	8
1.2 Scopo del prodotto	8
1.3 Glossario	8
1.4 Riferimenti	8
1.4.1 Normativi	8
1.4.2 Informativi	8
2 Tecnologie utilizzate	9
2.1 Node.js	9
2.1.1 Vantaggi	9
2.2 MongoDB	9
2.2.1 Vantaggi	9
2.3 React	10
2.3.1 Vantaggi	10
2.4 LoopBack	10
2.4.1 Vantaggi	10
2.5 HTML5	11
2.5.1 Vantaggi	11
2.6 CSS	11
2.6.1 Vantaggi	11
2.7 Sweet.js	11
2.7.1 Vantaggi	11
3 Descrizione architetturale	12
3.1 Metodo e formalismo di specifica	12
3.2 Architettura generale	12
3.3 Interfaccia REST	13
4 Back-end	16
4.1 Descrizione packages e classi	16
4.1.1 Back-end	16
4.1.1.1 Informazioni sul package	18
4.1.1.2 Classi	18
4.1.2 Back-end::Models	19
4.1.2.1 Informazioni sul package	20
4.1.2.2 Classi	20
4.1.3 Back-end::Models::DSLISModels	23
4.1.3.1 Informazioni sul package	24
4.1.3.2 Classi	24
4.1.4 Back-end::Datasources	33
4.1.4.1 Informazioni sul package	33
4.1.4.2 Classi	33
4.1.5 Back-end::Connectors	35
4.1.5.1 Informazioni sul package	35
4.1.5.2 Classi	35

4.1.6	Back-end::RESTAPIs	37
4.1.6.1	Informazioni sul package	37
4.1.6.2	Classi	38
4.1.7	Back-end::RESTAPI::DSLISRESTAPIs	41
4.1.7.1	Informazioni sul package	41
4.1.7.2	Classi	41
4.1.8	Back-end::Middlewares	44
4.1.8.1	Informazioni sul package	44
4.1.8.2	Classi	44
4.2	Interfaccia REST	46
4.2.1	Comunicazione tra client e server	46
4.2.2	Permessi	46
4.2.3	Richieste HTTP	46
4.2.3.1	Richieste Users	46
4.2.3.2	Richieste Super Admin	55
4.2.3.3	Richieste Companies	57
4.2.3.4	Richieste Databases	62
4.2.3.5	Richieste Collections	65
4.2.3.6	Richieste Documents	72
4.2.3.7	Richieste Dashboards	80
4.2.3.8	Richieste Cell	86
5	Front-end	94
5.1	Descrizione packages e classi	94
5.1.1	Front-end	94
5.1.1.1	Informazioni sul package	94
5.1.1.2	Classi	94
5.1.2	Front-end::ActionCreators	96
5.1.2.1	Informazioni sul package	96
5.1.2.2	Classi	96
5.1.3	Front-end::ActionCreators::DSLISActionCreators	100
5.1.3.1	Informazioni sul package	100
5.1.3.2	Classi	100
5.1.4	Front-end::WebAPIUtils	103
5.1.4.1	Informazioni sul package	103
5.1.4.2	Classi	103
5.1.5	Front-end::WebAPIUtils::DSLISWebAPIUtils	105
5.1.5.1	Informazioni sul package	105
5.1.5.2	Classi	106
5.1.6	Front-end::Stores	107
5.1.6.1	Informazioni sul package	107
5.1.6.2	Classi	108
5.1.7	Front-end::Stores::DSLISStores	110
5.1.7.1	Informazioni sul package	110
5.1.7.2	Classi	110
5.1.8	Front-end::Views	113
5.1.8.1	Informazioni sul package	114
5.1.8.2	Classi	115

6 Diagrammi di Attività	125
6.1 Sistema MaaS - Attività principali	125
6.2 Utente generico	127
6.2.1 Registrazione nuovo proprietario	128
6.2.2 Autenticazione	129
6.2.3 Recupero password	130
6.2.4 Reset password	131
6.2.5 LogOut	132
6.2.6 Modifica impostazioni personali del sistema	133
6.2.7 Modifica profilo	134
6.2.8 Modifica password da profilo	135
6.2.9 Modifica Dashboard principale	136
6.2.10 Creazione DSLIS	137
6.2.11 Modifica DSLIS	138
6.2.12 Esecuzione DSLIS	138
6.2.13 Eliminazione DSLIS	139
6.3 Gestione Utenti	140
6.3.1 Invito di un nuovo utente	140
6.3.2 Modifica del ruolo di un utente	141
6.3.3 Eliminazione di un utente	142
6.3.4 Modifica dei permessi per un DSLIS	143
6.3.5 Modifica dei permessi di accesso al database	144
6.4 Gestione Aziende	144
6.4.1 Connessione del database	145
6.4.2 Impersonificazione di un utente	146
Appendici	148
A Descrizione Design Pattern	148
A.1 Design Pattern Architetturali	148
A.1.1 Flux	148
A.2 Design Pattern Creazionali	149
A.2.1 Singleton	149
A.2.2 Factory method	149
A.2.3 Abstract Factory	150
A.3 Design Pattern Strutturali	151
A.3.1 Facade	151
A.4 Design Pattern Comportamentali	152
A.4.1 Chain of Responsibility	152
A.4.2 Template Method	153

Elenco delle figure

1 Diagramma del Deployment	13
2 Diagramma delle classi del back-end	16
3 Diagramma dei package del back-end	17
4 Diagramma delle classi del package Model	19
5 Diagramma delle figure del package: Back-end::Models::DSLISModels	23

6	Diagramma delle classi del datasource	33
7	Diagramma delle classi dei connectors	35
8	Diagramma delle classi dei RESTAPIs	37
9	Diagramma delle classi del DLSRESTAPI	41
10	Diagramma delle classi dei Middlewares	44
11	Diagramma di sequenza: POST /User	47
12	Diagramma di sequenza: POST /Users/login	48
13	Diagramma di sequenza: PUT /Users	48
14	Diagramma di sequenza: POST /Users/logout	49
15	Diagramma di sequenza: GET /Users	50
16	Diagramma di sequenza: GET /Users/{email}	51
17	Diagramma di sequenza: DELETE /Users/deleteUser/{email}	51
18	Diagramma di sequenza: PUT /Users/updateUser/{email}	52
19	Diagramma di sequenza: GET /User/SearchUser/{value}	53
20	Diagramma di sequenza: POST /Users/forgotPassword	53
21	Diagramma di sequenza: PUT /Users/promoteUser	54
22	Diagramma di sequenza: PUT /Users/demoteUser	55
23	Diagramma di sequenza: POST /Users/sendInvite	55
24	Diagramma di sequenza: POST /SuperAdmin/login	56
25	Diagramma di sequenza: POST /SuperAdmin/logout	57
26	Diagramma di sequenza: POST /SuperAdmin/impersonateUser/{userRole,email}	57
27	Diagramma di sequenza: GET /Companies	58
28	Diagramma di sequenza: POST /Companies	59
29	Diagramma di sequenza: GET /Companies/{companyName}	59
30	Diagramma di sequenza: PUT /Companies/{companyName}	60
31	Diagramma di sequenza: DELETE /Companies/deleteCompanies/{companyName}	61
32	Diagramma di sequenza: GET /Companies/searchCompany/{value}	61
33	Diagramma di sequenza: POST /ExternalDatabases	62
34	Diagramma di sequenza: DELETE /ExternalDatabases/{id}	63
35	Diagramma di sequenza: GET /ExternalDatabases/query	63
36	Diagramma di sequenza: GET /ExternalDatabases/searchExternalDatabase/value, companyName	64
37	Diagramma di sequenza: POST /ExternalDatabases/allowExternalDatabasesAccess	64
38	Diagramma di sequenza: POST /ExternalDatabases/denyExternalDatabaseAccess	65
39	Diagramma di sequenza: GET /Collections	66
40	Diagramma di sequenza: POST /Collections	66
41	Diagramma di sequenza: PUT /Collections/{id}	67
42	Diagramma di sequenza: DELETE /Collections/{id}	67
43	Diagramma di sequenza: GET /Collections/retrieveCollectionDSLIS/{id}	68
44	Diagramma di sequenza: GET /Collections/searchCollections/{id}	69
45	Diagramma di sequenza: GET /Collections/ExecuteCollection/{id}	69
46	Diagramma di sequenza: GET /Collections/SendEmail{id}	70
47	Diagramma di sequenza: GET /Collections/Export/{id}	71
48	Diagramma di sequenza: GET /Collections/exportCollectionDSLIS	71
49	Diagramma di sequenza: POST /Collections/importCollectionDSLIS	72
50	Diagramma di sequenza: GET /Documents	73
51	Diagramma di sequenza: POST /Documents	73
52	Diagramma di sequenza: PUT /Documents/{id}	74
53	Diagramma di sequenza: DELETE /Documents/{id}	75

54	Diagramma di sequenza: GET /Documents/searchDocuments/{id}	75
55	Diagramma di sequenza: GET /Documents/ExecuteDocument/{id}	76
56	Diagramma di sequenza: GET /Documents/ExecuteDocument/{CollectionName}/{id}	77
57	Diagramma di sequenza: GET /Documents/SendEmail{id}	77
58	Diagramma di sequenza: GET /Documents/Export/{id}	78
59	Diagramma di sequenza: GET /Documents/retrieveDocumentDSLIS	79
60	Diagramma di sequenza: GET /Documents/exportDocumentDSLIS	79
61	Diagramma di sequenza: POST /Documents/importDocumentDSLIS	80
62	Diagramma di sequenza: GET /Dashboards	81
63	Diagramma di sequenza: POST /Dashboards	81
64	Diagramma di sequenza: PUT /Dashboards/{id}	82
65	Diagramma di sequenza: DELETE /Dashboards/{id}	83
66	Diagramma di sequenza: GET /Dashboards/retrieveDashboardDSLIS/{id}	83
67	Diagramma di sequenza: GET /Dashboards/searchDashboards/{id}	84
68	Diagramma di sequenza: GET /Dashboards/ExecuteDashboard/{id}	85
69	Diagramma di sequenza: GET /Dashboards/exportDashboardDSLIS{id}	85
70	Diagramma di sequenza: POST /Dashboards/importDashboardDSLIS	86
71	Diagramma di sequenza: GET /Cells	87
72	Diagramma di sequenza: POST /Cells	87
73	Diagramma di sequenza: PUT /Cells/{id}	88
74	Diagramma di sequenza: DELETE /Cells/{id}	89
75	Diagramma di sequenza: GET /Cells/searchCell/{id}	90
76	Diagramma di sequenza: GET /Cells/ExecuteCell/{id}	91
77	Diagramma di sequenza: GET /Cells/GET /Cells/retrieveCellDSLIS{id}	92
78	Diagramma di sequenza: GET /Cells/exportCellDSLIS/{id}	93
79	Diagramma di sequenza: POST /Cells/importCellDSLIS	93
80	Diagramma delle classi del Action creators	96
81	Diagramma delle classi del DSLISActionCreator	100
82	Diagramma delle classi del WEBAPIUtils	103
83	Diagramma delle classi del DSLISWebAPIUtils	105
84	Diagramma delle classi dello stores	107
85	Diagramma delle classi del DSLISSStores	110
86	Diagramma delle classi del Views	113
87	Diagramma delle classi del package View	114
88	Attività principali del sistema MaaS	126
89	Attività di Registrazione al sistema MaaS	128
90	Attività di Autenticazione	129
91	Attività di Recupero Password	130
92	Attività di Reset Password	131
93	Attività di LogIn	132
94	Attività di Modifica delle impostazioni personali	133
95	Attività di Modifica dei dati personali	134
96	Attività di Modifica della password dopo l'accesso al profilo	135
97	Attività di Modifica della Dashboard principale dell'account	136
98	Attività di Creazione di un nuovo DSLIS	137
99	Attività di Modifica di un DSLIS esistente	138
100	Attività di Esecuzione di un DSLIS esistente	138
101	Attività di Eliminazione di un DSLIS esistente	139
102	Attività di invito di un nuovo utente	140

103	Attività di modifica del ruolo di un utente	141
104	Attività di eliminazione di un utente	142
105	Attività di modifica dei permessi per specifici DSLIS	143
106	Attività di modifica dei permessi di accesso al database	144
107	Attività di connessione del database aziendale	145
108	Attività di impersonificazione di un utente	146
109	Struttura logica del pattern Flux	148
110	Struttura logica del pattern Singleton	149
111	Struttura logica del pattern Factory Method	150
112	Struttura logica del pattern Abstract Factory	151
113	Struttura logica del pattern Facade	152
114	Struttura logica del pattern Chain of Responsibility	153
115	Struttura logica del pattern Template Method	154

Introduzione

Scopo del documento

Il documento ha lo scopo di definire la progettazione ad alto livello del prodotto software. Vengono illustrate le tecnologie, le componenti, le classi e i diversi design pattern adottati per la realizzazione di MaaS. Inoltre, verrà definito il tracciamento tra le componenti software ed i requisiti.

Scopo del prodotto

Lo scopo del progetto è quello di realizzare un sistema chiamato MaaS, fruibile dal web (e quindi distribuito), rivolto ai cosiddetti business man, persone con ruolo aziendale chiave che solitamente non sono esperte d'informatica, per aiutarli nelle decisioni di natura amministrativa e commerciale, fornendogli una piattaforma di visualizzazione dei dati salvati in un *database_G* di facile utilizzo, ma allo stesso tempo dalle buone potenzialità.

Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio impiegato nei documenti viene fornito il *Glossario v1.0.0*, contenente la definizione dei termini marcati con una G pedice.

Riferimenti

Normativi

Informativi

- **Interfaccia REST:**

https://it.wikipedia.org/wiki/Representational_State_Transfer
<http://rest.elkstein.org/>

Tecnologie utilizzate

In questa sezione vengono illustrate le tecnologie adottate per lo sviluppo del progetto. Per ognuna di esse, verrà indicato l'ambito di utilizzo ed i vantaggi che ne derivano.

Node.js

Node.js è un framework event-driven costruito sul motore JavaScript V8 di Google Chrome, per piattaforme UNIX like. Node.js usa un modello I/O non bloccante e ad eventi, che lo rendono un framework leggero ed efficiente per l'utilizzo server-side.

Vantaggi

- **Approccio asincrono:** grazie alla modalità event-drive node.js evita il classico modello basato su processi o thread concorrenti utilizzato dai classi web server. Tale caratteristica garantisce una migliore efficienza in termini di prestazioni, permettendo la gestione di altri processi durante le attese runtime in maniera asincrona.
- **Architettura modulare:** node.js si appoggia allo strumento node package manager (npm) che permette un'organizzazione semplice del lavoro grazie alla combinazione di librerie con moduli importati. Consentendo allo sviluppatore quanto di contribuire e accedere ai package messi a disposizione dalla community.

Le applicazioni Node.js vengono eseguite su un singolo thread, sebbene sia utilizzato un modello multi-thread per la gestione degli eventi legati a file e connessioni di rete.

MongoDB

MongoDb è un database NoSQL orientato ai documenti, cross-platform e open source; in MongoDB i dati vengono salvati sotto forma di documenti in stile JSON con schemi dinamici chiamati BSON.

Vantaggi

- **Schemaless:** non esiste uno schema e questo rende MongoDB molto più flessibile rispetto ad un tradizionale database relazionale;
- **Scalabilità orizzontale:** capacità di crescere o diminuire la scala in funzione alla reale necessità e delle disponibilità;
- **Affidabilità:** la caratteristica che rende MongoDb affidabile è la replicazione su server e la facilità con cui si configura, in particolare consente di proteggere i dati nel caso in cui avvengano fallimenti hardware, software o infrastrutturali;
- **Supporto del polimorfismo:** caratteristica che consente di estendere il proprio modello in modo semplice e veloce;

- **Alte performance:** l'assenza di join permette di non rallentare le operazioni di lettura e scrittura; l'indicizzazione inoltre include gli indici di chiave anche sui documenti innestati e sugli array, permettendo così una rapida interrogazione al database.

React

React è una libreria JavaScript utile alla costruzione di interfacce grafiche e il suo utilizzo è stato richiesto dal proponente. Esso sarà utilizzato per lo sviluppo del front-end, quindi del lato client, poiché fornisce molti strumenti utili alla progettazione, manutenzione e aggiornamento delle UI. In particolare supporta la definizione di modelli HTML sulla base di uno sviluppo a componenti.

Vantaggi

- Sviluppo per componenti: ogni componente è in grado di definire le proprie caratteristiche, quindi l'aspetto, il proprio stato e la propria parte logica;
- Permette una facile interoperabilità client-server;
- Può effettuare rendering sia sul client che sul server;
- Permette un aggiornamento efficiente della pagina quando cambia lo stato dei dati.

LoopBack

LoopBack è un framework di alto livello basato su node.js che consente di creare API permettendo il collegamento con sorgenti di dati; esso è creato a partire da Express e può, partendo da un modello di dati, generare facilmente delle API REST completamente funzionanti che possono venire chiamate da un qualsiasi client.

Vantaggi

- **Creazione di relazione tra modelli:** LoopBack consente di creare in maniera semplice delle relazioni tra modelli;
- **Supporto di più database:** LoopBack supporta molti database come ad esempio MySQL, Oracle, SQL Server e MongoDB; inoltre è possibile collegare diversi modelli a diversi tipi di database facendoli poi interagire tra loro;
- **Semplice definizione dei ruoli:** LoopBack integra la possibilità di mettere in relazione modelli ed utenti per mezzo della definizione di ACL(controlling data access);
- **Facilità di progettazione e implementazione:** essendo un'astrazione di Express, LoopBack consente di concentrarsi esclusivamente sulla logica che si vuole implementare evitando di focalizzare l'attenzione sulla progettazione di componenti quali, ad esempio, router.
- **Documentazione ufficiale:** chiara ed esaustiva.

HTML5

HTML5 è un linguaggio di markup per la strutturazione di pagine web, pubblicato dal W3C nell'ottobre del 2014 con lo scopo di migliorare l'Html4, permettendo supporto a nuovi file multimediali, mantenendo la leggibilità da parte di uomini e computer.

Vantaggi

- **Standard:** HTML5 è uno standard W3C.

CSS

CSS (Cascading Style Sheets) è un linguaggio utile a descrivere la presentazione di un documento scritto in un linguaggio di markup; uno degli scopi principali che hanno spinto alla creazione di questo linguaggio è il bisogno di separare il contenuto del documento dalla sua presentazione rendendo così maggiormente accessibili le pagine dei siti web.

Vantaggi

- **Standard:** CSS è uno standard W3C;
- **Accessibilità:** la separazione tra contenuto e presentazione consente di avere pagine molto più accessibili rispetto ad avere tutto nello stesso file.

Sweet.js

Sweet.js è un modulo che permette di definire macro utilizzando la sintassi di Javascript. Con esso è possibile modellare un linguaggio ad hoc basato su JavaScript, collezionare le macro in un singolo modulo e condividerlo sul Node Packaged Modules.

Vantaggi

- **Semplicità:** grazie a Sweet.js è possibile creare un linguaggio molto semplice.

Descrizione architetturale

Metodo e formalismo di specifica

Le scelte progettuali, adottate nello sviluppo di MaaS, sono state profondamente influenzate dalle tecnologie utilizzate.

In primo luogo il progetto è basato su Node.js ed è scritto quindi in JavaScript: un linguaggio che è orientato agli oggetti (OOP_G), ma che lascia grande libertà al programmatore nella scelta della tecnica da utilizzare per l'implementazione di pattern come l'*incapsulamento_G* e l'*ereditarietà_G*. Al contrario di altri linguaggi (C++, Java e derivati) non c'è un costrutto esplicito con il quale il programmatore può definire classi.

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

Per rappresentare l'utilizzo delle funzioni come parametro tipico della programmazione funzionale è stato necessario valutare se rappresentarlo come classe o se utilizzare una notazione particolare, dato che i diagrammi delle classi UML si adattano poco all'utilizzo di codice proveniente dalla programmazione funzionale. La prima opzione avrebbe richiesto di raddoppiare quasi il numero di classi progettate, quindi con l'intenzione di mantenere la specifica tecnica chiara e maneggevole si è scelto di utilizzare una notazione ad hoc. Tale notazione è della forma “`function(<parametri>)`” e rappresenta il tipo di dato di una funzione che richiede i parametri “`<parametri>`”.

Il nostro approccio alla progettazione è stato contemporaneamente top-down e bottom-up. Da un lato siamo partiti suddividendo il sistema in front-end e back-end, definendo l'interfaccia di comunicazione, scegliendo di seguire in ciascuno l'organizzazione suggeritaci dai framework (LoopBack e React). Dall'altro lato siamo partiti dal basso, componendo e cercando di riutilizzare il più possibile le librerie già esistenti. Per far questo abbiamo prima cercato e confrontato con attenzione la struttura e le scelte sia di progetti open source che di progetti proposti come best practice.

L'approccio top-down è stato schematizzato nei diagrammi di deployment e dei package.

Per descrivere il sistema sono stati utilizzati i diagrammi di sequenza e di attività, descrivendo l'interazione tra i singoli oggetti. In questo modo siamo riusciti a descrivere alcuni dei meccanismi tipici dell'applicazione, in particolar modo l'ordine in cui agiscono i *moduli_G* di LoopBack. Essi saranno molto utili per la progettazione di dettaglio e per la codifica.

I diagrammi di deployment, dei package, delle classi, di sequenza e di attività presentati di seguito utilizzano la specifica UML_G 2.0, incrementata con la convenzione per la rappresentazione delle funzioni. Nei diagrammi dei package in particolare, quelli colorati in verde rappresentano librerie esterne dei framework.

Architettura generale

L'architettura di MaaS si suddivide in due componenti interne principali e una esterna. La prima componente interna, Client, rappresenta il browser degli utenti che usufruiscono del servizio MaaS. Essi interagiranno con il front-end dell'applicazione.

La seconda componente, WebServer, rappresenta il server su cui viene implementato il back-end.

Entrambe le componenti interagiranno tra loro, utilizzando delle API messe a disposizione dal back-end, che andranno a rispettare i principi REST. La componente esterna, External Database Server, rappresenta i server su cui vengono ospitati i database esterni MongoDB. La configurazione di tale componente non è compito del progetto, in quanto all'applicativo MaaS interessa solo conoscere i dati di connessione dei database esterni, in modo da interfacciarsi con essi in modalità sola lettura.

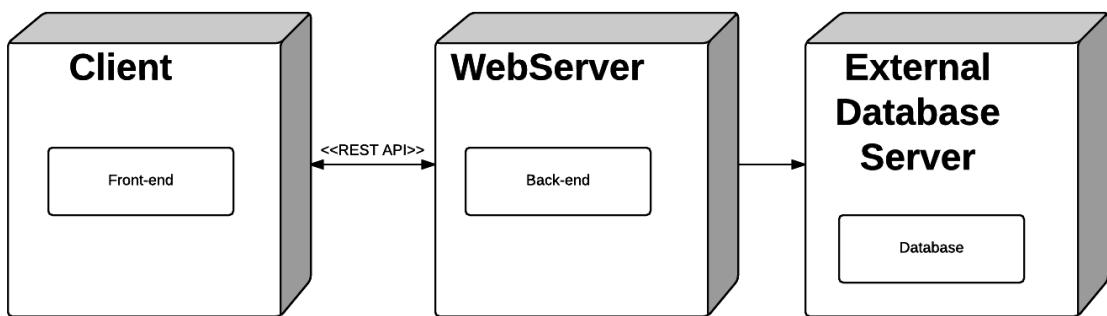


Figura 1: Diagramma del Deployment

Interfaccia REST

Per la realizzazione del back-end si è adottata una interfaccia che cerca di aderire il più possibile ai principi REST. I vantaggi che ci hanno portato a questa scelta sono:

- indipendenza dai linguaggi di programmazione;
- indipendenza dalle piattaforme software;
- utilizzo semplice in presenza di firewall;
- semplicità di utilizzo.

I componenti chiave di un'architettura REST sono:

- **Risorse:** sono identificate da URL logici. Lo stato dell'applicazione e le funzionalità sono divisi in risorse web;
- **Rete di risorse:** una singola risorsa non dovrebbe essere sovradimensionata e contenere dettagli troppo a grana fine;
- **Client-server:** un insieme di interfacce uniforme separa i client dai server. Server e client possono essere sostituiti e sviluppati indipendentemente fintanto che l'interfaccia non viene modificata;
- **Stateless:** la comunicazione client-server è ulteriormente vincolata in modo che nessun contesto client venga memorizzato sul server tra le richieste. Ogni client contiene tutte le informazioni necessarie per richiedere il servizio, e lo stato della sessione è contenuto sul client;

- **Cacheable:** come nel World Wide Web, i client possono fare caching delle risposte. Le risposte devono in ogni modo definirsi implicitamente o esplicitamente cacheable o no, in modo da prevenire che i client possano riusare stati vecchi o dati errati. Una gestione ben fatta della cache può ridurre o parzialmente eliminare le comunicazioni client server, migliorando scalabilità e performance;
- **Layered system:** un client non può dire se è connesso direttamente ad un server di livello più basso od intermedio, i server intermedi possono migliorare la scalabilità del sistema con load-balancing o con cache distribuite. Layer intermedi possono offrire inoltre politiche di sicurezza.

La seguente tabella mostra le relazioni tra gli URI ed i metodi HTTP:

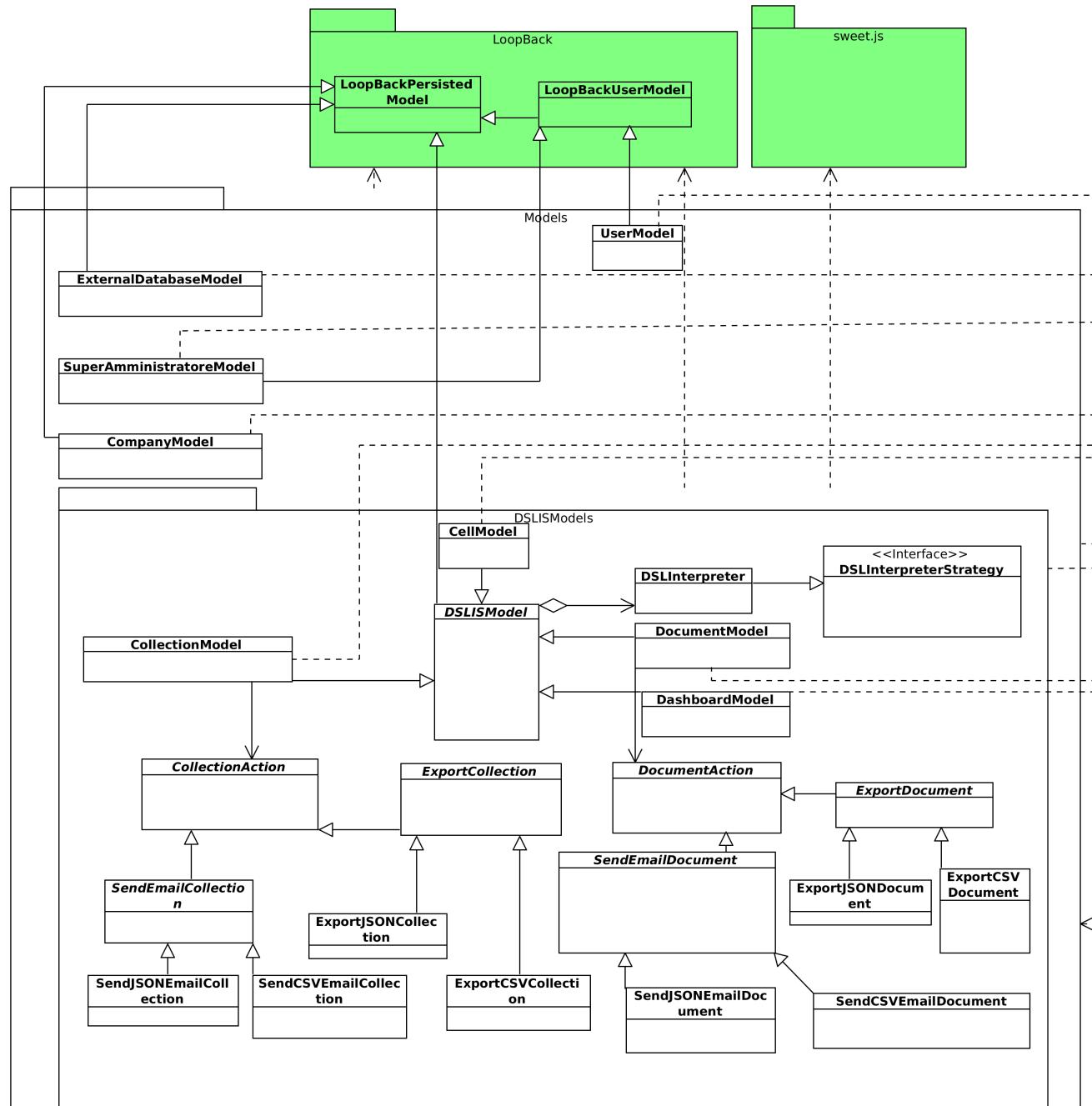
Risorsa	URI della collection es. http://example.com/resources/	URI di un elemento es. http://example.com/resources/item17
GET	Fornisce informazioni sui membri della collection.	Fornisce una rappresentazione dell'elemento della collection indicato, espresso in un appropriato formato.
PUT	Non usata.	Sostituisce l'elemento della collection indicato, o se non esiste, lo crea .
POST	Crea un nuovo elemento nella collection. La URI del nuovo elemento è generata automaticamente ed è di solito restituita dall'operazione.	Non usato.
DELETE	Non usata.	Cancella l'elemento della collection indicato.

I dati vengono rappresentati adottando il formato JSON, in quanto il suo utilizzo tramite JavaScript è particolarmente semplice.

Back-end

Descrizione packages e classi

Back-end



Pagina 16 di 154

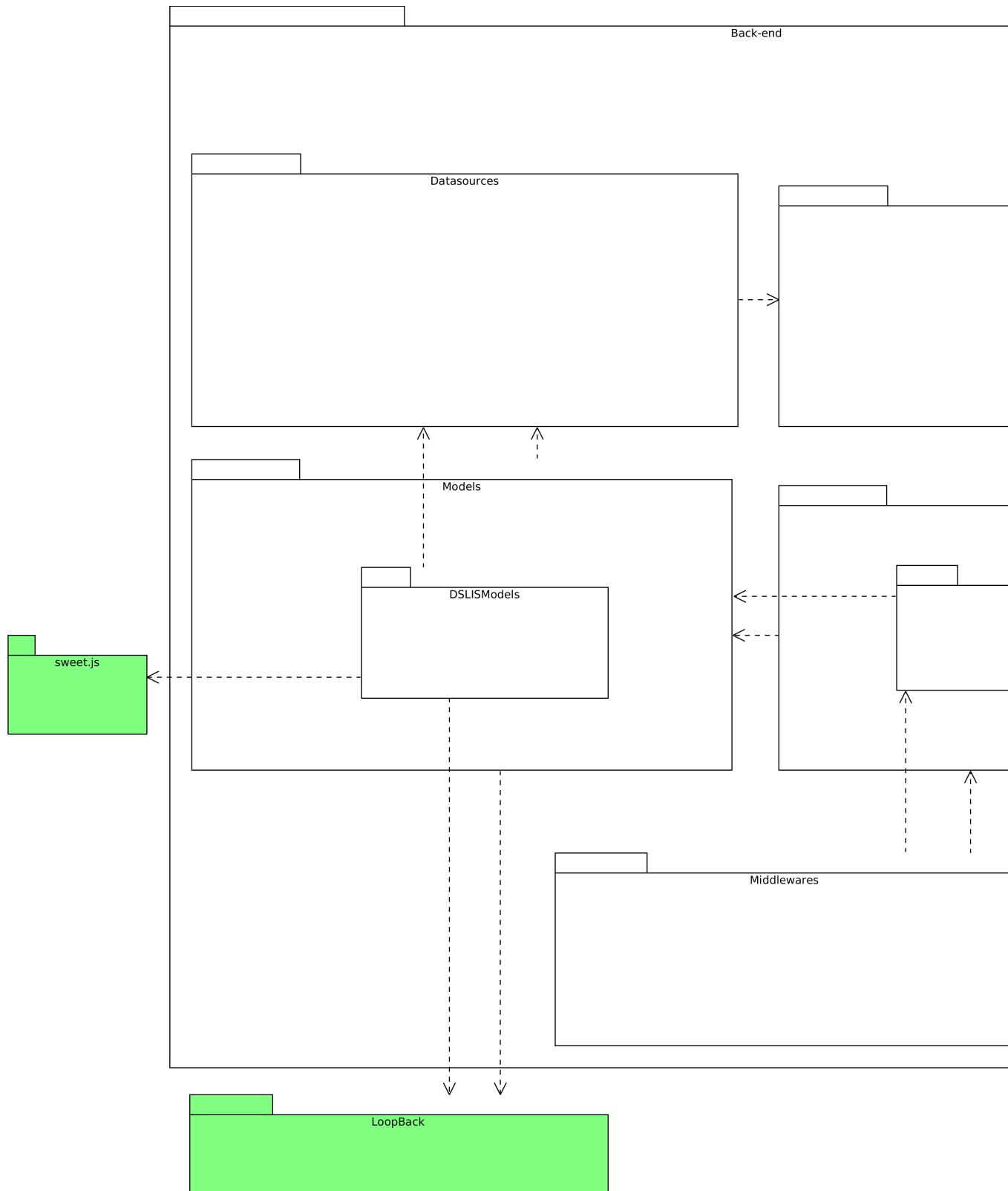
Specifica Tecnica

v 1.0.0

Middlewares

Router

processor



Informazioni sul package

4.1.1.1.1 Descrizione Package che racchiude tutte le componenti del Back-end. Comprende tutte le classi che rappresentano la parte server dell'applicativo. Le componenti sono organizzate secondo il design pattern previsto per il framework node.js LoopBack.

4.1.1.1.2 Package contenuti

- Models
- Connectors
- Datasources
- RESTAPIs
- Middlewares

4.1.1.1.3 Framework esterni

- LoopBack
- sweet.js

Classi

4.1.1.2.4 Application

Descrizione Classe che rappresenta la parte back-end dell'applicazione. Viene derivata dal framework LoopBack. Inoltre, rappresenta l'elemento client del design pattern Chain of Responsibility.

Utilizzo Viene utilizzata come elemento "main", per inizializzare le componenti del back-end. In particolare, si occupa di iniziare una richiesta ad un oggetto concreto della classe Back-end::Middlewares::MiddlewareHandler.

Relazioni con altre classi

- Back-end::Middlewares::MiddlewareHandler
- Back-end::Router
- Back-end::Bootstrapper

4.1.1.2.5 Router

Descrizione Classe che rappresenta la componente client del design pattern Abstract Factory.

Utilizzo Si occupa di ricevere le richieste HTTP da parte del client, ed in base all'URL utilizza i metodi offerti dalla classe Back-end::RESTAPIs::RESTAPIAbstractFactory per instanziare i corrispondenti oggetti dediti a soddisfare la natura della richiesta.

Relazioni con altri classi

- Back-end::RESTAPIs::RESTAPIAbstractFactory

4.1.1.2.6 Bootstrapper

Descrizione Classe che rappresenta gli script di boot eseguiti all'avvio dell'applicazione.

Utilizzo Viene utilizzata per fornire i metodi utili ad inizializzare l'applicazione, come le funzioni per configurare le impostazioni dell'applicazione ed aggiungere i dati che consentono l'avvio dell'applicazione. In particolare, inserisce i dati relativi al Super Amministratore nel database del sistema MaaS.

Back-end::Models

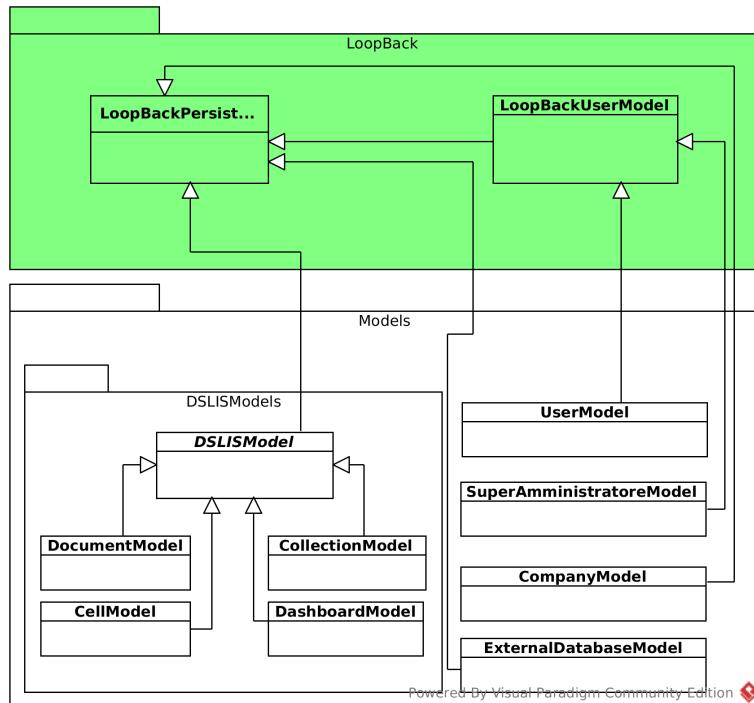


Figura 4: Diagramma delle classi del package Model

Informazioni sul package

4.1.2.1.1 Descrizione

Package contenente le componenti che rappresentano e gestiscono le risorse dei dati di back-end, come database, REST API e altri servizi di back-end.

4.1.2.1.2 Package contenuti

- DSLISModels

4.1.2.1.3 Framework esterni

- LoopBack
- sweet.js

Classi

4.1.2.2.4 UserModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito degli utenti.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito degli utenti. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- LoopBack::LoopBackUserModel

Relazioni con altre classi

- Back-end::Datasources::DatasourceFacade

4.1.2.2.5 SuperAmministratoreModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito del Super Amministratore.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito del Super Amministratore. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- LoopBack::LoopBackUserModel

Relazioni con altre classi

- Back-end::Datasources::DatasourceFacade

4.1.2.2.6 CompanyModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito delle aziende.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito delle aziende. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- LoopBack::LoopBackPersistedModel

Relazioni con altre classi

- Back-end::Datasources::DatasourceFacade

4.1.2.2.7 ExternalDatabaseModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito dei database MongoDB esterni.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito dei database MongoDB esterni. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- LoopBack::LoopBackPersistedModel

Relazioni con altre classi

- Back-end::Datasources::DatasourceFacade

Back-end::Models::DSLISModels

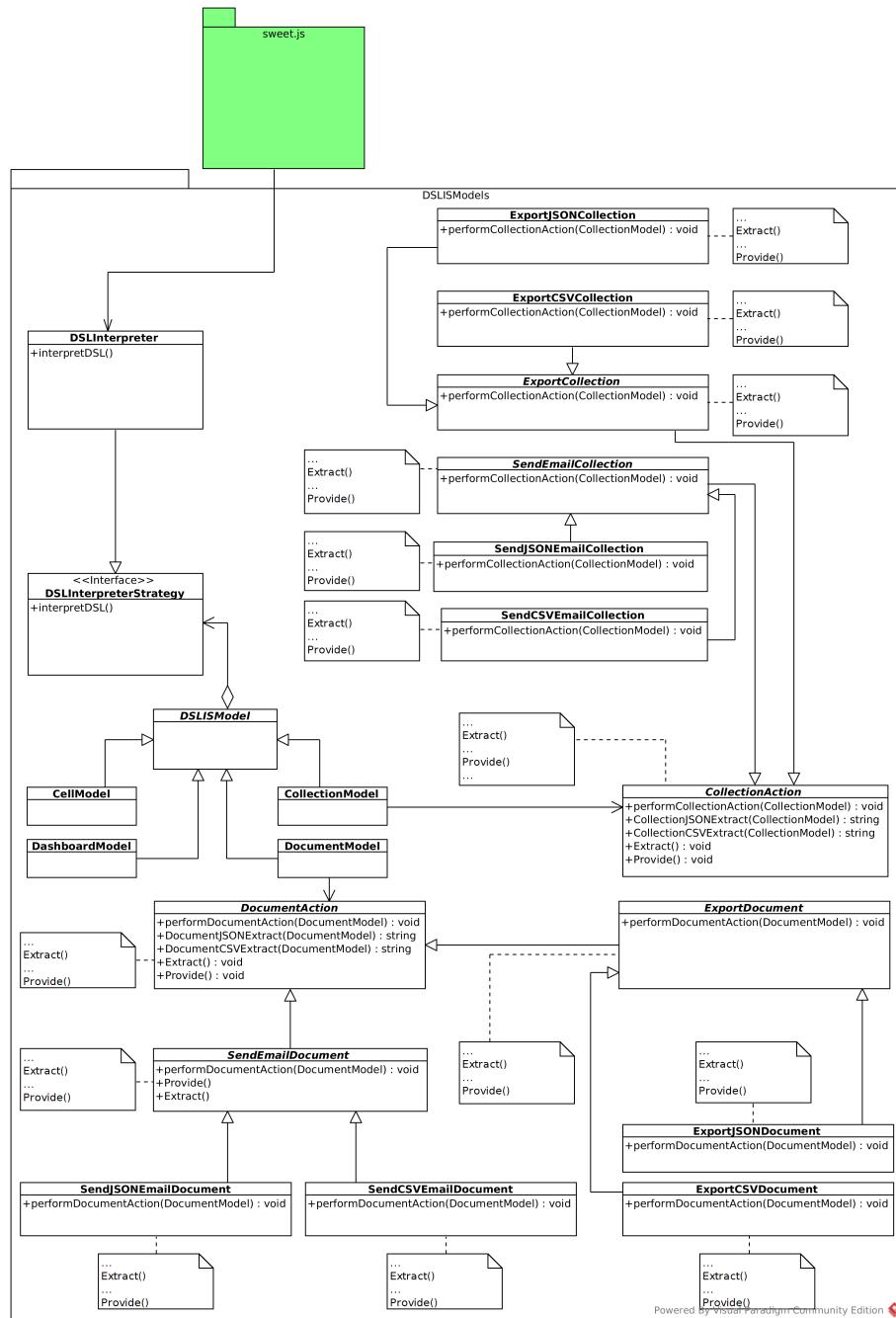


Figura 5: Diagramma delle figure del package: Back-end::Models::DSLISModels

Informazioni sul package

4.1.3.1.1 Descrizione

Package contenente le classi che rappresentano e gestiscono le risorse dei dati di back-end nell'ambito dei DSLIS.

4.1.3.1.2 Framework esterni

- LoopBack
- sweet.js

Classi

4.1.3.2.3 DSLISModel

Descrizione

Classe base astratta per i Model riguardanti i DSLIS. Viene derivata dalla classe LoopBackPersistedModel fornita dal framework LoopBack.

Utilizzo

Viene utilizzata come classe Context per utilizzare il metodo fornito dalla classe DSLInterpreterStrategy, appartenente al design pattern Strategy, per interpretare il DSLIS con un algoritmo, nel caso di necessità intercambiabile.

Estensioni

- Back-end::Models::DSLISModels::CollectionModel
- Back-end::Models::DSLISModels::CellModel
- Back-end::Models::DSLISModels::DocumentModel
- Back-end::Models::DSLISModels::DashboardModel

Classi ereditate

- LoopBack::LoopBackPersistedModel

Relazioni con altre classi

- Back-end::Models::DSLISModels::DSLInterpreter

4.1.3.2.4 CollectionModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito delle Collection, create mediante DSLIS.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito delle Collection. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- Back-end::Models::DSLISModels::DSLISModel

Relazioni con altre classi

- Back-end::Models::DSLISModels::CollectionAction
- Back-end::Datasources::DatasourceFacade

4.1.3.2.5 DocumentModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito dei Document, creati mediante DSLIS.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito dei Document. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- Back-end::Models::DSLISModels::DSLISModel

Relazioni con altre classi

- Back-end::Models::DSLISModels::DocumentAction
- Back-end::Datasources::DatasourceFacade

4.1.3.2.6 DashboardModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito delle Dashboard, create mediante DSLIS.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito delle Dashboard. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- Back-end::Models::DSLISModels::DSLISMModel

Relazioni con altre classi

- Back-end::Datasources::DatasourceFacade

4.1.3.2.7 CellModel

Descrizione

Classe che rappresenta la logica dell'applicazione nell'ambito delle Cell, create mediante DSLIS.

Utilizzo

Viene utilizzata per fornire le proprietà, gli ACL (Access Control Lists), le relazioni, i ruoli ed i metodi riguardanti l'ambito delle Cell. La classe sarà relazionata con il database MongoDB dell'applicazione, rappresentato dalla classe Back-end::Datasources::DatabaseDatasource.

Classi ereditate

- Back-end::Models::DSLISModels::DSLISMModel

Relazioni con altre classi

- Back-end::Datasources::DatasourceFacade

4.1.3.2.8 DSLInterpreterStrategy

Descrizione Classe astratta che definisce l'interfaccia dell'algoritmo di interpretazione del DSL utilizzato. E' il componente strategy del design pattern Strategy.

Utilizzo

Viene utilizzata per incapsulare e rendere intercambiabile l'algoritmo di interpretazione del DSL. In questo modo, se in futuro vi fosse necessità di cambiare il DSL e di conseguenza anche il relativo algoritmo di interpretazione, basterà variare esso indipendentemente dal client che ne farà uso.

Estensioni

- Back-end::Models::DSLISModels::DSLInterpreter

4.1.3.2.9 DSLInterpreter

Descrizione Classe che concretizza l'interprete del DSL. E' il componente ConcreteStrategy del design pattern Strategy.

Utilizzo Viene utilizzata per implementare l'algoritmo utilizzato nell'interfaccia Back-end::Models::DSLISModels::DSLInterpreterStrategy per l'interpretazione del linguaggio DSL. Conterrà al suo interno un metodo utile a generare il parser, a partire da una grammatica regolare.

Classi ereditate

- Back-end::Models::DSLISModels::DSLInterpreterStrategy

4.1.3.2.10 DocumentAction

Descrizione

Classe che rappresenta l'AbstractClass del design pattern Template Method per definire una Action di un Document.

Utilizzo

Definisce il metodo concreto ed i metodi primitivi astratti che verranno concretizzati dalle sottoclassi, utili ad eseguire una Action di un Document.

Estensioni

- Back-end::Models::DSLISModels::SendEmailDocument
- Back-end::Models::DSLISModels::ExportDocument

4.1.3.2.11 SendEmailDocument

Descrizione

Classe che rappresenta un ulteriore livello di AbstractClass del design pattern Template Method per definire una Action SendEmail per un Document.

Utilizzo

Definisce il metodo concreto ed i metodi primitivi astratti che verranno concretizzati dalle sottoclassi, utili ad eseguire una Action SendEmail per un Document.

Estensioni

- Back-end::Models::DSLISModels::SendJSONEmailDocument
- Back-end::Models::DSLISModels::SendCSVEmailDocument

Classe ereditate

- Back-end::Models::DSLISModels::DocumentAction

4.1.3.2.12 ExportDocument

Descrizione

Classe che rappresenta un ulteriore livello di AbstractClass del design pattern Template Method per definire una Action Export di un Document.

Utilizzo

Definisce il metodo concreto ed i metodi primitivi astratti che verranno concretizzati dalle sottoclassi, utili ad eseguire una Action Export di un Document.

Estensioni

- Back-end::Models::DSLISModels::ExportJSONDocument
- Back-end::Models::DSLISModels::ExportCSVDocument

Classe ereditate

- Back-end::Models::DSLISModels::DocumentAction

4.1.3.2.13 SendJSONEmailDocument

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action SendEmail per un Document, in modo che invii un file JSON.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action SendEmail per un Document, in modo che invii un file JSON.

Classe ereditate

- Back-end::Models::DSLISModels::SendEmailDocument

4.1.3.2.14 SendCSVEmailDocument

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action SendEmail per un Document, in modo che invii un file CSV.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action SendEmail per un Document, in modo che invii un file CSV.

Classe ereditate

- Back-end::Models::DSLISModels::SendEmailDocument

4.1.3.2.15 ExportJSONDocument

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action Export per un Document, in modo che invii un file JSON.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action Export per un Document, in modo che invii un file JSON.

Classe ereditate

- Back-end::Models::DSLISModels::ExportDocument

4.1.3.2.16 ExportCSVDocument

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action Export per un Document, in modo che invii un file CSV.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action Export per un Document, in modo che invii un file CSV.

Classe ereditate

- Back-end::Models::DSLISModels::ExportDocument

4.1.3.2.17 CollectionAction

Descrizione

Classe che rappresenta l'AbstractClass del design pattern Template Method per definire una Action di una Collection.

Utilizzo

Definisce il metodo concreto ed i metodi primitivi astratti che verranno concretizzati dalle sottoclassi, utili ad eseguire una Action di una Collection.

Estensioni

- Back-end::Models::DSLISModels::SendEmailCollection
- Back-end::Models::DSLISModels::ExportCollection

4.1.3.2.18 SendEmailCollection

Descrizione

Classe che rappresenta un ulteriore livello di AbstractClass del design pattern Template Method per definire una Action SendEmail per una Collection.

Utilizzo

Definisce il metodo concreto ed i metodi primitivi astratti che verranno concretizzati dalle sottoclassi, utili ad eseguire una Action SendEmail per una Collection.

Estensioni

- Back-end::Models::DSLISModels::SendJSONEmailCollection
- Back-end::Models::DSLISModels::SendCSVEmailCollection

Classe ereditate

- Back-end::Models::DSLISModels::CollectionAction

4.1.3.2.19 ExportCollection

Descrizione

Classe che rappresenta un ulteriore livello di AbstractClass del design pattern Template Method per definire una Action Export di una Collection.

Utilizzo

Definisce il metodo concreto ed i metodi primitivi astratti che verranno concretizzati dalle sottoclassi, utili ad eseguire una Action Export di una Collection.

Estensioni

- Back-end::Models::DSLISModels::ExportJSONCollection
- Back-end::Models::DSLISModels::ExportCSVCollection

Classe ereditate

- Back-end::Models::DSLISModels::CollectionAction

4.1.3.2.20 SendJSONEmailCollection

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action SendEmail per una Collection, in modo che invii un file JSON.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action SendEmail per una Collection, in modo che invii un file JSON.

Classe ereditate

- Back-end::Models::DSLISModels::SendEmailCollection

4.1.3.2.21 SendCSVEmailCollection

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action SendEmail per una Collection, in modo che invii un file CSV.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action SendEmail per una Collection, in modo che invii un file CSV.

Classe ereditate

- Back-end::Models::DSLISModels::SendEmailCollection

4.1.3.2.22 ExportJSONCollection

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action Export per una Collection, in modo che invii un file JSON.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action Export per una Collection, in modo che invii un file JSON.

Classe ereditate

- Back-end::Models::DSLISModels::ExportCollection

4.1.3.2.23 ExportCSVCollection

Descrizione

Classe che rappresenta la ConcreteClass del design pattern Template Method per definire una Action Export per una Collection, in modo che invii un file CSV.

Utilizzo

Implementa i metodi primitivi per svolgere i passi specifici, utili ad eseguire una Action Export per una Collection, in modo che invii un file CSV.

Classe ereditate

- Back-end::Models::DSLISModels::ExportCollection

Back-end::Datasources

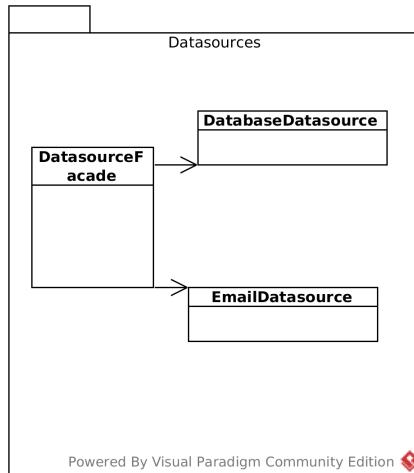


Figura 6: Diagramma delle classi del datasource

Informazioni sul package

4.1.4.1.1 Descrizione

Package contenente le classi che rappresentano i servizi di back-end, come per esempio i database.

Classi

4.1.4.2.2 DatasourceFacade

Descrizione

Classe che rappresenta l'implementazione del design pattern Facade per le classi contenute nel package Back-end::Datasources.

Utilizzo

Viene utilizzata per fornire un'interfaccia comune alle classi Datasource.

Relazioni con altre classi

- Back-end::Datasources::DatabaseDatasource
- Back-end::Datasources::EmailDatasource

4.1.4.2.3 DatabaseDatasource

Descrizione

Classe che rappresenta il database MongoDB dell'applicazione.

Utilizzo

Viene utilizzata per fornire i metodi per connettersi al database dell'applicazione ed impostare le opzioni di connessione. Rappresenta il supporto di memorizzazione dei dati delle classi dichiarate nel package Back-end::Models. La classe utilizza i metodi offerti dalla classe Back-end::Connectors::MongoDBConnector, per gestire le richieste delle classi presenti nel package Back-end::Models riguardanti i dati del database di MaaS.

Relazioni con altre classi

- Back-end::Connectors::MongoDBConnector

4.1.4.2.4 EmailDatasource

Descrizione

Classe che rappresenta il servizio email associato all'applicazione.

Utilizzo

Viene utilizzata per fornire i metodi per connettersi al servizio email dell'applicazione ed impostare le opzioni di connessione. Inoltre, fornisce il metodo per inviare un'email. Rappresenta il supporto per l'invio di Email, utile alle classi contenute nel package Back-end::Models.

Relazioni con altre classi

- Back-end::Connectors::EmailConnector

Back-end::Connectors

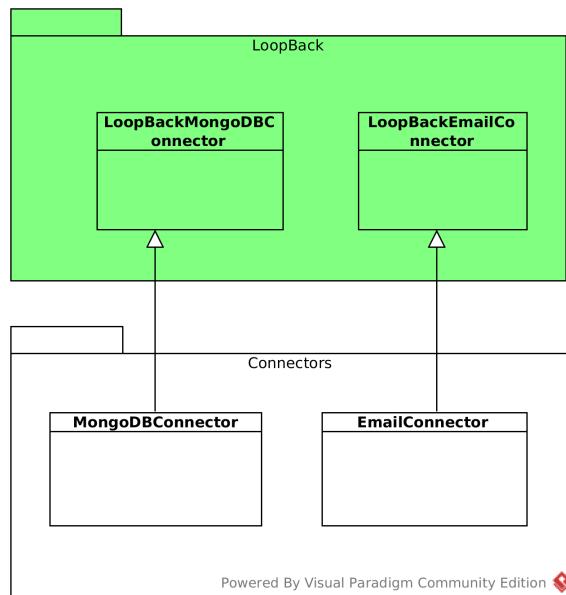


Figura 7: Diagramma delle classi dei connectors

Informazioni sul package

4.1.5.1.1 Descrizione

Package contenente le classi derivate dal framework LoopBack, che rappresentano i connettori alle risorse ed ai servizi di back-end e si occupano di comunicarci direttamente.

4.1.5.1.2 Framework esterni

- LoopBack

Classi

4.1.5.2.3 MongoDBConnector

Descrizione

Classe che contiene i metodi per interfacciarsi e gestire il database MongoDB dell'applicazione. Essa viene derivata dal connettore MongoDB offerto dal framework LoopBack.

Utilizzo

Viene utilizzata per fornire i metodi utili ad eseguire le operazioni sul database MongoDB dell'applicazione. Essa utilizza il metodo di connessione, con i relativi parametri, della classe DatabaseDatasource per connettersi al database.

Classi ereditate

- LoopBack::LoopBackMongoDBConnector

4.1.5.2.4 EmailConnector

Descrizione

Classe che contiene i metodi per interfacciarsi e gestire il servizio di email dell'applicazione. Essa viene derivata dal connettore Email offerto dal framework LoopBack.

Utilizzo

Viene utilizzata per fornire i metodi utili ad eseguire le operazioni sull'email dell'applicazione. Essa utilizza il metodo di connessione, con i relativi parametri, della classe EmailDatasource per connettersi al servizio email.

Classi ereditate

- LoopBack::LoopBackEmailConnector

Back-end::RESTAPIs

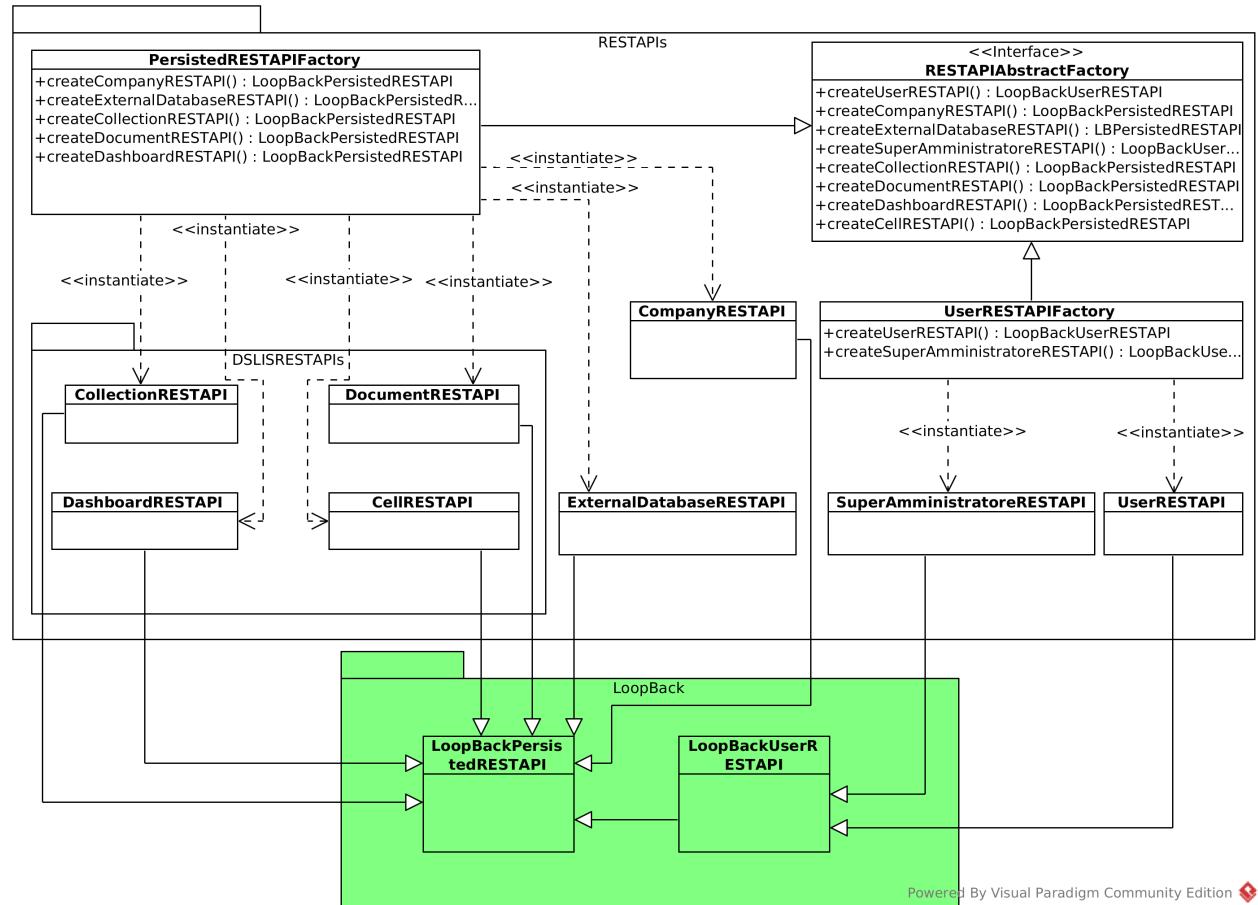


Figura 8: Diagramma delle classi del RESTAPIs

Informazioni sul package

4.1.6.1.1 Descrizione

Package contenente le classi che forniscono i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello. Le classi vengono derivate da quelle offerte dal framework LoopBack, in quanto sono strettamente associate ai Model e offrono una serie di metodi standard.

4.1.6.1.2 Package contenuti

- **DSLISRESTAPIs**

4.1.6.1.3 Framework esterni

- LoopBack

Classi

4.1.6.2.4 RESTAPIAbstractFactory

Descrizione

Classe rappresentante la componente AbstractFactory del design pattern Abstract Factory, utile per la creazione di oggetti RESTAPI.

Utilizzo

Definisce l'interfaccia per i metodi utili alla creazione delle classi RESTAPI, presenti nel package Back-end::RESTAPIs.

Estensioni

- Back-end::RESTAPIs::PersistedRESTAPIFactory
- Back-end::RESTAPIs::UserRESTAPIFactory

4.1.6.2.5 PersistedRESTAPIFactory

Descrizione

Classe rappresentante la componente ConcreteFactory del design pattern Abstract Factory, utile per la creazione di oggetti PersistedRESTAPI.

Utilizzo

Viene utilizzata per implementare i metodi utili a creare gli oggetti delle classi di tipo PersistedRESTAPI, presenti nel package Back-end::RESTAPIs.

Classi ereditate

- Back-end::RESTAPIs::RESTAPIAbstractFactory

Relazioni con altre classi

- Back-end::RESTAPIs::CompanyRESTAPI
- Back-end::RESTAPIs::ExternalDatabaseRESTAPI
-
- Back-end::RESTAPIs::DSLISRESTAPIs::CellRESTAPI

- Back-end::RESTAPIs::DSLISRESTAPIs::CollectionRESTAPI
- Back-end::RESTAPIs::DSLISRESTAPIs::DocumentRESTAPI
- Back-end::RESTAPIs::DSLISRESTAPIs::DashboardRESTAPI

4.1.6.2.6 UserRESTAPIFactory

Descrizione

Classe rappresentante la componente ConcreteFactory del design pattern Abstract Factory, utile per la creazione di oggetti UserRESTAPI.

Utilizzo

Viene utilizzata per implementare i metodi utili a creare gli oggetti delle classi di tipo UserRESTAPI, presenti nel package Back-end::RESTAPIs.

Classi ereditate

- Back-end::RESTAPIs::RESTAPIAbstractFactory

Relazioni con altre classi

- Back-end::RESTAPIs::UserRESTAPI
- Back-end::RESTAPIs::SuperAmministratoreRESTAPI

4.1.6.2.7 UserRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello, corrispondente alla classe UserModel. Essa viene derivata dalla classe LoopBackUserRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello UserModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello UserModel.

Classi ereditate

- LoopBack::LoopBackUserRESTAPI

4.1.6.2.8 SuperAmministratoreRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello, corrispondente alla classe SuperAmministratoreModel. Essa viene derivata dalla classe LoopBackUserRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello SuperAmministratoreModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello SuperAmministratoreModel.

Classi ereditate

- LoopBack::LoopBackUserRESTAPI

4.1.6.2.9 CompanyRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello, corrispondente alla classe CompanyModel. Essa viene derivata dalla classe LoopBackPersistedRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello CompanyModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello CompanyModel.

Classi ereditate

- LoopBack::LoopBackPersistedRESTAPI

4.1.6.2.10 ExternalDatabaseRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello, corrispondente alla classe ExternaldatabaseModel. Essa viene derivata dalla classe LoopBackPersistedRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello ExternalDatabaseModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello ExternalDatabaseModel.

Classi ereditate

- LoopBack::LoopBackPersistedRESTAPI

Back-end::RESTAPI::DSLISRESTAPIs

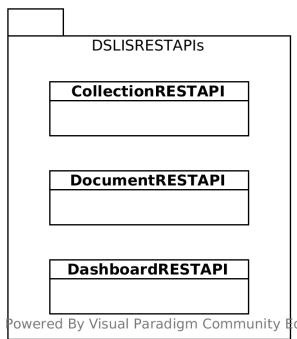


Figura 9: Diagramma delle classi del DLSRESTAPI

Informazioni sul package

4.1.7.1.1 Descrizione

Package contenente le classi che forniscono i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati dei modelli riguardanti i DSLIS. Le classi vengono derivate da quelle offerte dal framework LoopBack, in quanto sono strettamente associate ai Model e offrono una serie di metodi standard.

4.1.7.1.2 Framework esterni

- LoopBack

Classi

4.1.7.2.3 CollectionRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui

dati del modello, corrispondente alla classe CollectionModel. Essa viene derivata dalla classe LoopBackPersistedRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello CollectionModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello CollectionModel.

Classi ereditate

- LoopBack::LoopBackPersistedRESTAPI

4.1.7.2.4 DocumentRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello, corrispondente alla classe DocumentModel. Essa viene derivata dalla classe LoopBackPersistedRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello DocumentModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello DocumentModel.

Classi ereditate

- LoopBack::LoopBackPersistedRESTAPI

4.1.7.2.5 DashboardRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello, corrispondente alla classe DashboardModel. Essa viene derivata dalla classe LoopBackPersistedRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello DashboardModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello DashboardModel.

Classi ereditate

- LoopBack::LoopBackPersistedRESTAPI

4.1.7.2.6 CellRESTAPI

Descrizione

Classe che fornisce i metodi utili a mappare le richieste HTTP del client, con specifiche funzioni rappresentanti le REST API, che vanno ad implementare le operazioni CRUD sui dati del modello, corrispondente alla classe CellModel. Essa viene derivata dalla classe LoopBackPersistedRESTAPI fornita dal framework LoopBack, in quanto è strettamente associata ai dati del modello CellModel e offre una serie di metodi standard in questo ambito.

Utilizzo

Viene utilizzata per offrire i metodi utili a mappare le richieste HTTP del client, con le funzioni appartenenti al modello CellModel.

Classi ereditate

- LoopBack::LoopBackPersistedRESTAPI

Back-end::Middlewares

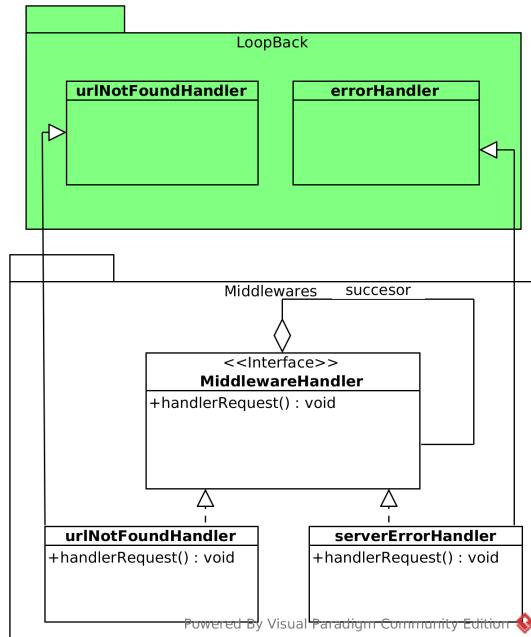


Figura 10: Diagramma delle classi del Middlewares

Informazioni sul package

4.1.8.1.1 Descrizione

Package contenente le classi che costituiscono gli handler della catena di chiamate utili a verificare la presenza di errori nelle richieste del client.

4.1.8.1.2 Framework esterni

- LoopBack

Classi

4.1.8.2.3 MiddlewareHandler

Descrizione

Classe che rappresenta la componente Handler del design pattern Chain of Responsibility.

Utilizzo

Viene utilizzata per fornire l'interfaccia per gestire le richieste.

Relazioni con altre classi

- Back-end::Middlewares::MiddlewareHandler

4.1.8.2.4 urlNotFoundHandler**Descrizione**

Classe che rappresenta la componente ConcreteHandler del design pattern Chain of Responsibility. Essa viene derivata dalla classe middleware offerta dal framework LoopBack Back-end::Middlewares::LoopBackurlNotFoundHandler.

Utilizzo

Viene utilizzata per gestire le richieste del client e verificare la presenza di errori riguardanti gli URL errati in esse.

Classi ereditate

- LoopBack::LoopBackurlNotFoundHandler

Relazioni con altre classi

- Back-end::Middlewares::MiddlewareHandler

4.1.8.2.5 serverErrorHandler**Descrizione**

Classe che rappresenta la componente ConcreteHandler del design pattern Chain of Responsibility. Essa viene derivata dalla classe middleware offerta dal framework LoopBack Back-end::Middlewares::LoopBackerrorHandler.

Utilizzo

Viene utilizzata per gestire le richieste del client e verificare la presenza di errori riguardanti l'impossibilità del server nel gestirle.

Classi ereditate

- LoopBack::LoopBackerrorHandler

Relazioni con altre classi

- Back-end::Middlewares::MiddlewareHandler

Interfaccia REST

Comunicazione tra client e server

Come visto in precedenza, per l'implementazione del *back-end_G* di *MaaS_G* si è deciso di utilizzare il framework *LoopBack_G* che permette in modo semplice e veloce di creare una solida interfaccia *REST_G*, generando automaticamente un insieme di *API_G* per ogni modello. Ad ogni API il server risponde con un oggetto in formato JSON per fornire le informazioni. Nell'eventualità fornisce invece un messaggio di errore, inviato con un pacchetto HTTP di tipo 4xx (errore nella richiesta del client) o 5xx (errore nella risposta del server), nel seguente formato JSON:

```
{  
    "code": [codice numerico dell'errore],  
    "message": [descrizione testuale dell'errore],  
    "data": [eventuali dati aggiuntivi sull'errore]  
}
```

Di seguito sono descritte le API del back-end, elencando per ogni risorsa REST il tipo di metodo che è possibile richiedere su di essa e i permessi richiesti per poter effettuare la richiesta.

Permessi

I tipi di permessi possibili sono:

- **Utente Non Autenticato:** questa risorsa può essere richiesta solo dagli utenti autenticati a MaaS;
- **Utente Autenticato:** questa risorsa può essere richiesta solo dagli utenti autenticati a MaaS, con qualsiasi ruolo;
- **Ospite:** tale risorsa può essere richiesta solo da utenti con il ruolo di Ospite.
- **Membro:** tale risorsa può essere richiesta solo da utenti con il ruolo di Membro.
- **Amministratore:** tale risorsa può essere richiesta solo da utenti con il ruolo di Amministratore.
- **Proprietario:** tale risorsa può essere richiesta solo da utenti con il ruolo di Proprietario.
- **Super Amministratore:** tale risorsa può essere richiesta solo da utenti con il ruolo di Super Amministratore.

Richieste HTTP

Richieste Users

4.2.3.1.1 Richiesta POST /Users

- **Descrizione:** Crea una richiesta di registrazione.
- **Richiesta HTTP:** /Users
- **Metodo:** POST
- **Permessi:** Utente non autenticato, Utente invitato.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa Users. Creata la UserRESTAPI il metodo createUsers() si occupa di gestire la richiesta.

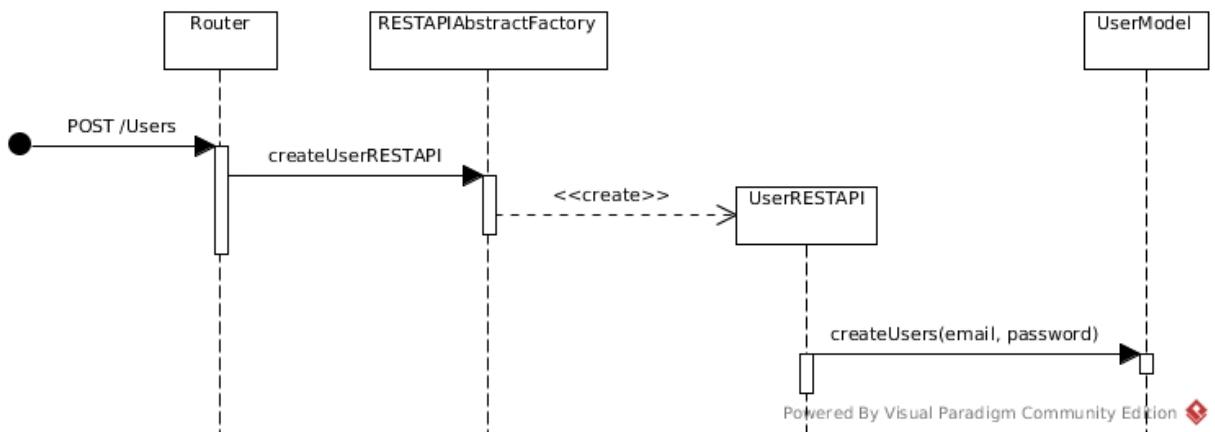


Figura 11: Diagramma di sequenza: POST /User

4.2.3.1.2 Richiesta POST /Users/login

- **Descrizione:** Esegue l'autenticazione e crea la sessione corrispondente all'utente.
- **Richiesta HTTP:** /Users/login
- **Permessi:** Utente non autenticato.
- **Metodo:** POST
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta POST per la risorsa /Users/login. Istanziate le UserRESTAPI la chiamata login(user) esegue l'autenticazione e crea la sessione per l'utente.

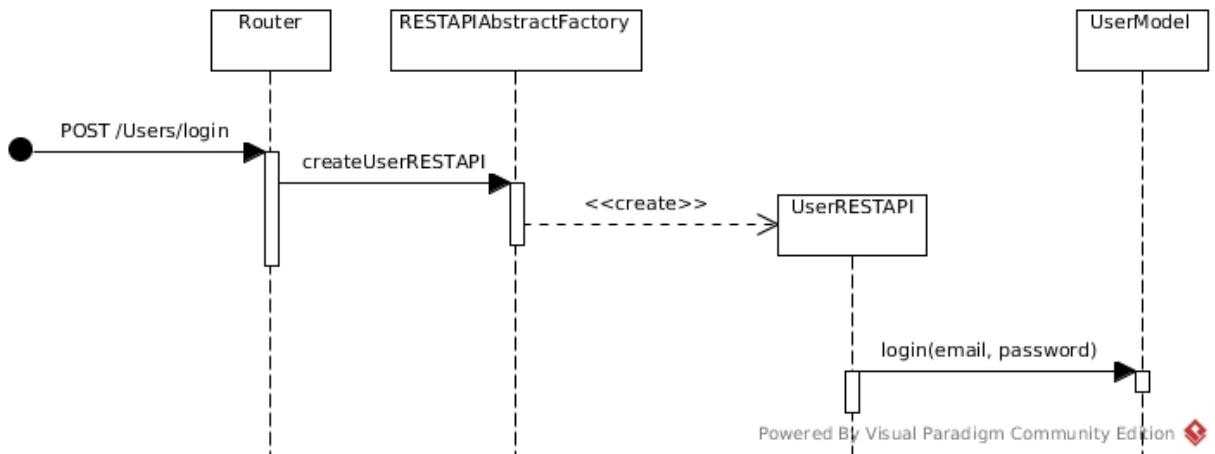


Figura 12: Diagramma di sequenza: POST /Users/login

4.2.3.1.3 Richiesta PUT /Users

- **Descrizione:** Modifica i dati dell'utente.
- **Richiesta HTTP:** /Users
- **Metodo:** PUT
- **Permessi:** Utente autenticato, Super Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta PUT per la risorsa Users. Creata la UserRESTAPI il metodo setUsers(user) permette la modifica dei dati dell'utente users.

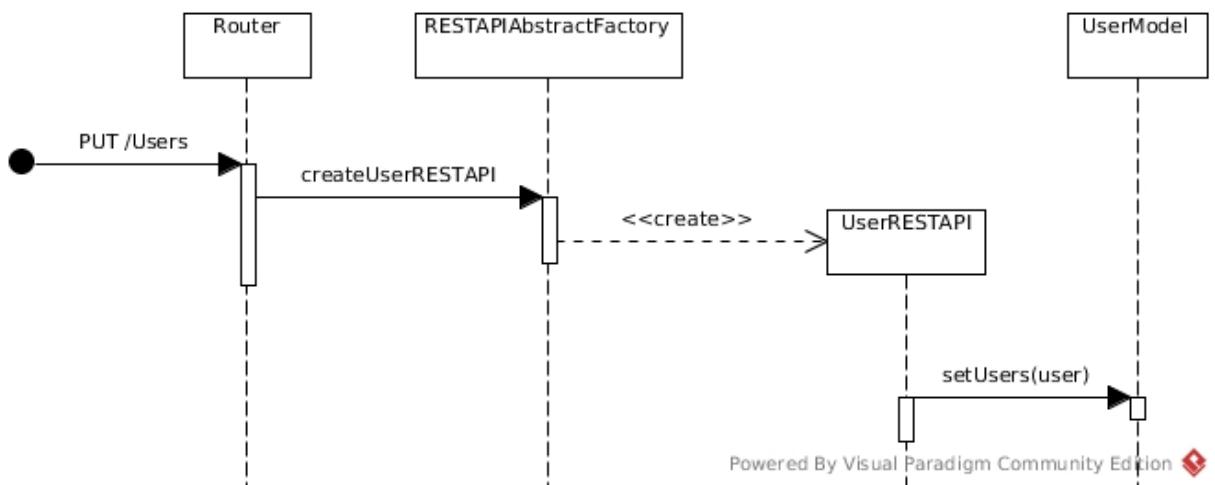


Figura 13: Diagramma di sequenza: PUT /Users

4.2.3.1.4 Richiesta POST /Users/logout

- **Descrizione:** Elimina la sessione utente, corrispondente al logout.
- **Richiesta HTTP:** /Users/logout
- **Metodo:** POST
- **Permessi:** Utente autenticato.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta POST per la risorsa /Users/logout. Istanziate le UserRESTAPI la chiamata logout(user) si occupa di effettuare il logout per l'utente user.

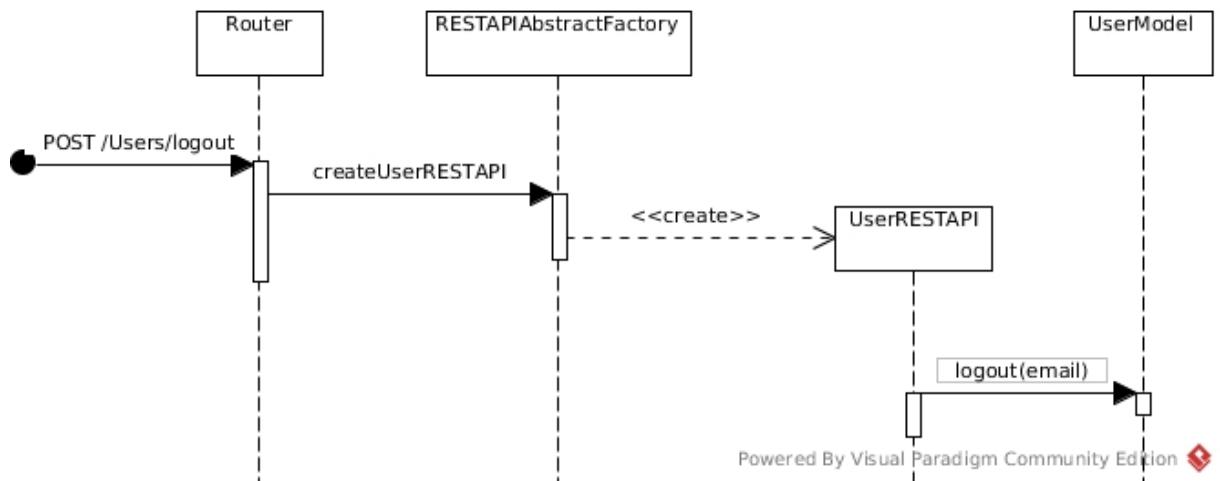


Figura 14: Diagramma di sequenza: POST /Users/logout

4.2.3.1.5 Richiesta GET /Users

- **Descrizione:** Restituisce i dati relativi agli utenti.
- **Richiesta HTTP:** /Users
- **Metodo:** GET
- **Permessi:** Proprietario, Amministratore, Super Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa Users. Creata la UserRESTAPI il metodo getUsers() chiede all'UserModel i dati di tutti gli utenti. L'UserModel passa i dati all'UserRESTAPI che li restituisce in formato JSON.

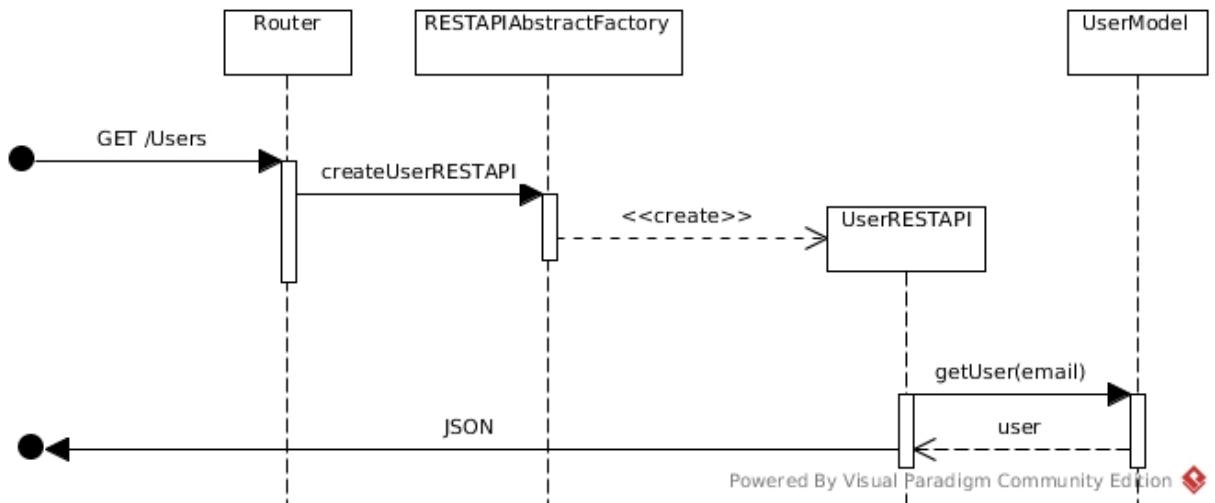


Figura 15: Diagramma di sequenza: GET /Users

4.2.3.1.6 Richiesta GET /Users/{email}

- Descrizione:** Restituisce i dati di un utente.
- Richiesta HTTP:** /Users/{email}
- Metodo:** GET
- Permessi:** Utente autenticato.
- Scenario:** Il seguente scenario mostra la gestione di una richiesta GET per la risorsa /Users/{email}. Istanziante le UserRESTAPI la chiamata getUsersData(email) chiede all'UserModel i dati di un utente specifico. I dati ricevuti saranno restituiti in formato JSON.

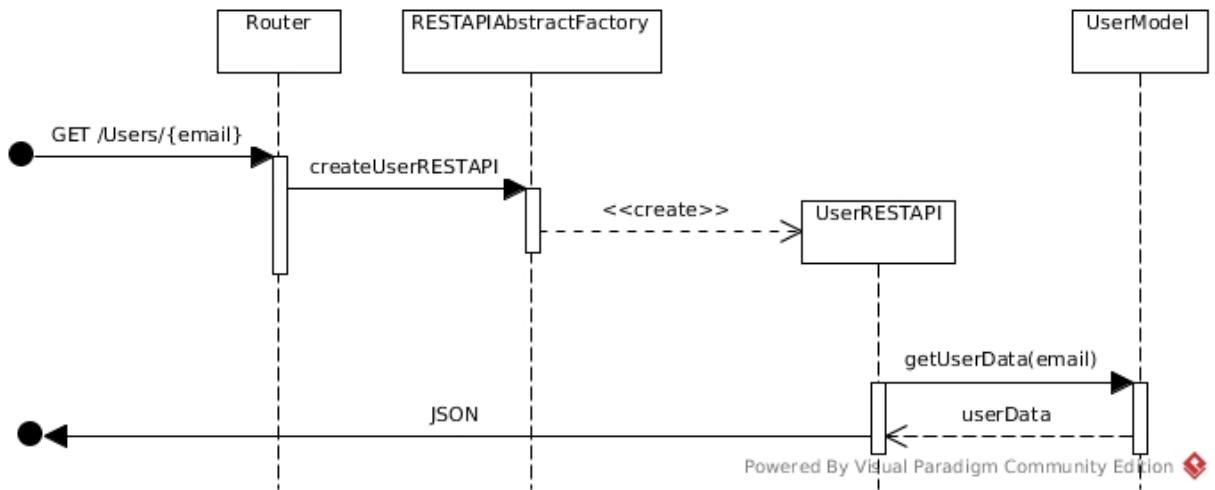


Figura 16: Diagramma di sequenza: GET /Users/{email}

4.2.3.1.7 Richiesta DELETE /Users/deleteUser/{email}

- **Descrizione:** Elimina un utente.
- **Richiesta HTTP:** /Users/deleteUser/{email}
- **Metodo:** DELETE
- **Permessi:** Proprietario, Amministratore, Super Amministratore.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta DELETE per la risorsa /Users/deleteUser/{email}. Istanziate le UserRESTAPI la chiamata deleteUser(email) viene eliminato un utente nell'UserModel.

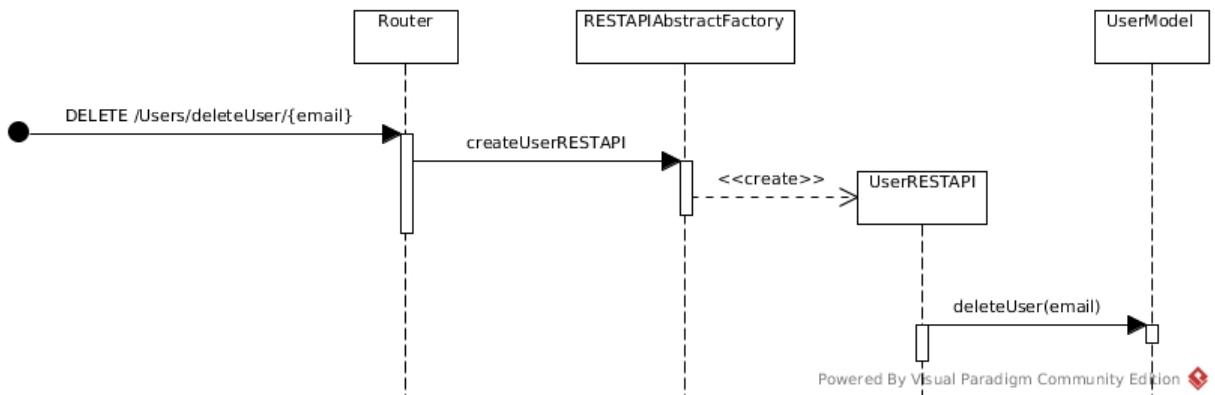


Figura 17: Diagramma di sequenza: DELETE /Users/deleteUser/{email}

4.2.3.1.8 Richiesta PUT /Users/updateUser/{email}

- **Descrizione:** Modifica i dati di un utente.

- **Richiesta HTTP:** /Users/updateUser/{email}
- **Metodo:** PUT
- **Permessi:** Super Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta PUT per la risorsa /Users/updateUser/{email}. Creata la UserRESTAPI il metodo setUsers(email) si occupa modificare i dati di un utente nel UserModel.

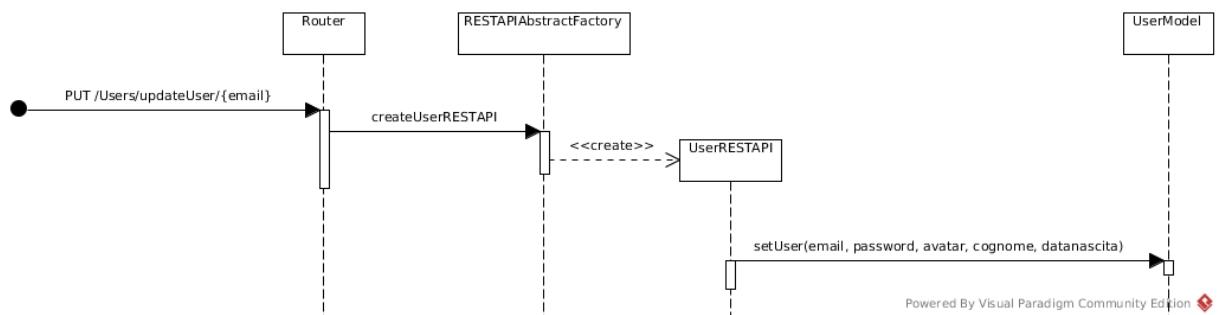


Figura 18: Diagramma di sequenza: PUT /Users/updateUser/{email}

4.2.3.1.9 Richiesta GET /User/SearchUser/{value}

- **Descrizione:** Dato un valore, ricerca un utente.
- **Richiesta HTTP:** /User/SearchUser/{value}
- **Metodo:** GET
- **Permessi:** Super Amministratore
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta GET per la risorsa /User/SearchUser/{value}. Istanziate le UserRESTAPI la chiamata searchUser(value) effettua la ricerca di un utente nell'UserModel.

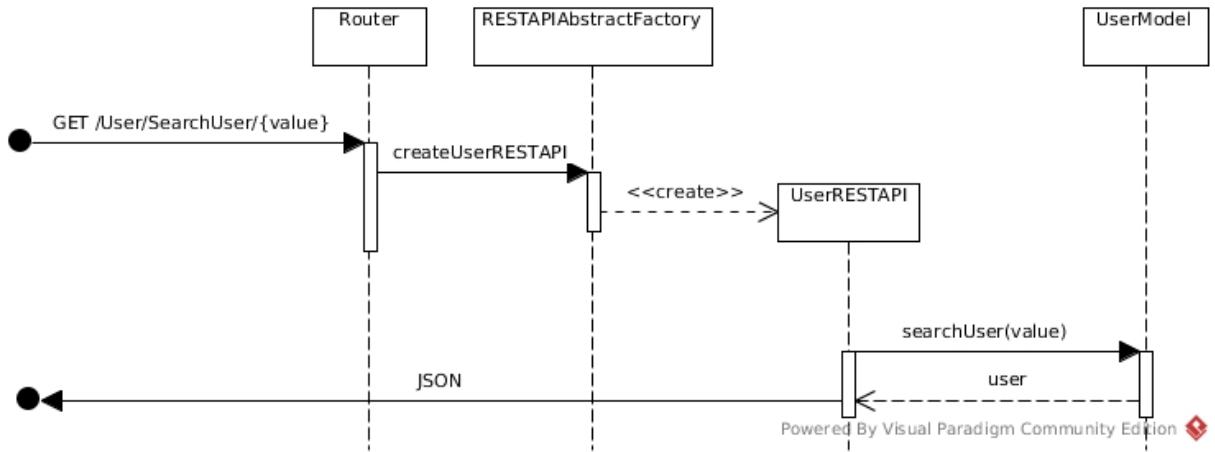


Figura 19: Diagramma di sequenza: GET /User/SearchUser/{value}

4.2.3.1.10 Richiesta POST /Users/forgotPassword

- **Descrizione:** Richiede il recupero della password.
- **Richiesta HTTP:** /Users/forgotPassword
- **Metodo:** POST
- **Permessi:** Utente non autenticato.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Users/forgotPassword. Creata la UserRESTAPI il metodo setUsers(email) si occupa richiedere il recupero della password all'UserModel.

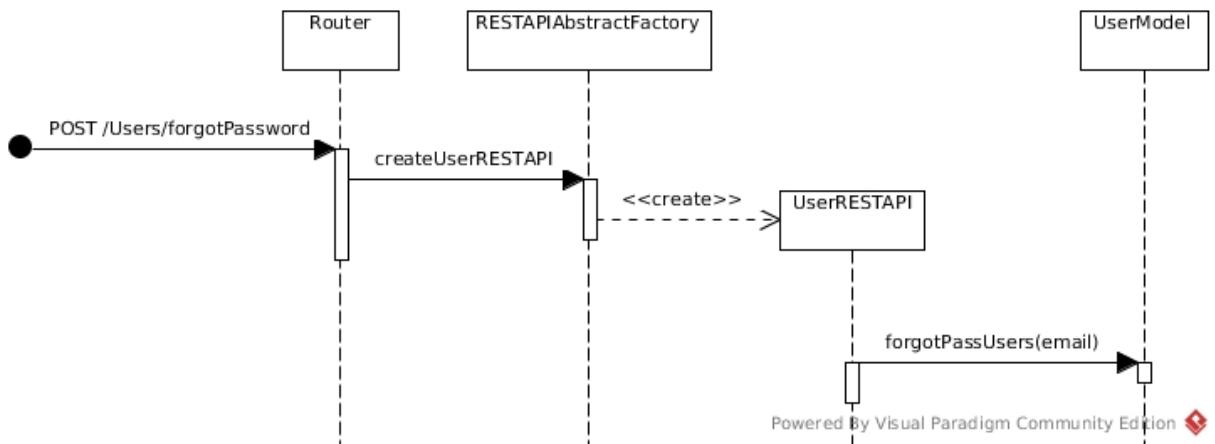


Figura 20: Diagramma di sequenza: POST /Users/forgotPassword

4.2.3.1.11 Richiesta PUT /Users/promoteUser

- **Descrizione:** Richiesta di promozione del ruolo di un utente.
- **Richiesta HTTP:** /Users/promoteUser
- **Metodo:** PUT
- **Permessi:** Proprietario, Amministratore.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta PUT per la risorsa /Users/promoteUser. Istanziate le UserRESTAPI la chiamata promoteUser(email,role) permette la promozione di un utente nell'UserModel.

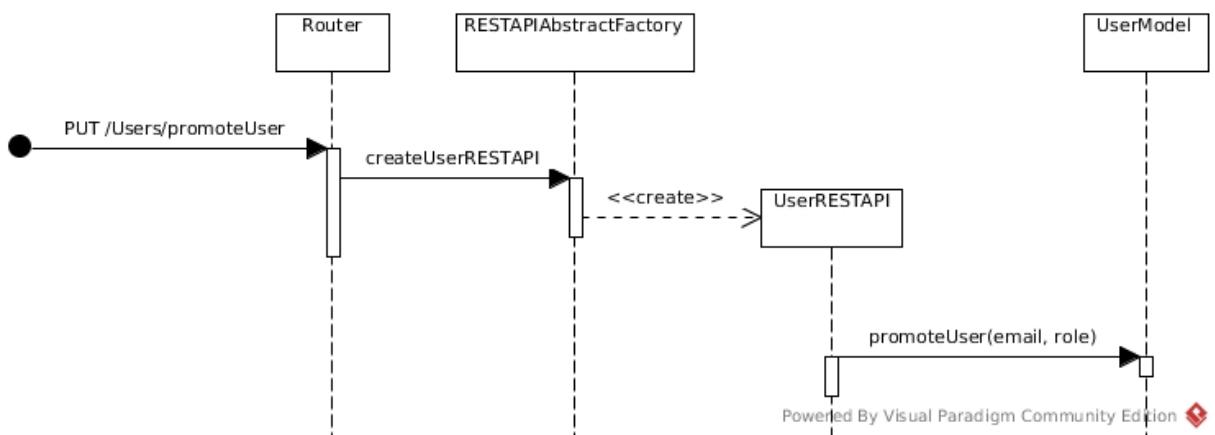


Figura 21: Diagramma di sequenza: PUT /Users/promoteUser

4.2.3.1.12 Richiesta PUT /Users/demoteUser

- **Descrizione:** Richiesta di retrocessione del ruolo di un utente.
- **Richiesta HTTP:** /Users/demoteUser
- **Metodo:** PUT
- **Permessi:** Proprietario, Amministratore.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta PUT per la risorsa /Users/demoteUser. Istanziate le UserRESTAPI la chiamata promoteUser(email,role) permette la retrocessione di un utente nell'UserModel.

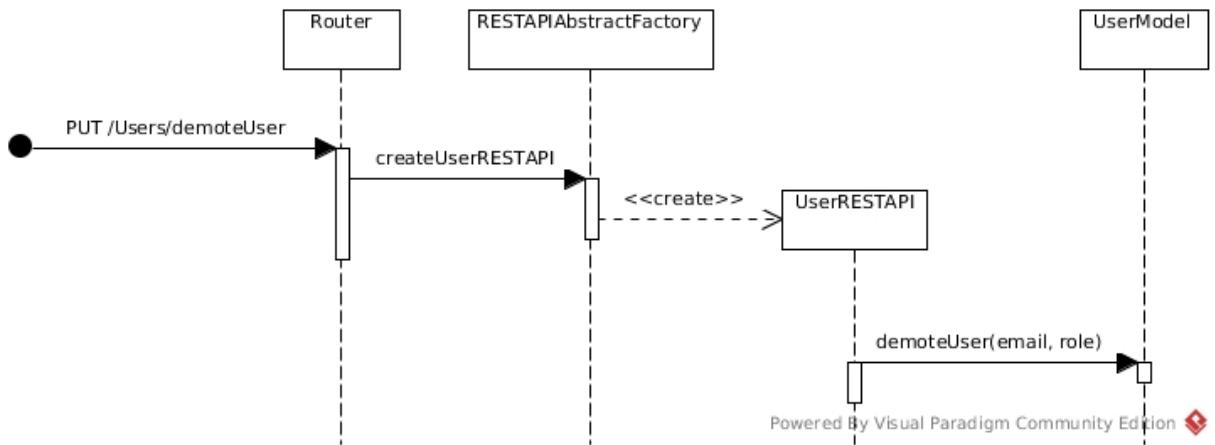


Figura 22: Diagramma di sequenza: PUT /Users/demoteUser

4.2.3.1.13 Richiesta POST /Users/sendInvite

- Descrizione:** Invio di una richiesta di registrazione all'azienda.
- Richiesta HTTP:** /Users/sendInvite
- Metodo:** POST
- Permessi:** Proprietario, Amministratore.
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Users/sendInvite. Creata la UserRESTAPI il metodo sendInvite() si occupa inviare una richiesta di registrazione azienda all'UserModel.

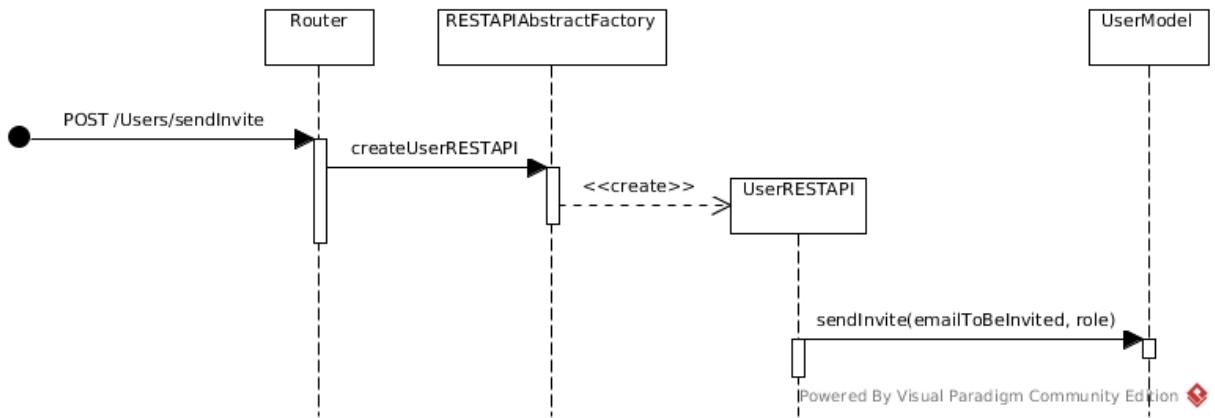


Figura 23: Diagramma di sequenza: POST /Users/sendInvite

Richieste Super Admin

4.2.3.2.14 Richiesta POST /SuperAdmin/login

- **Descrizione:** Esegue l'autenticazione e crea la sessione corrispondente al Super Amministratore.
- **Richiesta HTTP:** /SuperAdmin/login
- **Metodo:** POST
- **Permessi:** Super Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /SuperAdmin/login. Creata la SuperAdminRESTAPI il metodo login(user) si occupa inviare effettuare l'autenticazione e sessione per l'utente user nel SuperAdminModel.

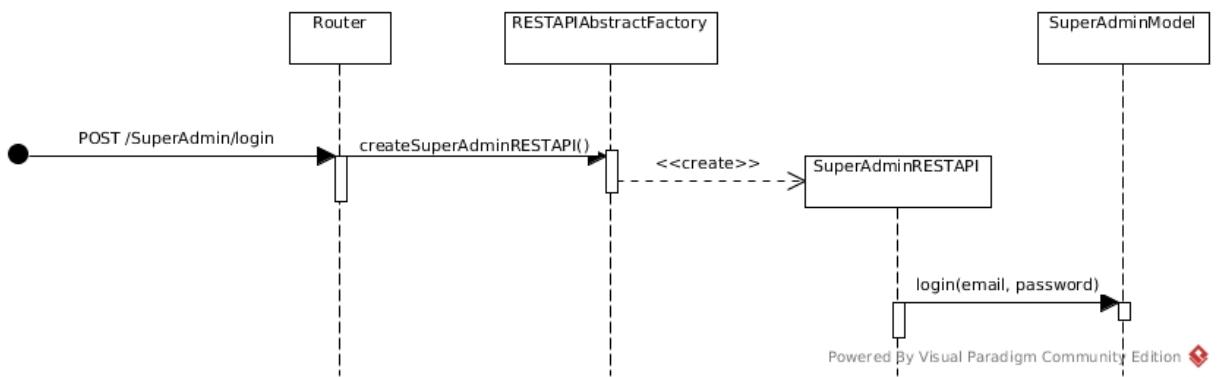
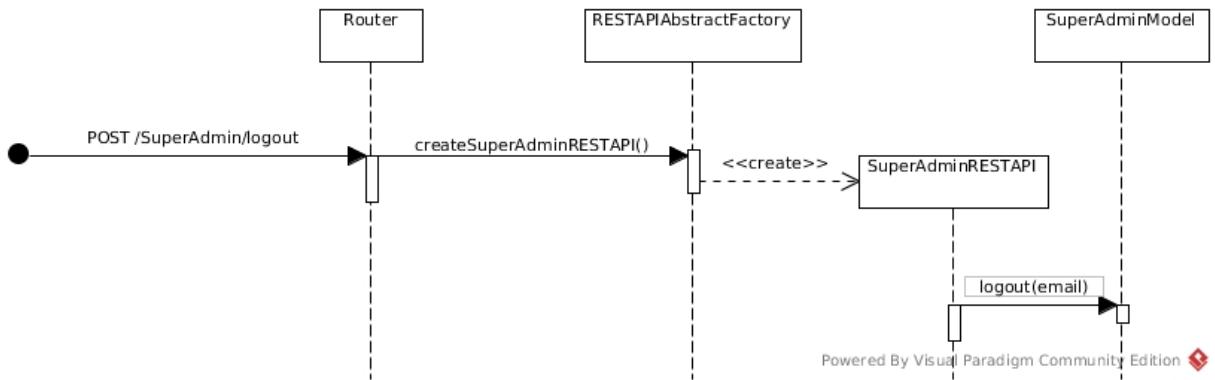


Figura 24: Diagramma di sequenza: POST /SuperAdmin/login

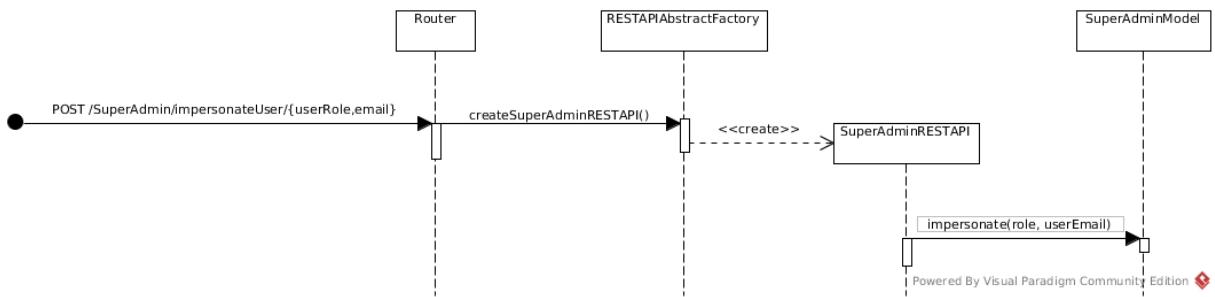
4.2.3.2.15 Richiesta POST /SuperAdmin/logout

- **Descrizione:** Elimina la sessione del Super Amministratore, corrispondente al logout.
- **Richiesta HTTP:** /SuperAdmin/logout
- **Metodo:** POST
- **Permessi:** Super Amministratore.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta POST per la risorsa /SuperAdmin/logout. Istanziate le SuperAdminRESTAPI la chiamata logout(user) effettua l'eliminazione della sessione nel SuperAdmin Model.


 Figura 25: Diagramma di sequenza: `POST /SuperAdmin/logout`

4.2.3.2.16 Richiesta `POST /SuperAdmin/impersonateUser/{userRole,email}`

- Descrizione:** Consente l'impersonificazione del Super Amministratore in un utente.
- Richiesta HTTP:** POST
- Metodo:** `/SuperAdmin/impersonateUser/{userRole,email}`
- Permessi:** Super Amministratore.
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa `/SuperAdmin/impersonateUser/{userRole,email}`. Creata la `SuperAdminRESTAPI` il metodo `impersonate(role,userEmail)` permette al Super Amministratore di impersonificare un utente nell'UserModel.


 Figura 26: Diagramma di sequenza: `POST /SuperAdmin/impersonateUser/{userRole,email}`

Richieste Companies

4.2.3.3.17 Richiesta GET /Companies

- Descrizione:** Mostra i dati di tutte le aziende.
- Richiesta HTTP:** /Companies
- Metodo:** GET

- **Permessi:** Super Amministratore.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta GET per la risorsa /Companies. Istanziate le CompanyRESTAPI la chiamata getAllCompanies() chiede i dati relativi a tutte le aziende nel CompanyModel. Il CompanyModel passerà i dati al CompanyRESTAPI che gli invia in formato JSON.

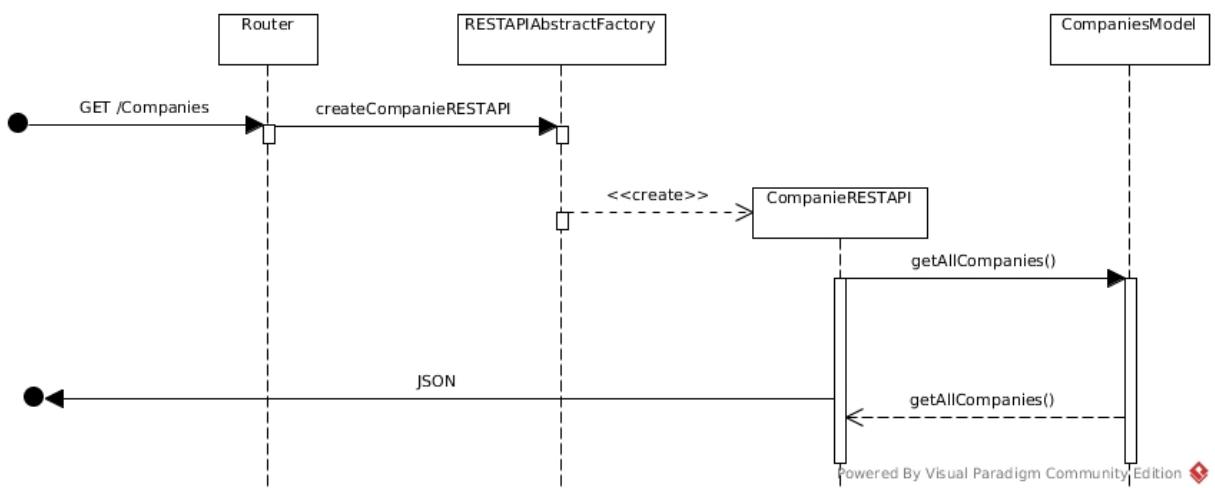


Figura 27: Diagramma di sequenza: GET /Companies

4.2.3.3.18 Richiesta POST /Companies

- **Descrizione:** Crea una azienda.
- **Richiesta HTTP:** /Companies
- **Metodo:** POST
- **Permessi:** Super Amministratore.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta POST per la risorsa /Companies. Istanziate le CompanyRESTAPI la chiamata createCompany(name) effettua la creazione di un'azienda nel CompanyModel.

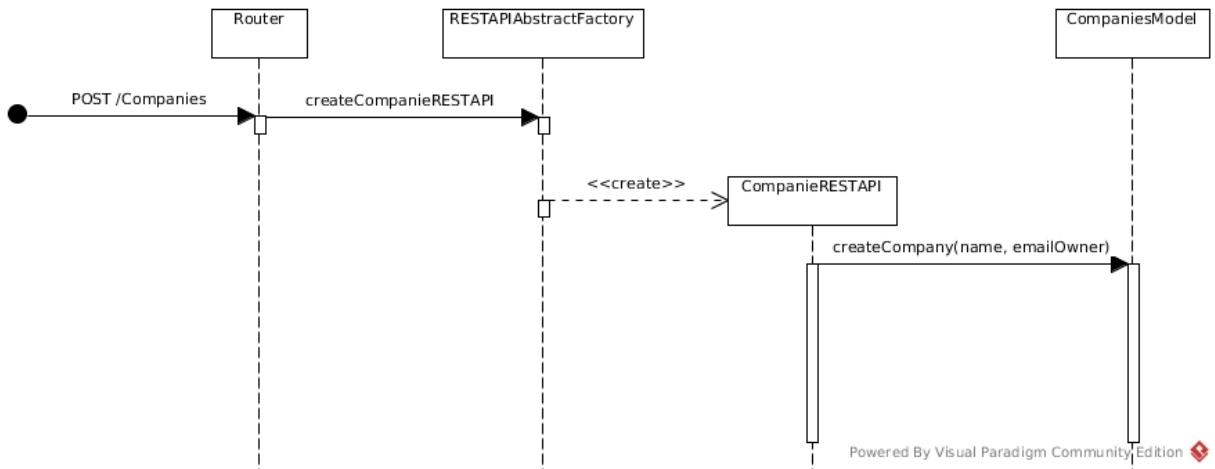


Figura 28: Diagramma di sequenza: POST /Companies

4.2.3.3.19 Richiesta GET /Companies/{companyName}

- **Descrizione:** Mostra i dati di una singola azienda identificandola col parametro companyName.
- **Richiesta HTTP:** /Companies/{companyName}
- **Metodo:** GET
- **Permessi:** Super Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Companies/{companyName}. Creata la CompanyRESTAPI il metodo getCompany(name) richiede i dati relativi ad una singola azienda al CompanyModel. I dati ricevuti sono inviati in formato JSON.

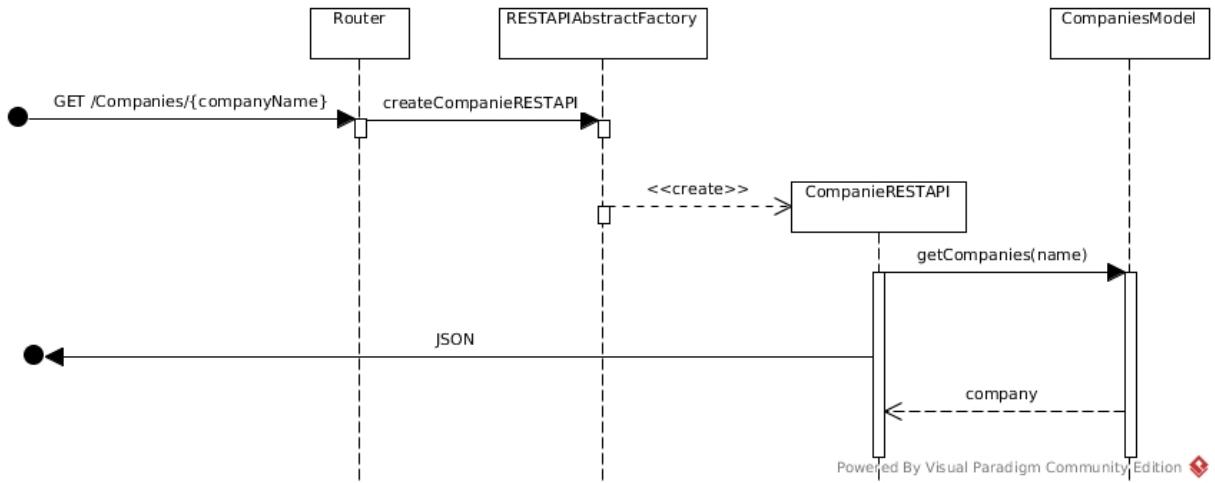


Figura 29: Diagramma di sequenza: GET /Companies/{companyName}

4.2.3.3.20 Richiesta PUT /Companies/{companyName}

- **Descrizione:** Modifica i dati di una azienda.
- **Richiesta HTTP:** /Companies/{companyName}
- **Metodo:** PUT
- **Permessi:** Super Amministratore.
- **Scenario:** Il seguente scenario mostra la gestione di una richiesta PUT per la risorsa /Companies/{companyName}. Istanziate le CompanyRESTAPI la chiamata setCompany(name) modifica i dati di dell'azienda "name" nel CompanyModel.

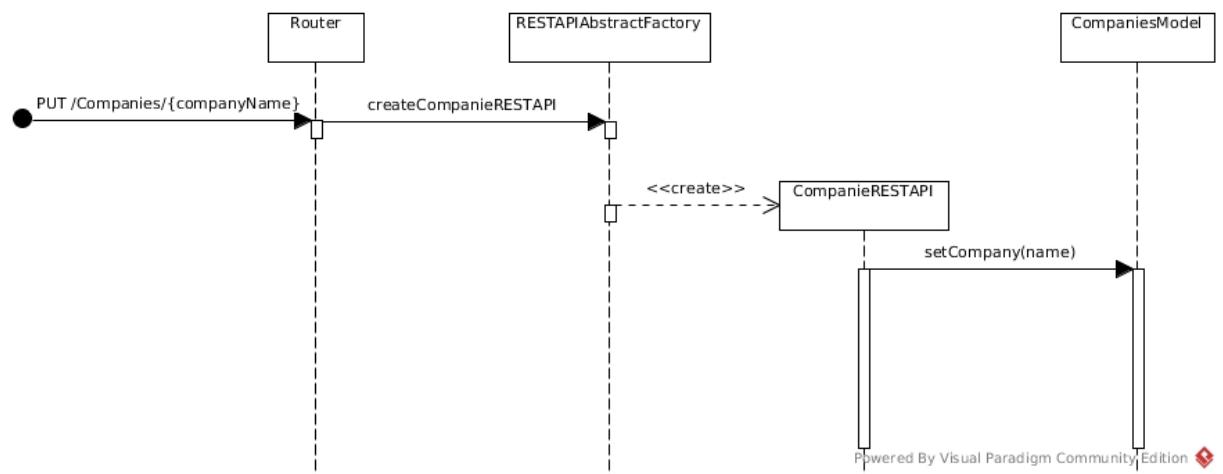


Figura 30: Diagramma di sequenza: PUT /Companies/{companyName}

4.2.3.3.21 Richiesta DELETE /Companies/deleteCompanies/{companyName}

- **Descrizione:** Elimina un'azienda.
- **Richiesta HTTP:** /Companies/deleteCompanies/{companyName}
- **Metodo:** DELETE
- **Permessi:** Super Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta DELETE per la risorsa /Companies/deleteCompanies/{companyName}. Creata la CompanyRESTAPI il metodo deleteCompany(name) richiede l'eliminazione di una singola azienda al CompanyModel.

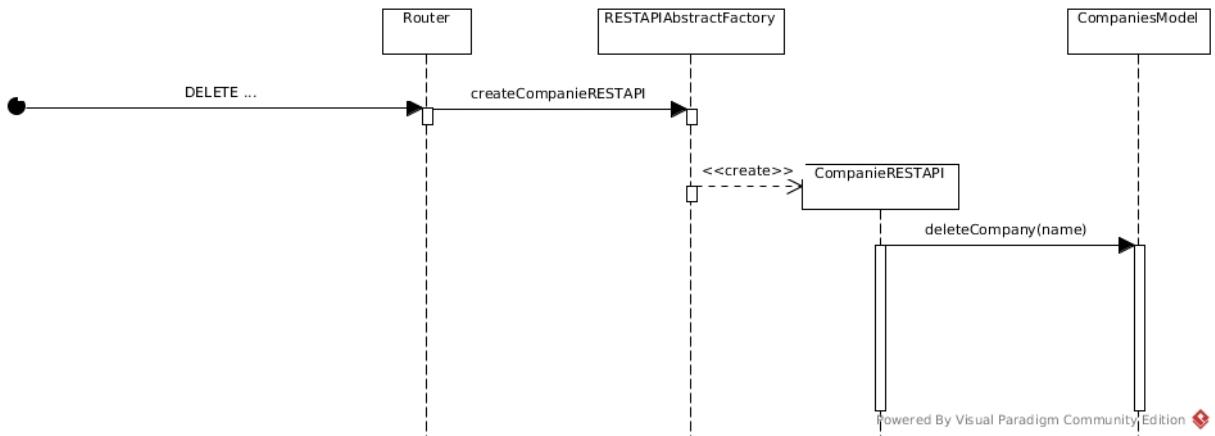


Figura 31: Diagramma di sequenza: DELETE /Companies/deleteCompanies/{companyName}

4.2.3.3.22 Richiesta GET /Companies/searchCompany/{value}

- Descrizione:** Ricerca di una azienda.
- Richiesta HTTP:** /Companies/searchCompany/{value}
- Metodo:** GET
- Permessi:** Super Amministratore.
- Scenario:** Il seguente scenario mostra la gestione di una richiesta GET per la risorsa /Companies/searchCompany/{value}. Istanziate le CompanyRESTAPI la chiamata searchCompany(name) effettua la ricerca di un'azienda nel CompanyModel. I dati ricevuti sono passati in formato JSON.

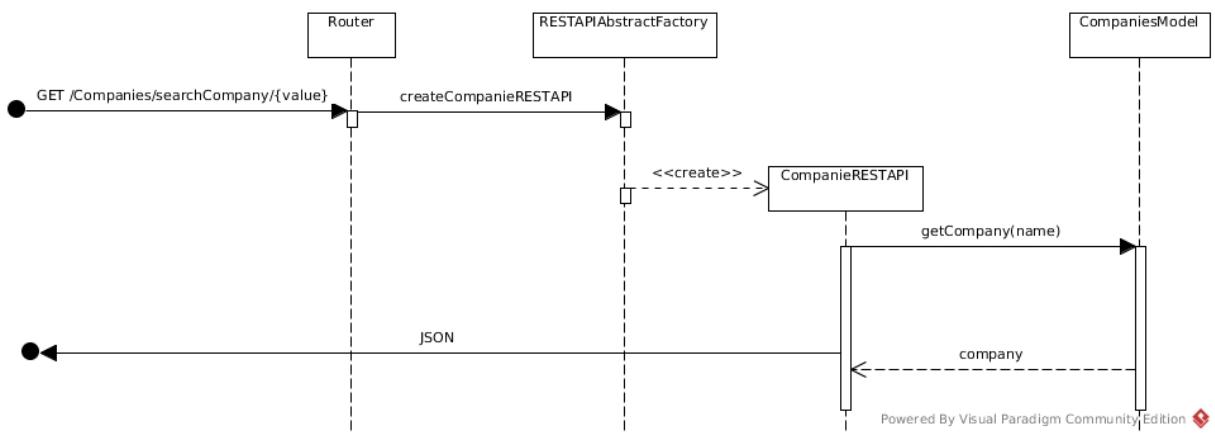


Figura 32: Diagramma di sequenza: GET /Companies/searchCompany/{value}

Richieste Databases

4.2.3.4.23 Richiesta POST /ExternalDatabases

- Descrizione:** Crea una risorsa rappresentante un database esterno mongoDB.
- Richiesta HTTP:** /ExternalDatabases
- Metodo:** POST
- Permessi:** Proprietario, Amministratore.
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /ExternalDatabaseModel. Creata la ExternalDatabaseRESTAPI il metodo addResource() si occupa di gestire la richiesta e creare la risorsa rappresentante un database esterno nel ExternalDatabaseModel.

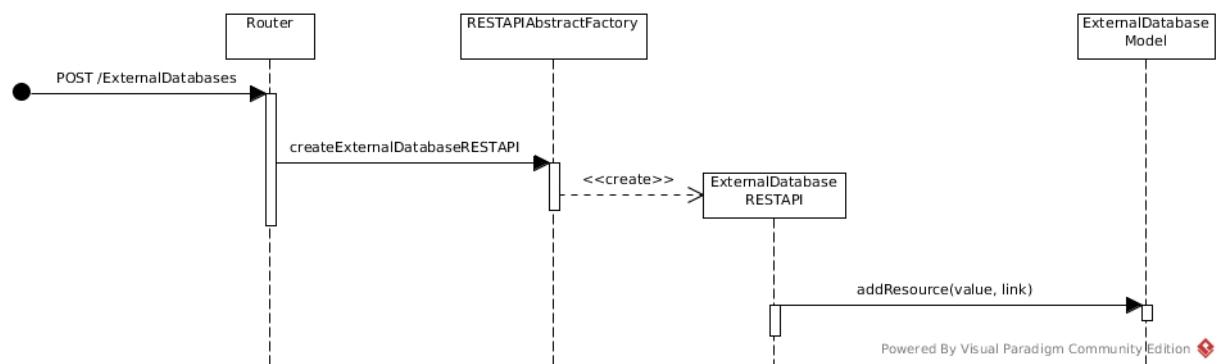


Figura 33: Diagramma di sequenza: POST /ExternalDatabases

4.2.3.4.24 Richiesta DELETE /ExternalDatabases/{id}

- Descrizione:** Elimina una risorsa rappresentante un database esterno mongoDB.
- Richiesta HTTP:** /ExternalDatabases/{id}
- Metodo:** DELETE
- Permessi:** Proprietario, Amministratore.
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta DELETE per la risorsa /ExternalDatabases/{id}. Creata la ExternalDatabaseRESTAPI il metodo deleteResource(id) si occupa di eliminare la risorsa rappresentante un database esterno nell'ExternalDatabaseModel.



Figura 34: Diagramma di sequenza: DELETE /ExternalDatabases/{id}

4.2.3.4.25 Richiesta /ExternalDatabases/query

- **Descrizione:** Mostra i dati dei database esterni appartenenti ad una certa azienda.
- **Richiesta HTTP:** /ExternalDatabases/query
- **Metodo:** GET
- **Permessi:** Proprietario, Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /ExternalDatabases/query. Creata la ExternalDatabaseRESTAPI il metodo visualizeExternalDatabases(query), in ExternalDatabaseModel, si occupa di visualizzare i dati di un database esterno.

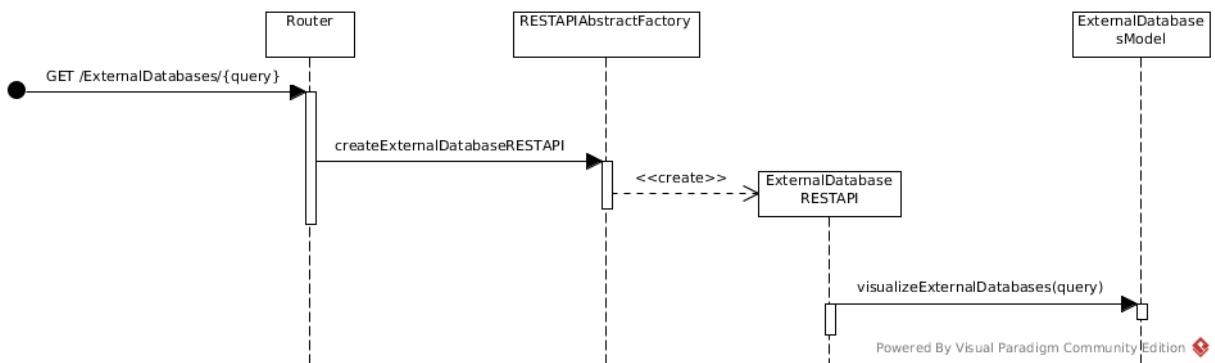


Figura 35: Diagramma di sequenza: GET /ExternalDatabases/query

4.2.3.4.26 Richiesta GET /ExternalDatabases/searchExternalDatabase/value, companyName

- **Descrizione:** Cerca un database esterno.
- **Richiesta HTTP:** /ExternalDatabases/searchExternalDatabase/value, companyName

- **Metodo:** GET
- **Permessi:** Proprietario, Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /ExternalDatabases/searchExternalDatabase/value, companyName. Creata la ExternalDatabaseRESTAPI il metodo searchDB(value, companyName) visualizza il database esterno, se trovato dall'ExternalDatabaseModel.

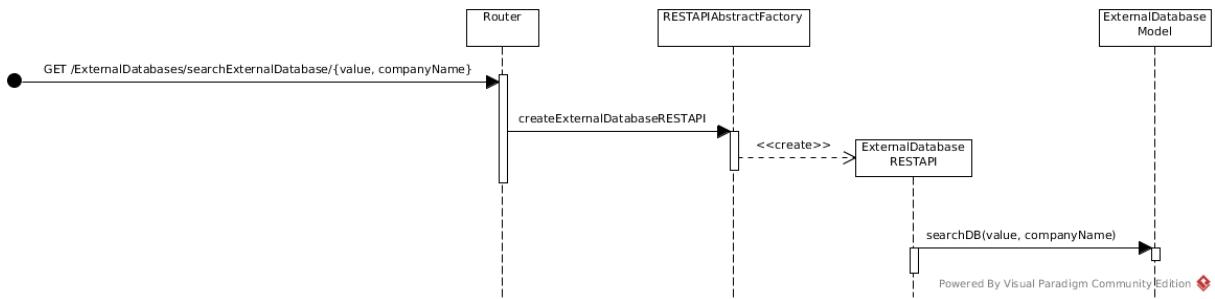


Figura 36: Diagramma di sequenza: GET /ExternalDatabases/searchExternalDatabase/value, companyName

4.2.3.4.27 Richiesta POST /ExternalDatabases/allowExternalDatabasesAccess

- **Descrizione:** Imposta positivamente i permessi di accesso ad un database esterno, da parte di un utente.
- **Richiesta HTTP:** /ExternalDatabases/allowExternalDatabasesAccess
- **Metodo:** POST
- **Permessi:** Proprietario, Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /ExternalDatabases/allowExternalDatabasesAccess. Creata la ExternalDatabaseRESTAPI il metodo allowAccess(email, id) si occupa di permettere l'accesso di un utente ad un database nell'ExternalDatabaseModel.

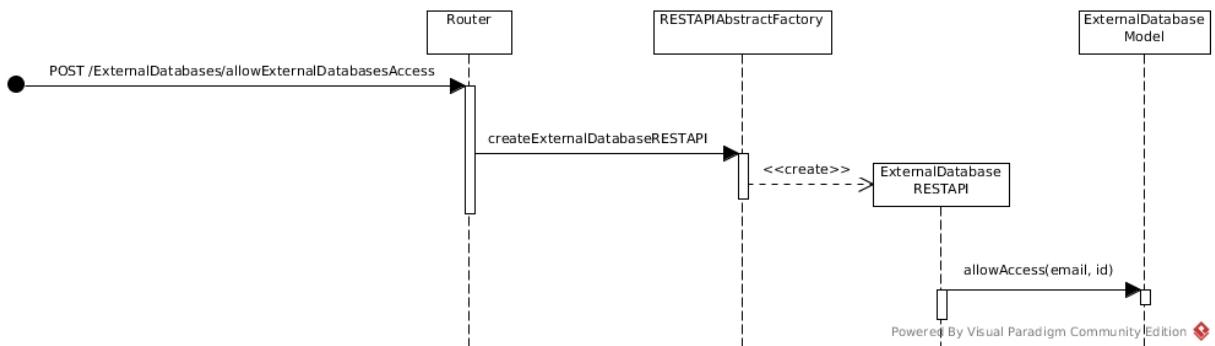


Figura 37: Diagramma di sequenza: POST /ExternalDatabases/allowExternalDatabasesAccess

4.2.3.4.28 Richiesta POST /ExternalDatabases/denyExternalDatabaseAccess

- **Descrizione:** Imposta negativamente i permessi di accesso ad un database esterno, da parte di un utente.
- **Richiesta HTTP:** /ExternalDatabases/{id}
- **Metodo:** POST
- **Permessi:** Proprietario, Amministratore.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /ExternalDatabases/denyExternalDatabaseAccess. Creata la ExternalDatabase: RESTAPI il metodo denyAccess(email, id) si occupa di impedire l'accesso di un utente ad un database nell'ExternalDatabaseModel.

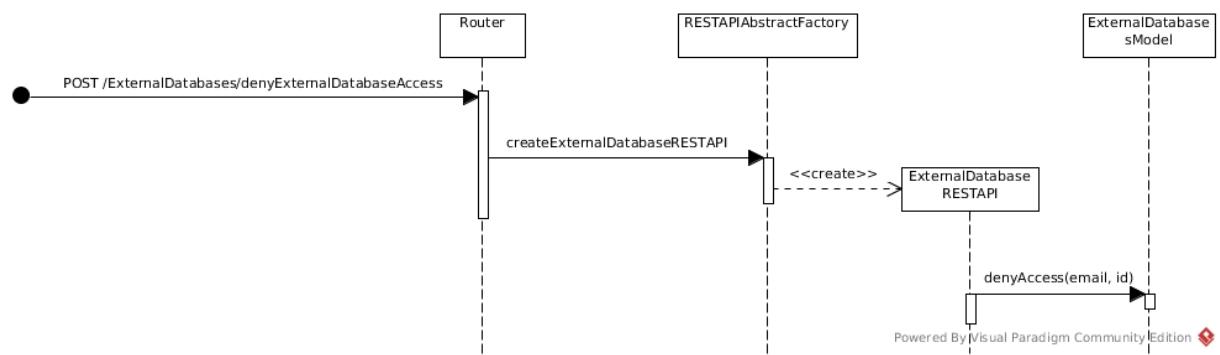


Figura 38: Diagramma di sequenza: POST /ExternalDatabases/denyExternalDatabaseAccess

Richieste Collections

4.2.3.5.29 Richiesta GET /Collections

- **Descrizione:** Mostra la lista delle Collection.
- **Richiesta HTTP:** /Collections
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Collections. Creata la CollectionRESTAPI il metodo getCollection() si occupa di richiedere la lista delle Collection nel CollectionModel e restituire i dati in formato JSON.

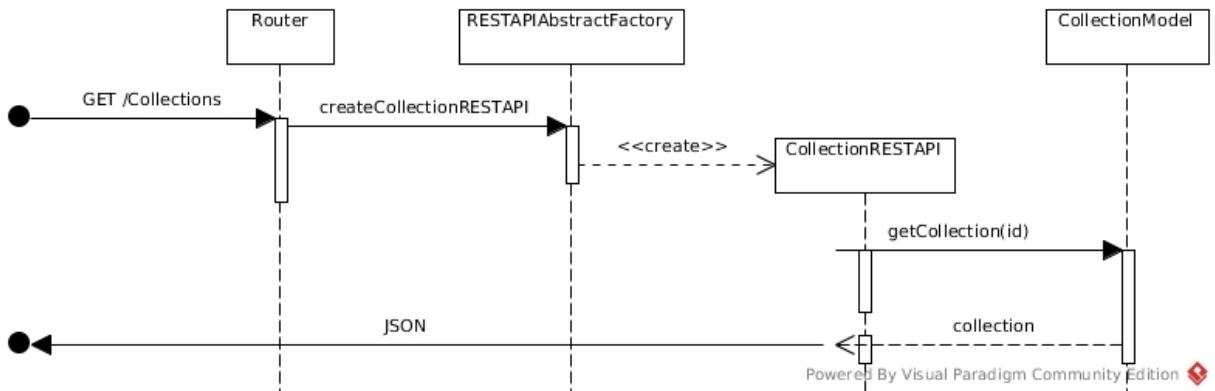


Figura 39: Diagramma di sequenza: GET /Collections

4.2.3.5.30 Richiesta POST /Collections

- **Descrizione:** Creazione di una Collection
- **Richiesta HTTP:** /Collections
- **Metodo:** POST
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Collections. Creata la CollectionRESTAPI il metodo createCollection() si creare una Collection nel CollectionModel.

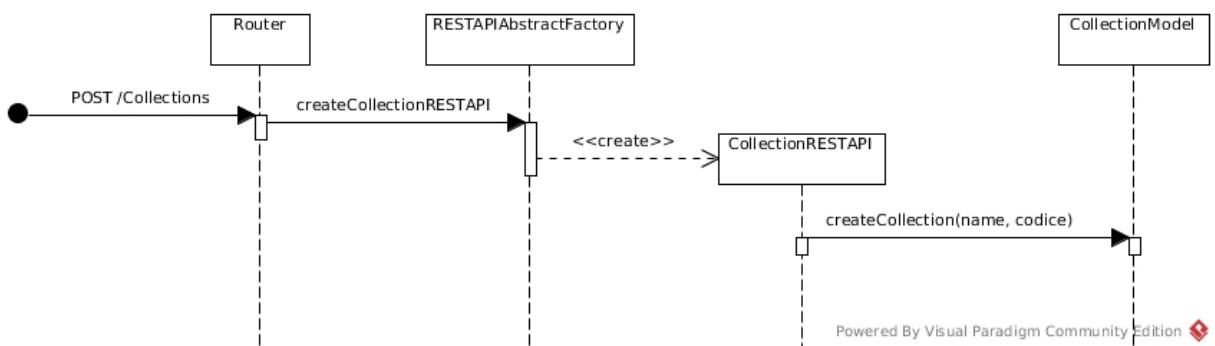


Figura 40: Diagramma di sequenza: POST /Collections

4.2.3.5.31 Richiesta PUT /Collections/{id}

- **Descrizione:** Modifica di una Collection
- **Richiesta HTTP:** /Collections/{id}
- **Metodo:** PUT
- **Permessi:** Utente autenticato

- Scenario:** Il seguente scenario mostra la gestione di una richiesta PUT per la risorsa /Collections/{id}. Istanziate le CollectionRESTAPI la chiamata setCompany(name) permette la modifica di una collection nel CollectionModel.

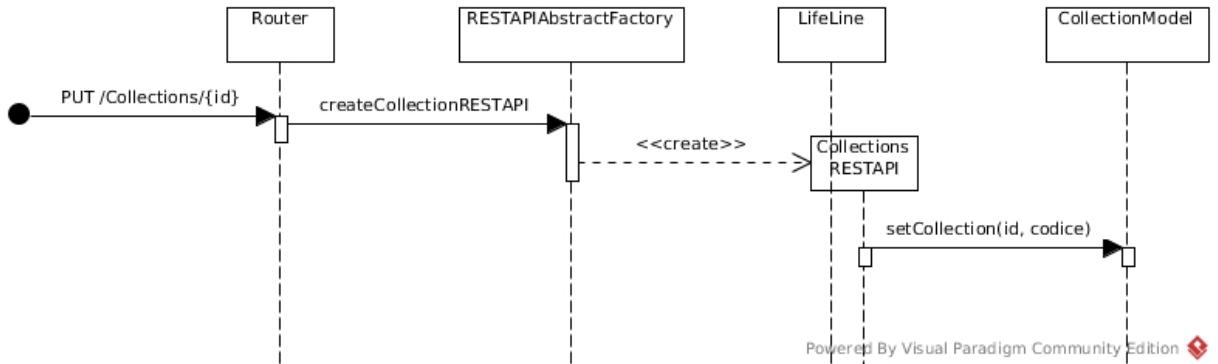


Figura 41: Diagramma di sequenza: PUT /Collections/{id}

4.2.3.5.32 Richiesta DELETE /Collections/{id}

- Descrizione:** Eliminazione di una Collection mediante il suo identificativo.
- Richiesta HTTP:** /Collections/{id}
- Metodo:** DELETE
- Permessi:** Utente autenticato.
- Scenario:** Il seguente scenario mostra la gestione di una richiesta DELETE per la risorsa /Collections/{id}. Istanziate le CollectionRESTAPI la chiamata deleteCompany(name) permette l'eliminazione di una collection nel CollectionModel.

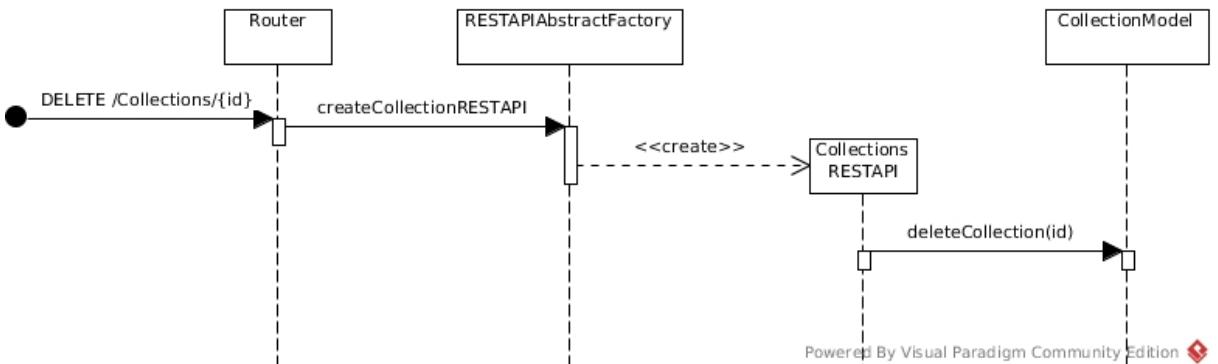


Figura 42: Diagramma di sequenza: DELETE /Collections/{id}

4.2.3.5.33 Richiesta GET /Collections/retrieveCollectionDSLIS/{id}

- Descrizione:** Ritorna il codice della Collection

- **Richiesta HTTP:** /Collections/retrieveCollectionDSLIS/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Collections/retrieveCollectionDSLIS/{id}. Creata la CollectionRESTAPI il metodo getCollection() richiede nel CollectionModel il codice della Collection che viene restituita in formato JSON.

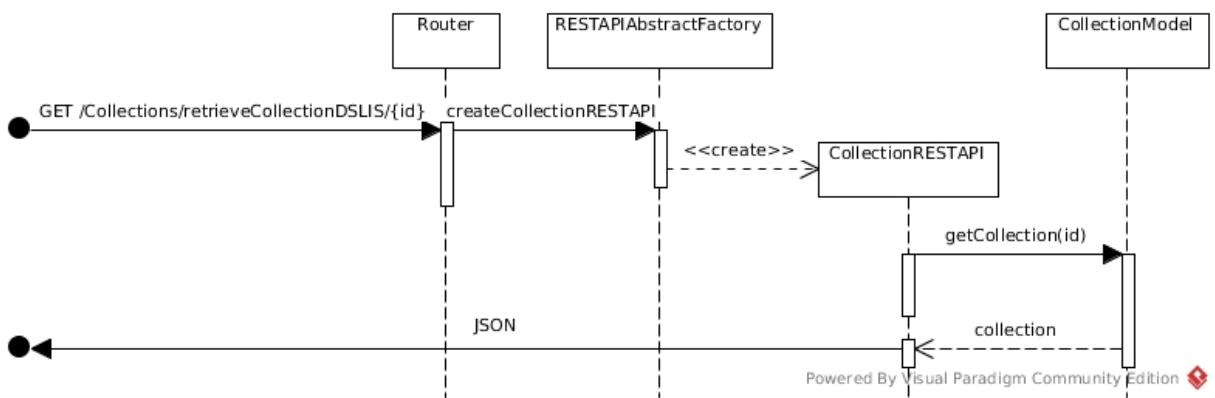


Figura 43: Diagramma di sequenza: GET /Collections/retrieveCollectionDSLIS/{id}

4.2.3.5.34 Richiesta GET /Collections/searchCollections/{id}

- **Descrizione:** Ricerca di una Collection tramite il suo identificativo.
- **Richiesta HTTP:** /Collections/searchCollections/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Collections/searchCollections/{id}. Creata la CollectionRESTAPI il metodo searchCollection(id) effettua una ricerca nel CollectionModel. I dati della Collection se trovati sono restituiti in formato JSON.

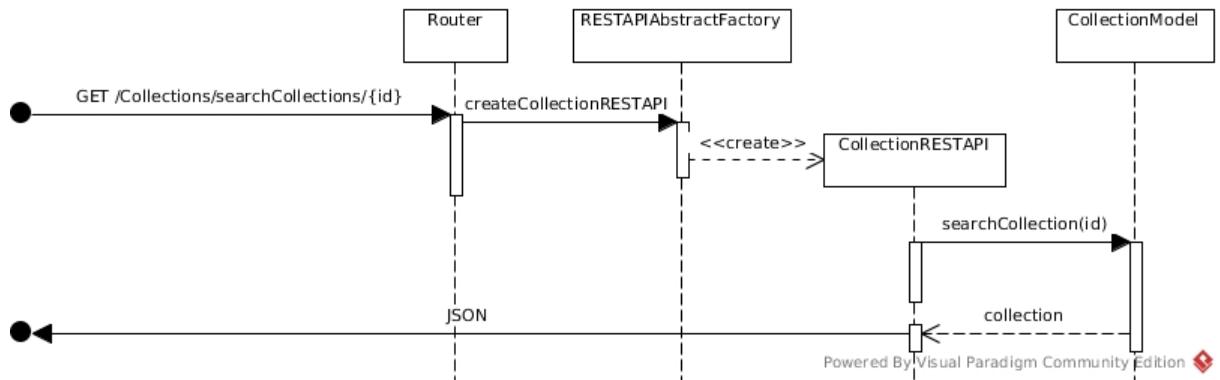


Figura 44: Diagramma di sequenza: GET /Collections/searchCollections/{id}

4.2.3.5.35 Richiesta GET /Collections/ExecuteCollection/{id}

- Descrizione:** Ritorna la lista dei Document appartenenti ad una specifica Collection, oltre ad altri campi.
- Richiesta HTTP:** /Collections/ExecuteCollection/{id}
- Metodo:** GET
- Permessi:** Utente autenticato.
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Collections/ExecuteCollection/{id}. Creata la CollectionRESTAPI il metodo docCollection(id) richiede al CollectionModel la lista dei Document della Collection "id". I dati sono restituiti in formato JSON.

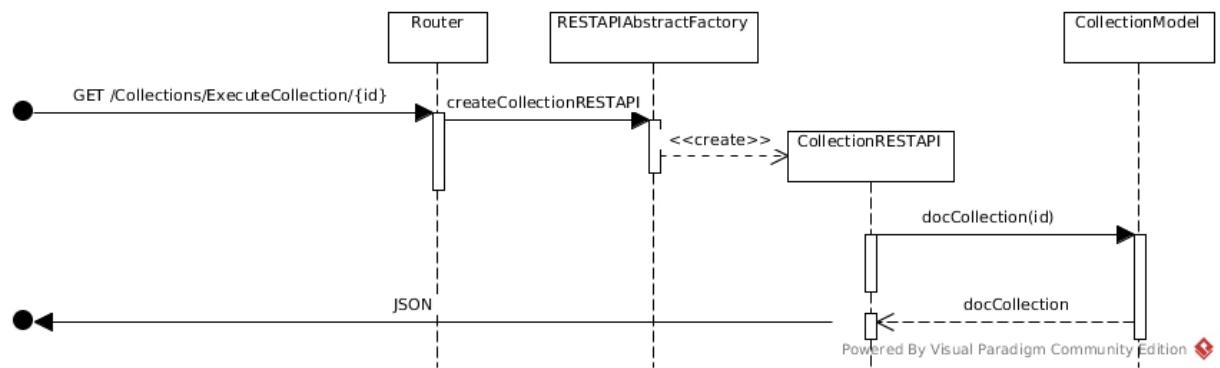


Figura 45: Diagramma di sequenza: GET /Collections/ExecuteCollection/{id}

4.2.3.5.36 Richiesta GET /Collections/SendEmail{id}

- Descrizione:** Permette di inviare tramite email la struttura dei dati della Collection selezionata in un formato a scelta tra JSON e CSV.

- **Richiesta HTTP:** /Collections/SendEmail{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Collections/SendEmail{id}. Creata la CollectionRESTAPI il metodo sendEmailCollection(id,type) richiede al CollectionModel di inviare tramite e mail la struttura dati di una Collection. I dati sono restituiti in formato JSON o CSV in base al parametro "type".

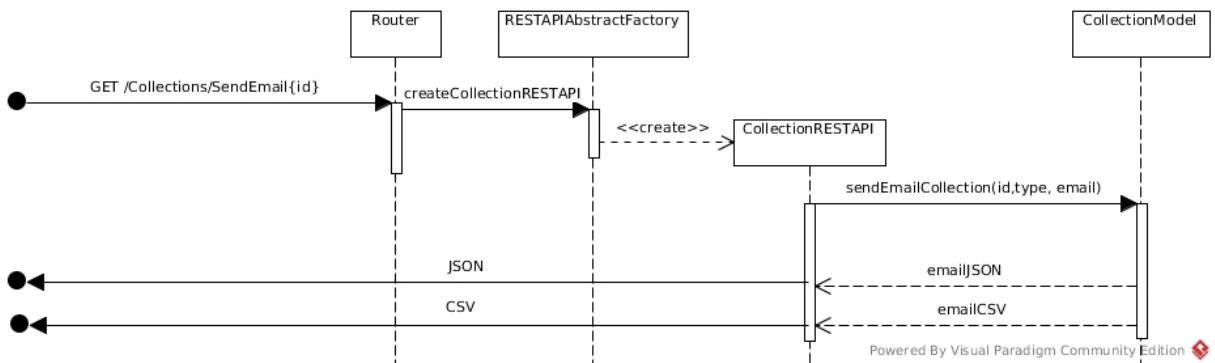


Figura 46: Diagramma di sequenza: GET /Collections/SendEmail{id}

4.2.3.5.37 Richiesta GET /Collections/Export/{id}

- **Descrizione:** Permette di esportare la struttura dei dati della Collection selezionata in un formato a scelta tra JSON e CSV.
- **Richiesta HTTP:** /Collections/Export/{id}
- **Metodo:**
- **Permessi:** Utente autenticato.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Collections/Export/{id}. Creata la CollectionRESTAPI il metodo exportCollection(id,type) richiede al CollectionModel di esportare la struttura dati di una Collection. I dati sono restituiti in formato JSON o CSV in base al parametro "type".

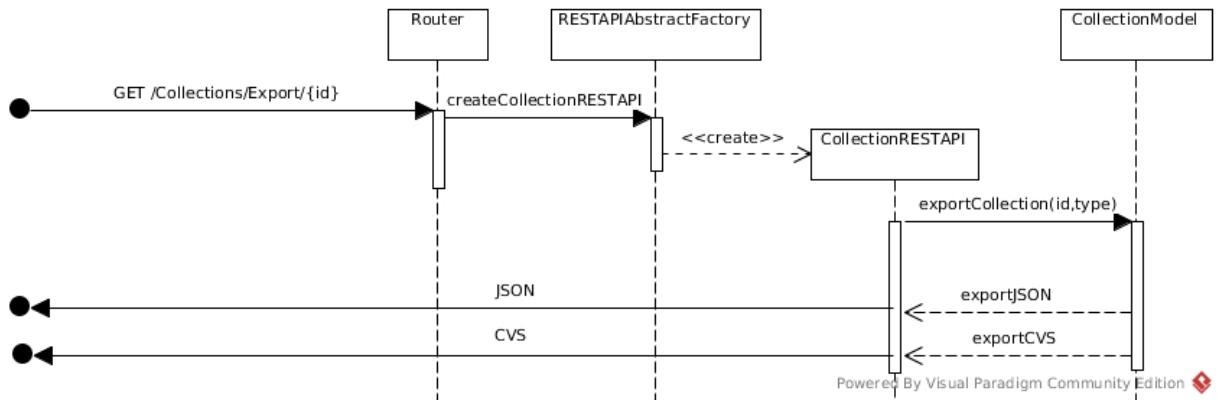


Figura 47: Diagramma di sequenza: GET /Collections/Export/{id}

4.2.3.5.38 Richiesta GET /Collections/exportCollectionDSLIS

- Descrizione:** Permette di esportare un DSLIS rappresentante una Collection.
- Richiesta HTTP:** /Collections/exportCollectionDSLIS
- Metodo:** GET
- Permessi:** Proprietario, Amministratore, Membro
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Collections/exportCollectionDSLIS. Creata la CollectionRESTAPI il metodo exportCollectionDSLIS(id) richiede al CollectionModel di esportare un DSLIS rappresentante una Collection. I dati sono restituiti in formato JSON.

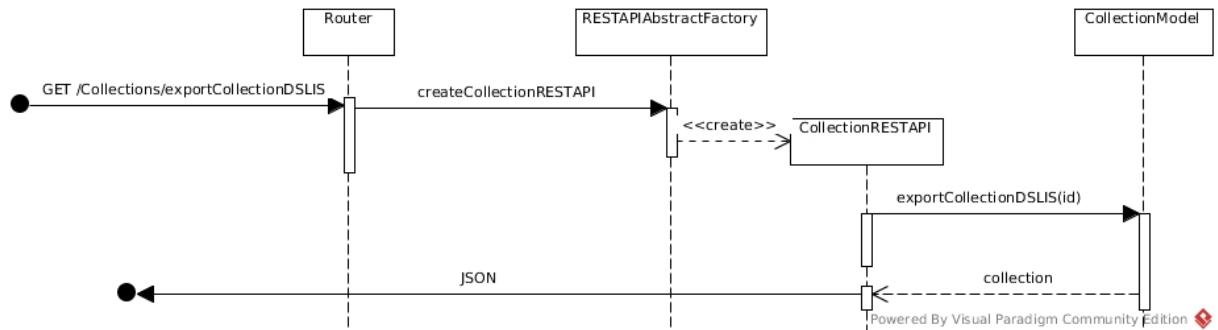


Figura 48: Diagramma di sequenza: GET /Collections/exportCollectionDSLIS

4.2.3.5.39 Richiesta POST /Collections/importCollectionDSLIS

- Descrizione:** Permette di importare un DSLIS rappresentante una Collection.
- Richiesta HTTP:** /Collections/importCollectionDSLIS
- Metodo:** POST

- **Permessi:** Proprietario, Amministratore, Membro
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Collections/importCollectionDSLIS. Creata la CollectionRESTAPI il metodo importCollectionDSLIS(id) richiede al CollectionModel di importare un DSLIS rappresentante una Collection.

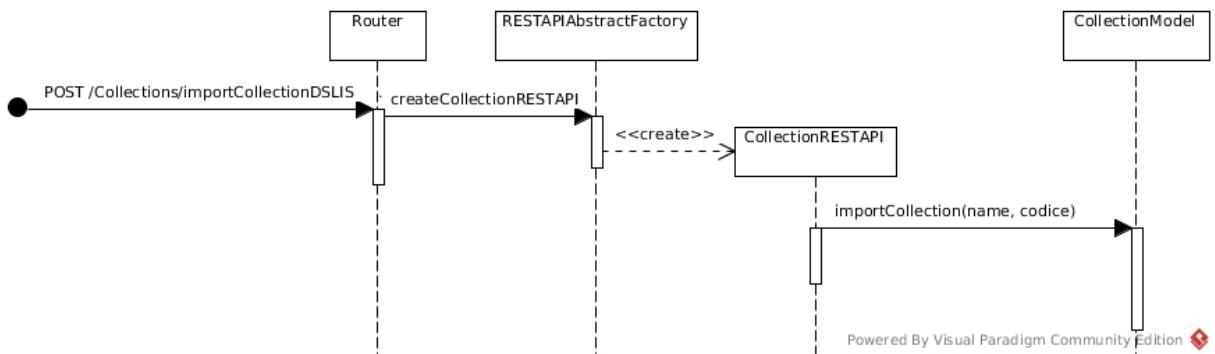


Figura 49: Diagramma di sequenza: POST /Collections/importCollectionDSLIS

Richieste Documents

4.2.3.6.40 Richiesta GET /Documents

- **Descrizione:** Mostra la lista dei Documenti.
- **Richiesta HTTP:** /Documents
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents. Creata la DocumentRESTAPI il metodo importDocumentDSLIS(id) richiede al DocumentModel di ottenere un DSLIS rappresentante un Document.

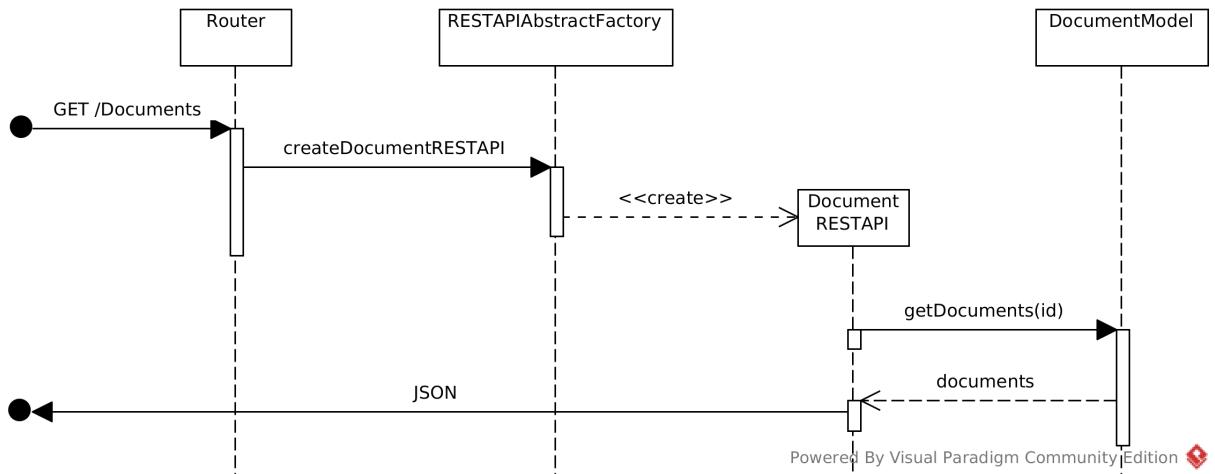


Figura 50: Diagramma di sequenza: GET /Documents

4.2.3.6.41 Richiesta POST /Documents

- **Descrizione:** Creazione di un Document.
- **Richiesta HTTP:** /Documents
- **Metodo:** POST
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Documents. Creata la DocumentRESTAPI il metodo addDocuments(name, codice) richiede al DocumentModel di creare un nuovo DSLIS rappresentante un Document.

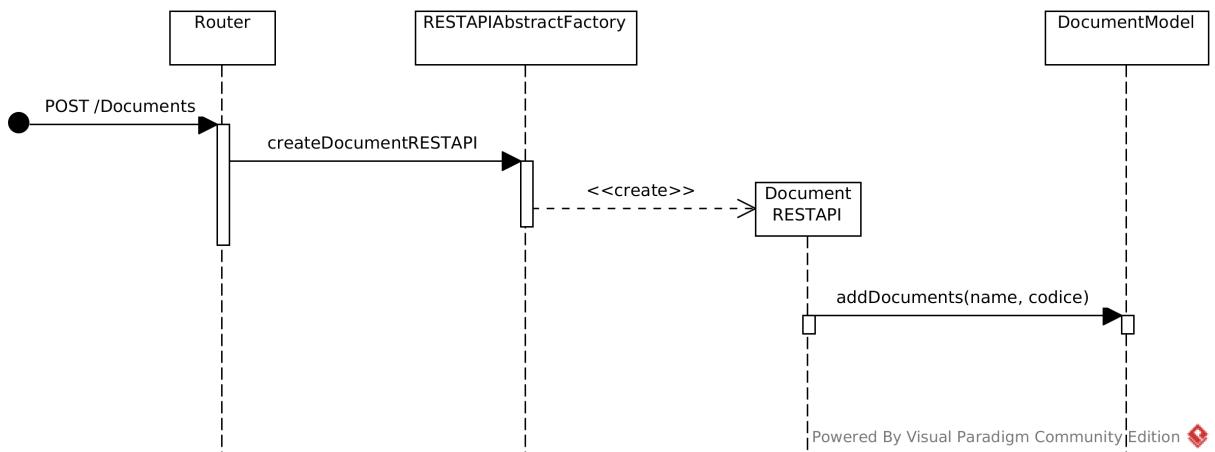


Figura 51: Diagramma di sequenza: POST /Documents

4.2.3.6.42 Richiesta PUT /Documents/{id}

- Descrizione:** Modifica di un Document
- Richiesta HTTP:** /Documents/{id}
- Metodo:** PUT
- Permessi:** Utente autenticato
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta PUT per la risorsa /Documents/{id}. Creata la DocumentRESTAPI il metodo setDocuments(id, codice) richiede al DocumentModel di modificare un DSLIS rappresentante un Document.

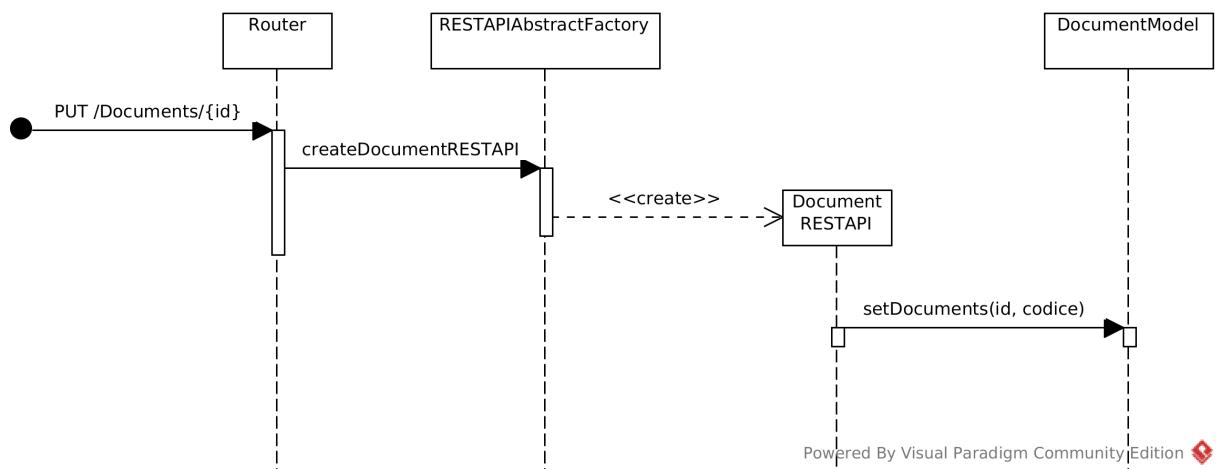
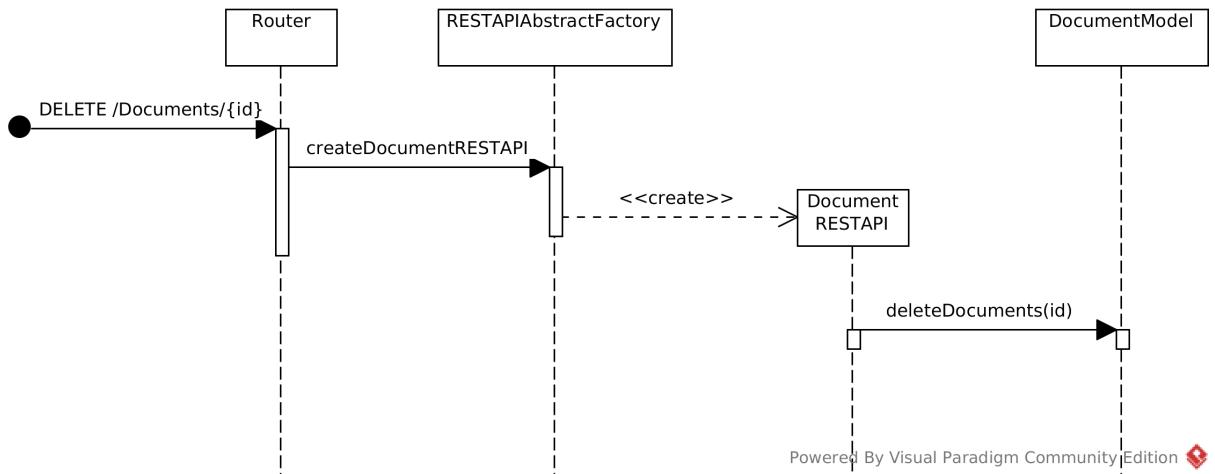


Figura 52: Diagramma di sequenza: PUT /Documents/{id}

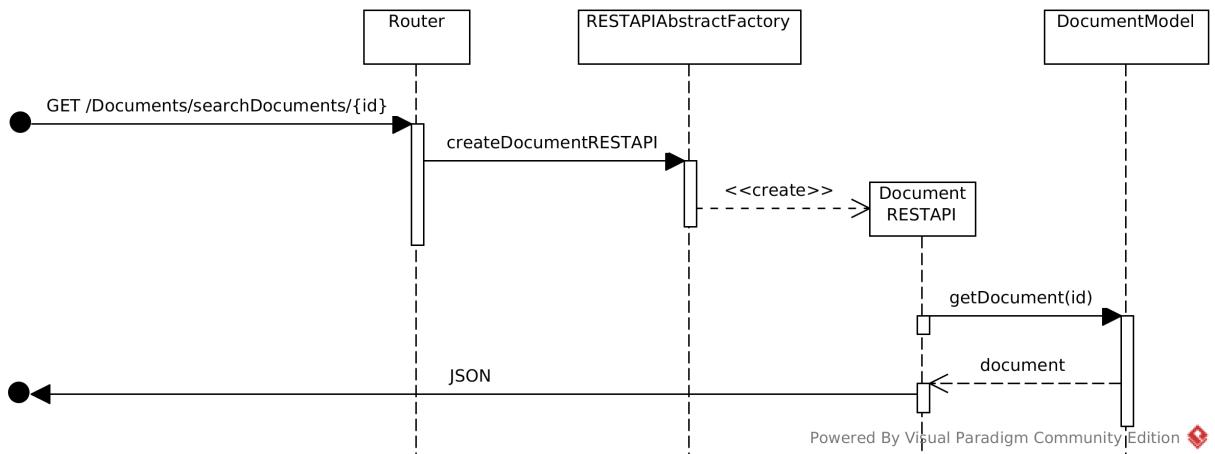
4.2.3.6.43 Richiesta DELETE /Documents/{id}

- Descrizione:** Eliminazione di un Document tramite identificativo.
- Richiesta HTTP:** /Documents/{id}
- Metodo:** DELETE
- Permessi:** Utente autenticato
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta DELETE per la risorsa /Documents/{id}. Creata la DocumentRESTAPI il metodo deleteDocuments(id) richiede al DocumentModel di eliminare un DSLIS rappresentante un Document.


 Figura 53: Diagramma di sequenza: `DELETE /Documents/{id}`

4.2.3.6.44 Richiesta GET /Documents/searchDocuments/{id}

- **Descrizione:** Ricerca di un Document.
- **Richiesta HTTP:** `/Documents/searchDocuments/{id}`
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa `/Documents/searchDocuments/{id}`. Creato la DocumentRESTAPI il metodo `getDocument(id)` richiede al DocumentModel di ottenere un DSLIS rappresentante un Document, se presente.


 Figura 54: Diagramma di sequenza: `GET /Documents/searchDocuments/{id}`

4.2.3.6.45 Richiesta GET /Documents/ExecuteDocument/{id}

- **Descrizione:** Ritorna la lista degli attributi del Document.
- **Richiesta HTTP:** /Documents/ExecuteDocument/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents/ExecuteDocument/{id}. Creata la DocumentRESTAPI il metodo getAttributeDocument(id) richiede al DocumentModel di eseguire un DSLIS rappresentante un Document.

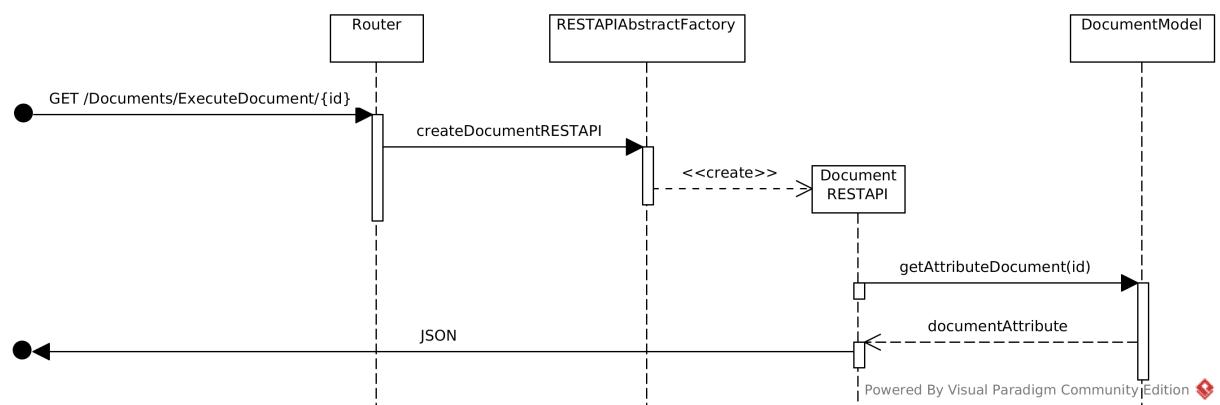


Figura 55: Diagramma di sequenza: GET /Documents/ExecuteDocument/{id}

4.2.3.6.46 Richiesta GET /Documents/ExecuteDocument/{CollectionName}/{id}

- **Descrizione:** Ritorna la lista degli attributi del Document appartenente alla CollectionName.
- **Richiesta HTTP:** /Documents/ExecuteDocument/{CollectionName}/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents/Execute/{CollectionName}/{id}. Creata la DocumentRE: STAPI il metodo getAttributeListDocumentDocumentCollection(name, id) richiede al DocumentModel di eseguire un DSLIS rappresentante un Document di una Collection.

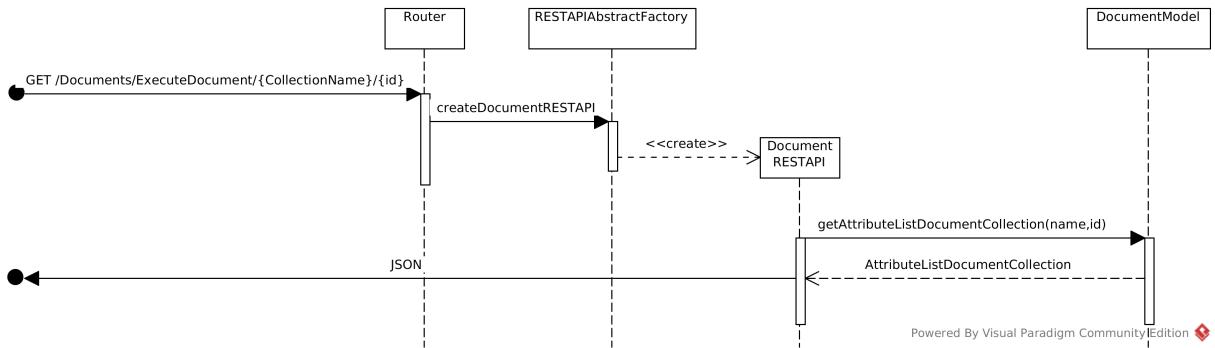


Figura 56: Diagramma di sequenza: GET /Documents/ExecuteDocument/{CollectionName}/{id}

4.2.3.6.47 Richiesta GET /Documents/SendEmail{id}

- Descrizione:** Permette di inviare tramite email la struttura dei dati del Document selezionato in un formato a scelta tra JSON e CSV.
- Richiesta HTTP:** /Documents/SendEmail{id}
- Metodo:** GET
- Permessi:** Utente autenticato.
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents/SendEmail{id}. Creata la DocumentRESTAPI il metodo sendEmailDocument(id, type, email) richiede al DocumentModel di inviare una email contenente un DSLIS rappresentante un Document.

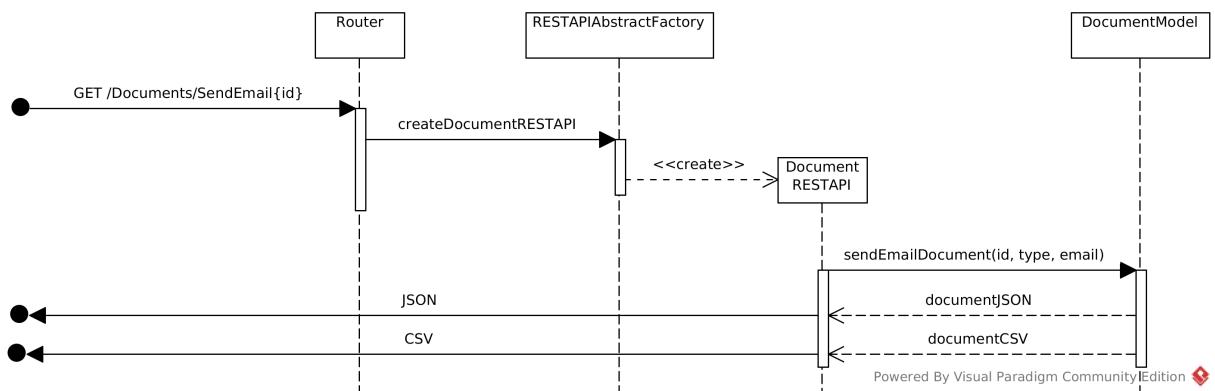


Figura 57: Diagramma di sequenza: GET /Documents/SendEmail{id}

4.2.3.6.48 Richiesta GET /Documents/Export/{id}

- Descrizione:** Permette di esportare la struttura dei dati del Document selezionato in un formato a scelta tra JSON e CVS.

- **Richiesta HTTP:** /Documents/Export/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents/Export/{id}. Creata la DocumentRESTAPI il metodo exportDocument(id, type) richiede al DocumentModel di esportare il risultato di un DSLIS rappresentante un Document.

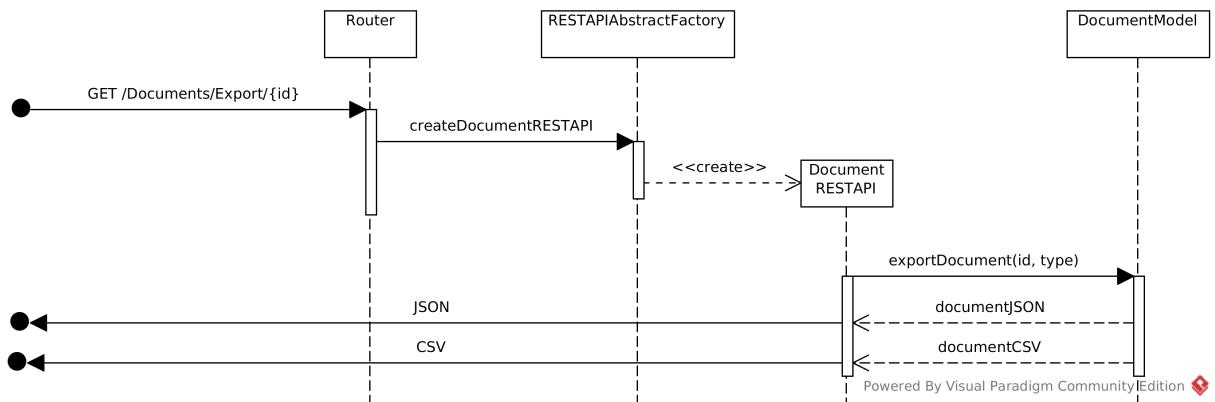


Figura 58: Diagramma di sequenza: GET /Documents/Export/{id}

4.2.3.6.49 Richiesta GET /Documents/retrieveDocumentDSLIS

- **Descrizione:** Ritorna il codice del Document.
- **Richiesta HTTP:** /Documents/retrieveDocumentDSLIS
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents/retrieveDocumentDSLIS. Creata la DocumentRESTAPI il metodo DocumentDSLIS(id) richiede al DocumentModel di ottenere il codice di un DSLIS rappresentante un Document.

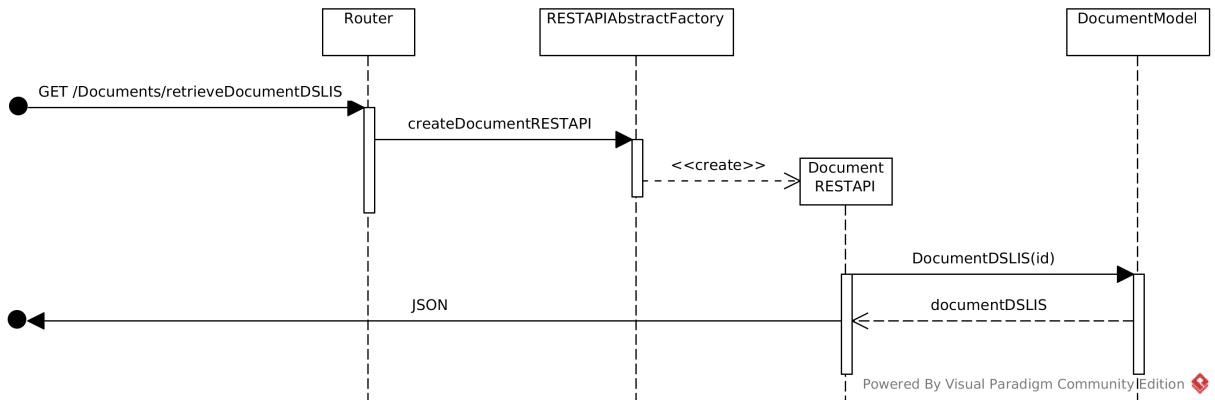


Figura 59: Diagramma di sequenza: GET /Documents/retrieveDocumentDSLIS

4.2.3.6.50 Richiesta GET /Documents/exportDocumentDSLIS

- Descrizione:** Permetto di esportare un DSLIS rappresentante un Document.
- Richiesta HTTP:** /Documents/exportDocumentDSLIS
- Metodo:** GET
- Permessi:** Proprietario, Amministratore, Membro.
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents/exportDocumentDSLIS. Creata la DocumentRESTAPI il metodo exportDocumentDSLIS(id, type) richiede al DocumentModel di esportare il codice di un DSLIS rappresentante un Document.

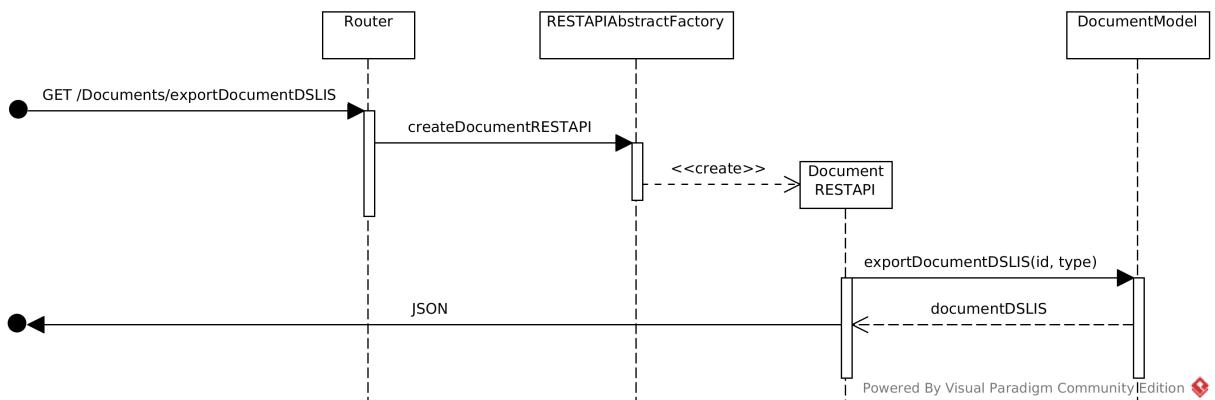


Figura 60: Diagramma di sequenza: GET /Documents/exportDocumentDSLIS

4.2.3.6.51 Richiesta POST /Documents/importDocumentDSLIS

- Descrizione:** Permette di importare un DSLIS rappresentante un Document.
- Richiesta HTTP:** /Documents/importDocumentDSLIS

- **Metodo:** POST
- **Permessi:** Proprietario, Amministratore, Membro.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Documents/importCollectionDSLIS. Creata la DocumentRESTAPI il metodo importDocumentDSLIS(name, codice) richiede al DocumentModel di importare un DSLIS rappresentante un Document.

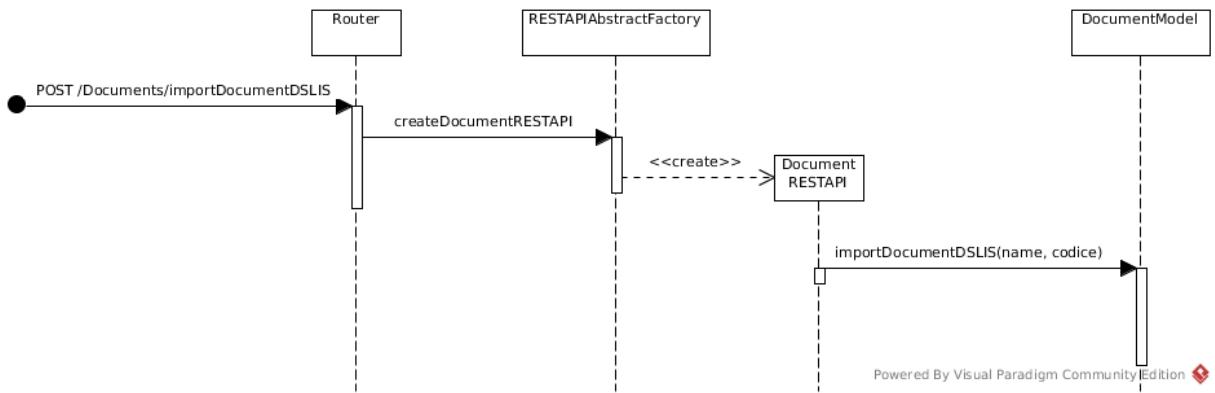


Figura 61: Diagramma di sequenza: POST /Documents/importDocumentDSLIS

Richieste Dashboards

4.2.3.7.52 Richiesta GET /Dashboards

- **Descrizione:** Mostra la lista della Dashboard.
- **Richiesta HTTP:** /Dashboards
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Dashboards. Creata la DashboardRESTAPI il metodo getDashboardDSLIS(id) richiede al DashboardModel di ottenere un DSLIS rappresentante una Dashboard.

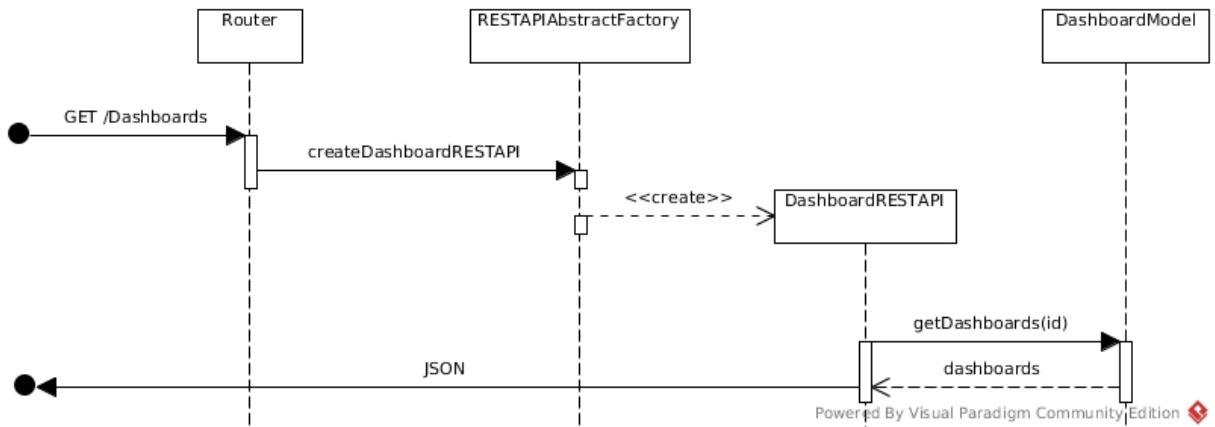


Figura 62: Diagramma di sequenza: GET /Dashboards

4.2.3.7.53 Richiesta POST /Dashboards

- **Descrizione:** Creazione di una Dashboards.
- **Richiesta HTTP:** /Dashboards
- **Metodo:** POST
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Dashboards. Creato la DashboardRESTAPI il metodo createDashboard(name) richiede al DashboardModel di creare un nuovo DSLIS rappresentante una Dashboard.

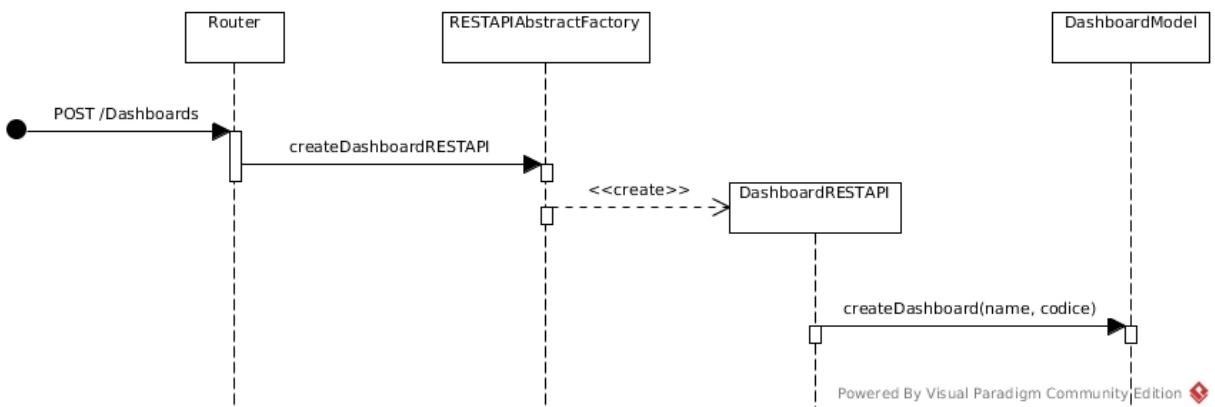


Figura 63: Diagramma di sequenza: POST /Dashboards

4.2.3.7.54 Richiesta PUT /Dashboards/{id}

- **Descrizione:** Modifica di una Dashboards.

- **Richiesta HTTP:** /Dashboards/{id}
- **Metodo:** PUT
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta PUT per la risorsa /Dashboards. Creata la DashboardRESTAPI il metodo setDashboard(id) richiede al DashboardModel di modificare un DSLIS rappresentante una Dashboard.

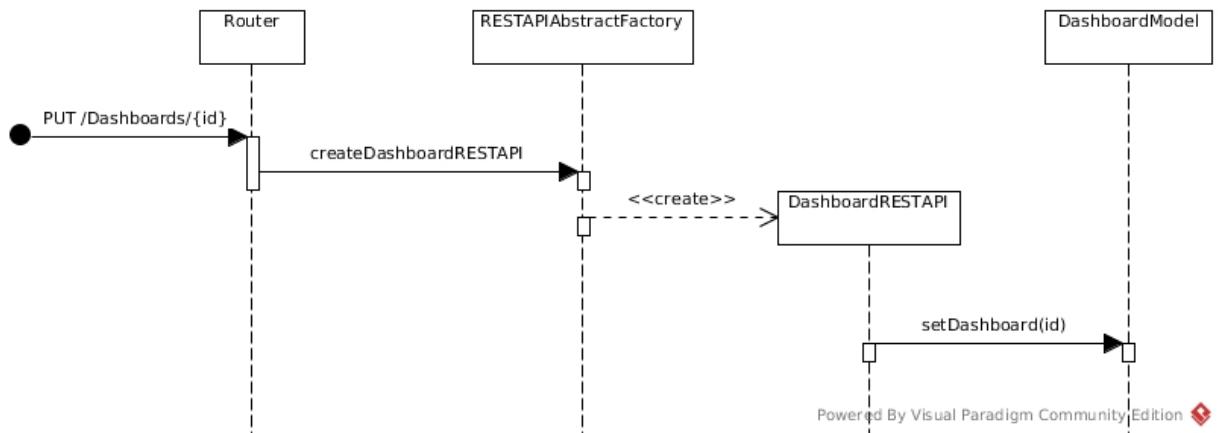


Figura 64: Diagramma di sequenza: PUT /Dashboards/{id}

4.2.3.7.55 Richiesta DELETE /Dashboards/{id}

- **Descrizione:** Eliminazione di una Dashboard mediante il identificativo.
- **Richiesta HTTP:** /Dashboards/{id}
- **Metodo:** DELETE
- **Permessi:** Utente autenticato.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta DELETE per la risorsa /Dashboards/{id}. Creata la DashboardRESTAPI il metodo deleteDashboard(id) richiede al DashboardModel di eliminare un DSLIS rappresentante una Dashboard.

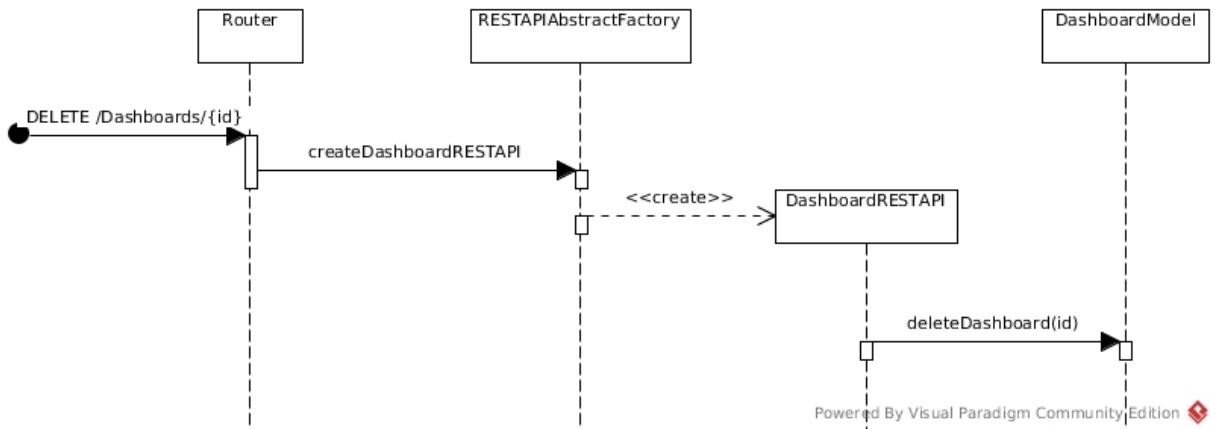


Figura 65: Diagramma di sequenza: DELETE /Dashboards/{id}

4.2.3.7.56 Richiesta GET /Dashboards/retrieveDashboardDSLIS/{id}

- Descrizione:** Ritorna il codice della Dashboard
- Richiesta HTTP:** /Dashboards/retrieveDashboardDSLIS/{id}
- Metodo:** GET
- Permessi:** Utente autenticato
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Dashboards/retrieveDashboardDSLIS/{id}. Creata la DashboardRESTAPI il metodo getDashboardDSLIS(id) richiede al DashboardModel di visualizzare il codice di un DSLIS rappresentante una Dashboard.

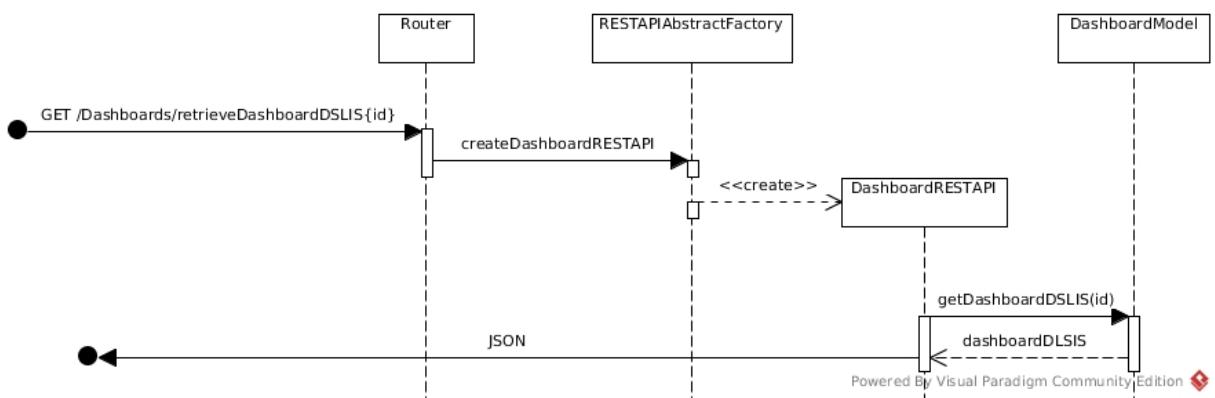


Figura 66: Diagramma di sequenza: GET /Dashboards/retrieveDashboardDSLIS/{id}

4.2.3.7.57 Richiesta GET /Dashboards/searchDashboards/{id}

- Descrizione:** Ricerca di una Dashboard.

- **Richiesta HTTP:** /Dashboards/searchDashboards/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Dashboards/searchDashboards/{id}. Creata la DashboardRESTAPI il metodo getDashboard(id) richiede al DashboardModel di cercare un DSLIS rappresentante una Dashboard.

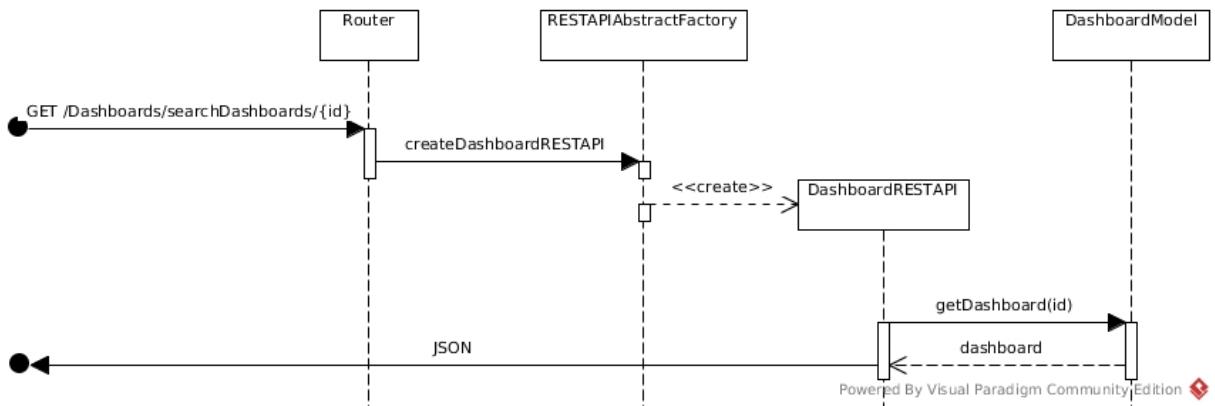


Figura 67: Diagramma di sequenza: GET /Dashboards/searchDashboards/{id}

4.2.3.7.58 Richiesta GET /Dashboards/ExecuteDashboard/{id}

- **Descrizione:** Ritorna l'insieme delle Row della Dashboard.
- **Richiesta HTTP:** /Dashboards/ExecuteDashboard/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Dashboards/ExecuteDashboard/{id}. Creata la DashboardRESTAPI il metodo getDashboardRow(id) richiede al DashboardModel di eseguire un DSLIS rappresentante una Dashboard.

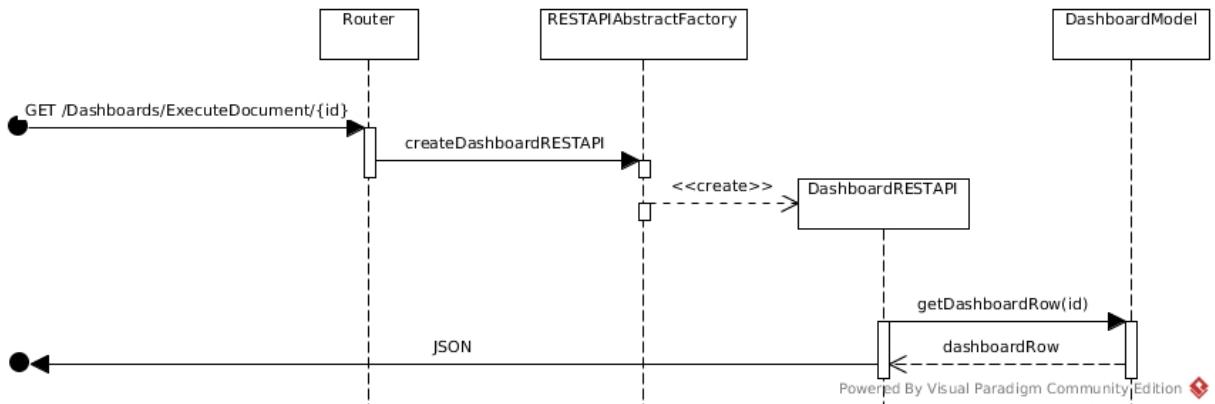


Figura 68: Diagramma di sequenza: GET /Dashboards/ExecuteDashboard/{id}

4.2.3.7.59 Richiesta GET /Dashboards/exportDashboardDSLIS{id}

- Descrizione:** Permette di esportare un DSLIS rappresentante una Dashboard.
- Richiesta HTTP:** /Dashboards/exportDashboardDSLIS{id}
- Metodo:** GET
- Permessi:** Proprietario, Amministratore, Membro
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Dashboards/exportDashboardDSLIS{id}. Creata la DashboardRESTAPI il metodo getDashboardRow(id) richiede al DashboardModel di esportare il codice di un DSLIS rappresentante una Dashboard.

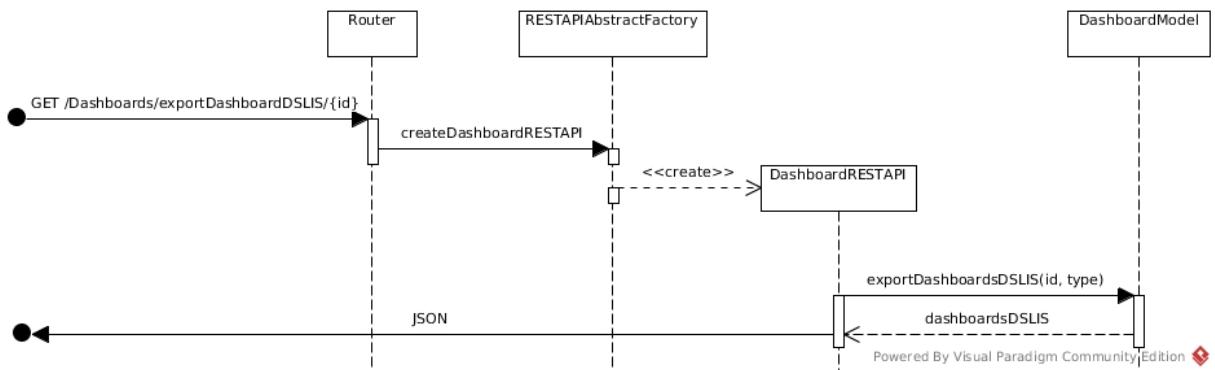


Figura 69: Diagramma di sequenza: GET /Dashboards/exportDashboardDSLIS{id}

4.2.3.7.60 Richiesta POST /Dashboards/importDashboardDSLIS

- Descrizione:** Permette di importare un DSLIS rappresentante una Dashboard.
- Richiesta HTTP:** /Dashboards/importDashboardDSLIS

- **Metodo:** GET
- **Permessi:** Proprietario, Amministratore, Membro.
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Dashboards/importDashboardDSLIS. Creata la DashboardRESTAPI il metodo getDashboardRow(id) richiede al DashboardModel di importare un DSLIS rappresentante una Dashboard.

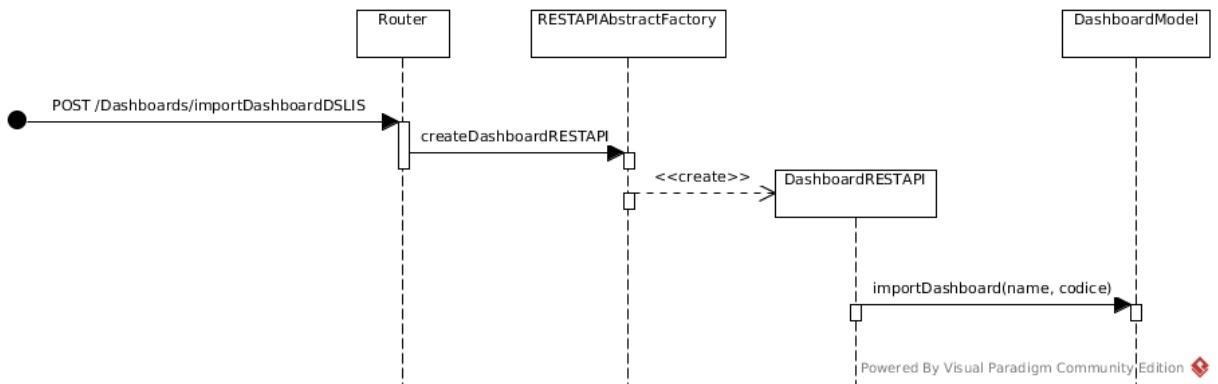


Figura 70: Diagramma di sequenza: POST /Dashboards/importDashboardDSLIS

Richieste Cell

4.2.3.8.61 Richiesta GET /Cells

- **Descrizione:** Mostra la lista dei Cells.
- **Richiesta HTTP:** /Cells
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Cells. Creata la CellRESTAPI il metodo getCells(id) richiede al CellModel di eseguire un DSLIS rappresentante una Cell.

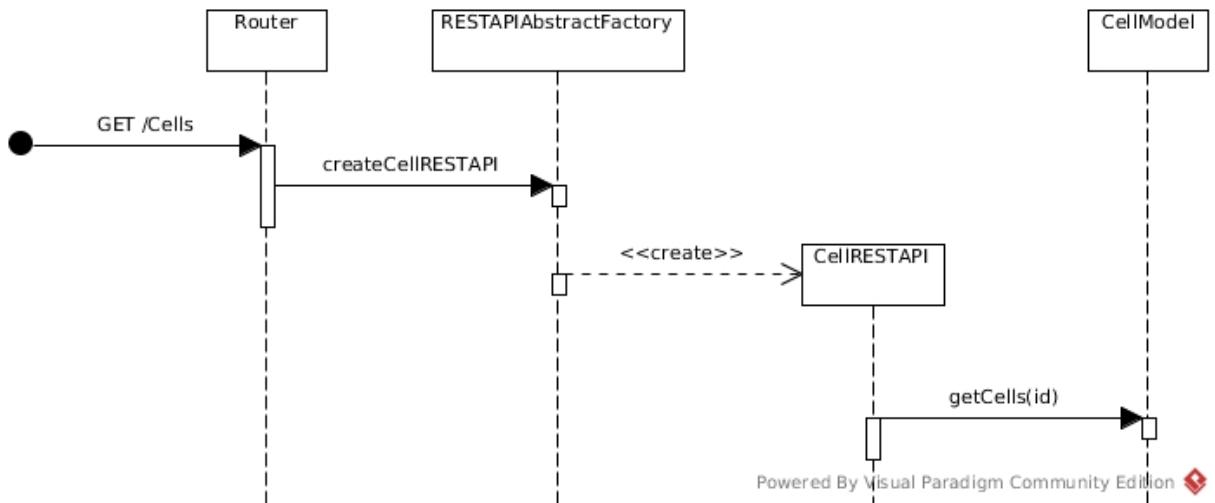


Figura 71: Diagramma di sequenza: GET /Cells

4.2.3.8.62 Richiesta POST /Cells

- **Descrizione:** Creazione di una Cell.
- **Richiesta HTTP:** /Cells
- **Metodo:** POST
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Cells. Creata la CellRESTAPI il metodo createCell(name, codice) richiede al CellModel di creare un nuovo DSLIS rappresentante una Cell.

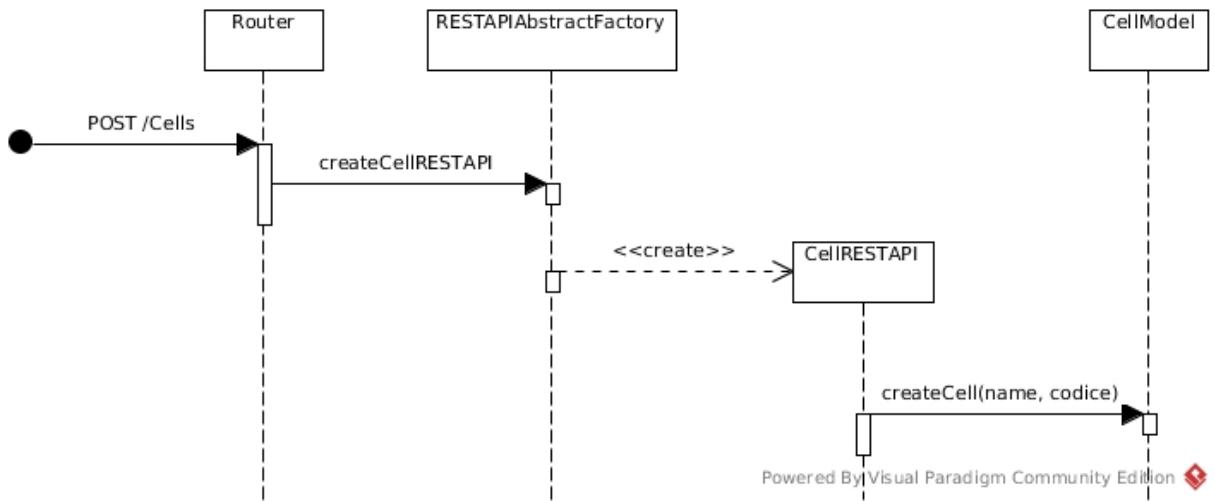


Figura 72: Diagramma di sequenza: POST /Cells

4.2.3.8.63 Richiesta PUT /Cells/{id}

- **Descrizione:** Modifica di una Cell.
- **Richiesta HTTP:** /Cells/{id}
- **Metodo:** PUT
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta PUT per la risorsa /Cells/{id}. Creata la CellRESTAPI il metodo setCell(id, codice) richiede al CellModel di modificare un DSLIS rappresentante una Cell.

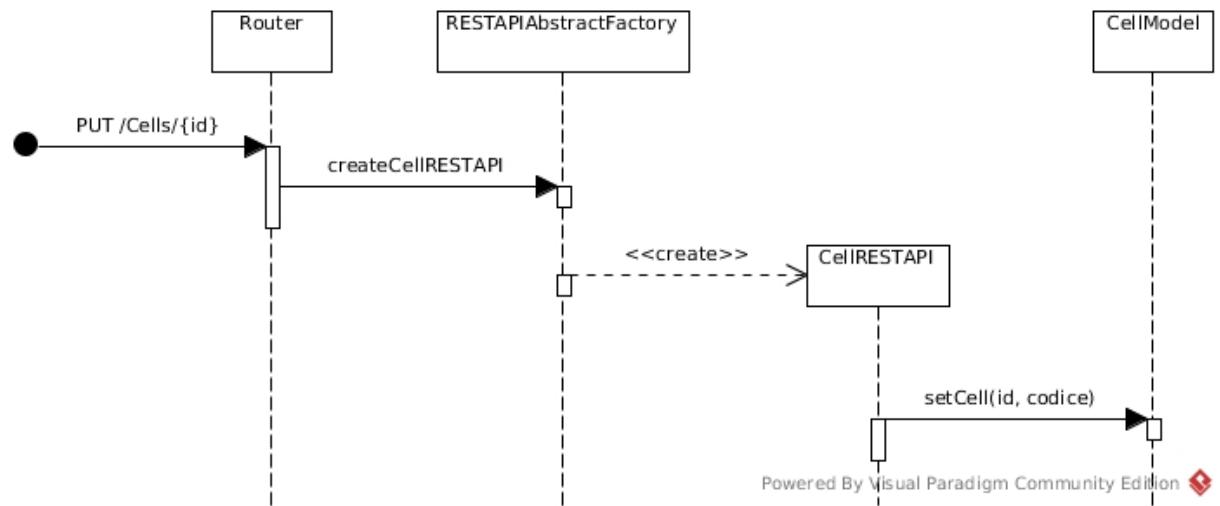


Figura 73: Diagramma di sequenza: PUT /Cells/{id}

4.2.3.8.64 Richiesta DELETE /Cells/{id}

- **Descrizione:** Eliminazione di una Cell mediante il suo identificativo.
- **Richiesta HTTP:** /Cells/{id}
- **Metodo:** DELETE
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta DELETE per la risorsa /Cells/{id}. Creata la CellRESTAPI il metodo deleteCell(id) richiede al CellModel di eliminare un DSLIS rappresentante una Cell.

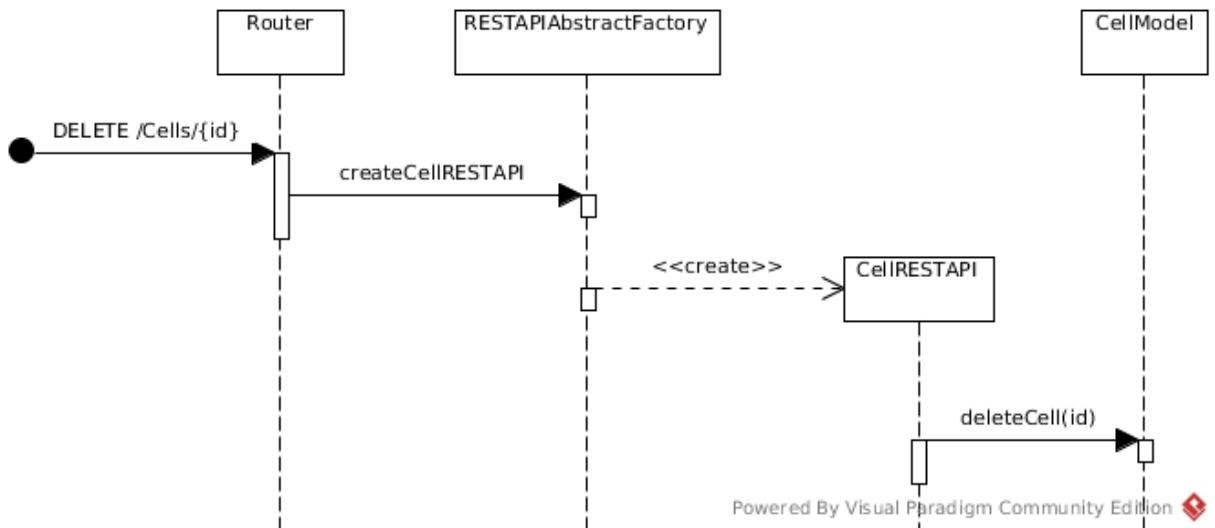


Figura 74: Diagramma di sequenza: DELETE /Cells/{id}

4.2.3.8.65 Richiesta GET /Cells/searchCell/{id}

- **Descrizione:** Ricerca di una Cell.
- **Richiesta HTTP:** /Cells/searchCell/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Cells/searchCell/{id}. Creata la CellRESTAPI il metodo getCell(id) richiede al CellModel di cercare un DSLIS rappresentante una Cell.

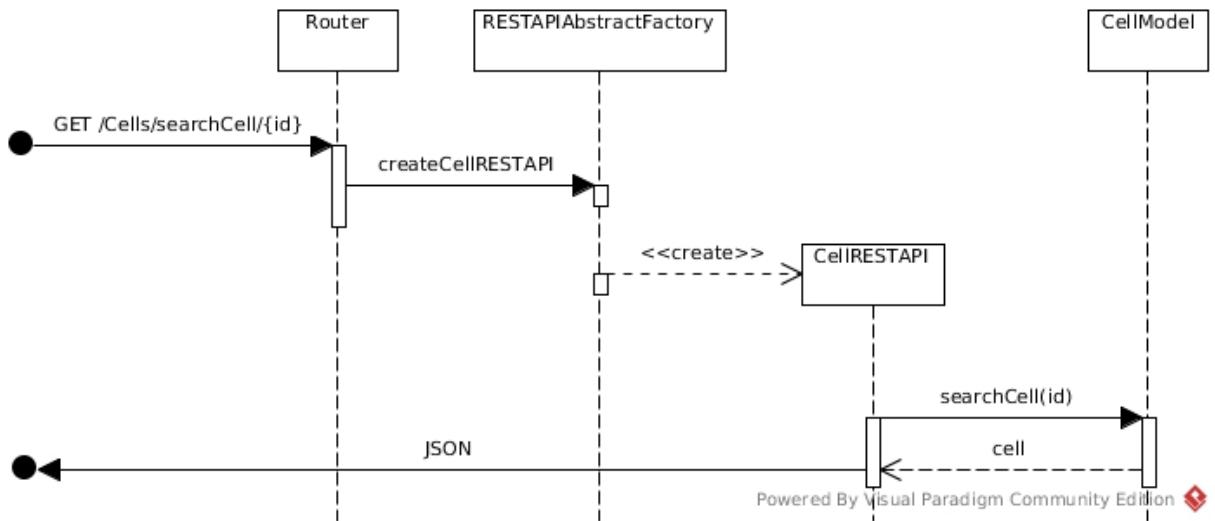


Figura 75: Diagramma di sequenza: GET /Cells/searchCell/{id}

4.2.3.8.66 Richiesta GET /Cells/ExecuteCell/{id}

- **Descrizione:** Esegue la Cell.
- **Richiesta HTTP:** /Cells/ExecuteCell/{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Cells/ExecuteCell/{id}. Creata la CellRESTAPI il metodo executeCell(id) richiede al CellModel di eseguire un DSLIS rappresentante una Cell.

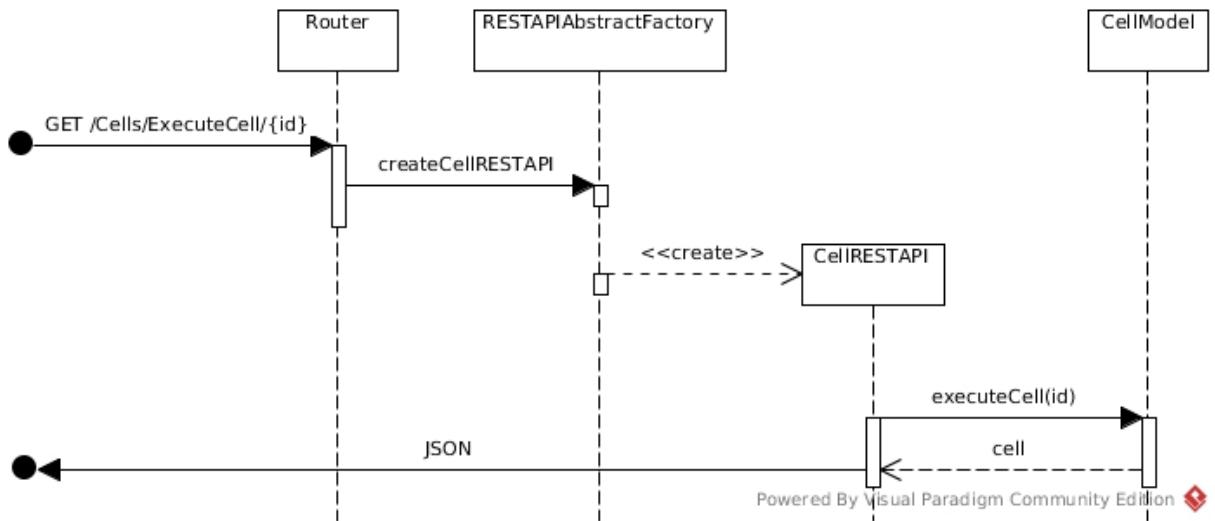


Figura 76: Diagramma di sequenza: GET /Cells/ExecuteCell/{id}

4.2.3.8.67 Richiesta GET /Cells/retrieveCellDSLIS{id}

- **Descrizione:** Ritorna il codice di una Cell.
- **Richiesta HTTP:** /Cells/retrieveCellDSLIS{id}
- **Metodo:** GET
- **Permessi:** Utente autenticato
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Cells/retrieveCellDSLIS. Creata la CellRESTAPI il metodo getCells(id) richiede al CellModel di visualizzare il codice un DSLIS rappresentante una Cell.

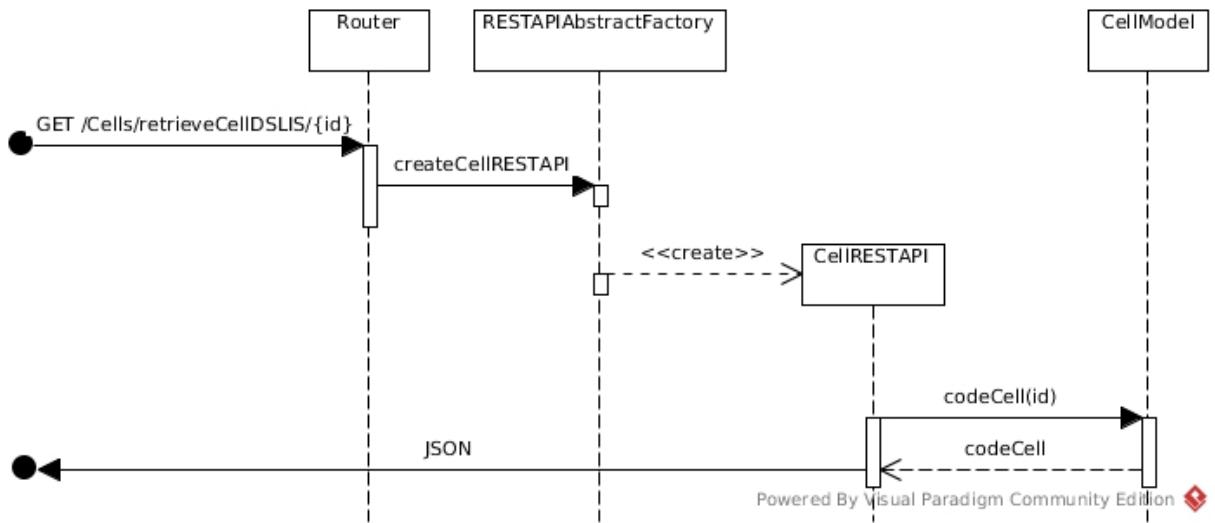


Figura 77: Diagramma di sequenza: GET /Cells/GET /Cells/retrieveCellDSLIS{id}

4.2.3.8.68 Richiesta GET /Cells/exportCellDSLIS/{id}

- **Descrizione:** Permette di esportare un DSLIS rappresentante una Cell.
- **Richiesta HTTP:** /Cells/exportCellDSLIS/{id}
- **Metodo:** GET
- **Permessi:** Proprietario, Amministratore, Membro
- **Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta GET per la risorsa /Cells. Creata la CellRESTAPI il metodo exportCellsDSLIS(id) richiede al CellModel di esportare il codice di un DSLIS rappresentante una Cell.

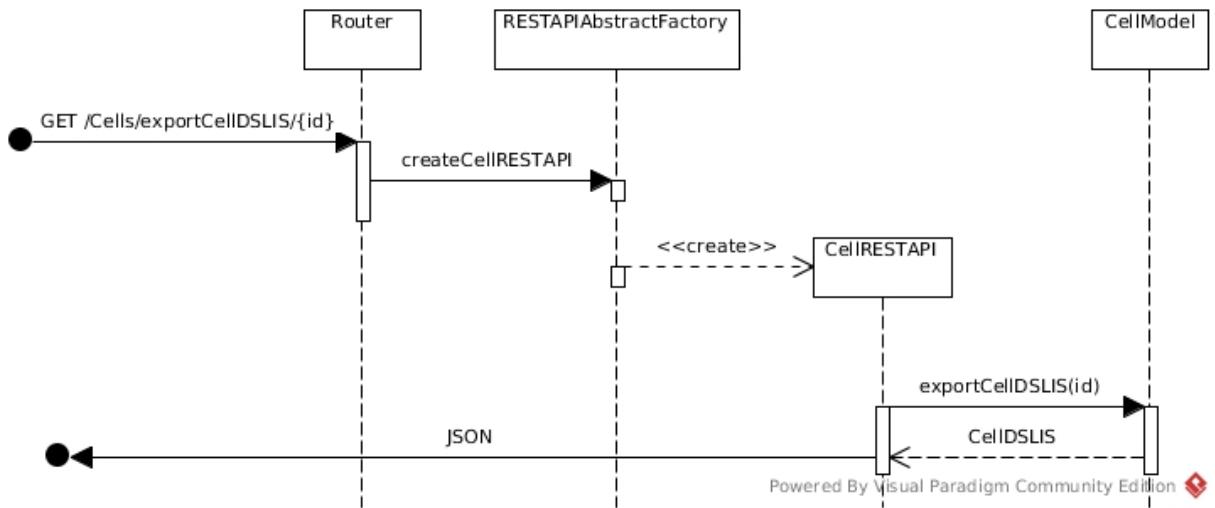


Figura 78: Diagramma di sequenza: GET /Cells/exportCellDSLIS/{id}

4.2.3.8.69 Richiesta POST /Cells/importCellDSLIS

- Descrizione:** Permette di importare un DSLIS rappresentante una Cell.
- Richiesta HTTP:** /Cells/importCellDSLIS
- Metodo:** POST
- Permessi:** Proprietario, Amministratore, Membro
- Scenario:** Il diagramma di sequenza mostra lo scenario di una richiesta POST per la risorsa /Cells/importCellDSLIS. Creata la CellRESTAPI il metodo getCells(id) richiede al CellModel di importare un DSLIS rappresentante una Cell.

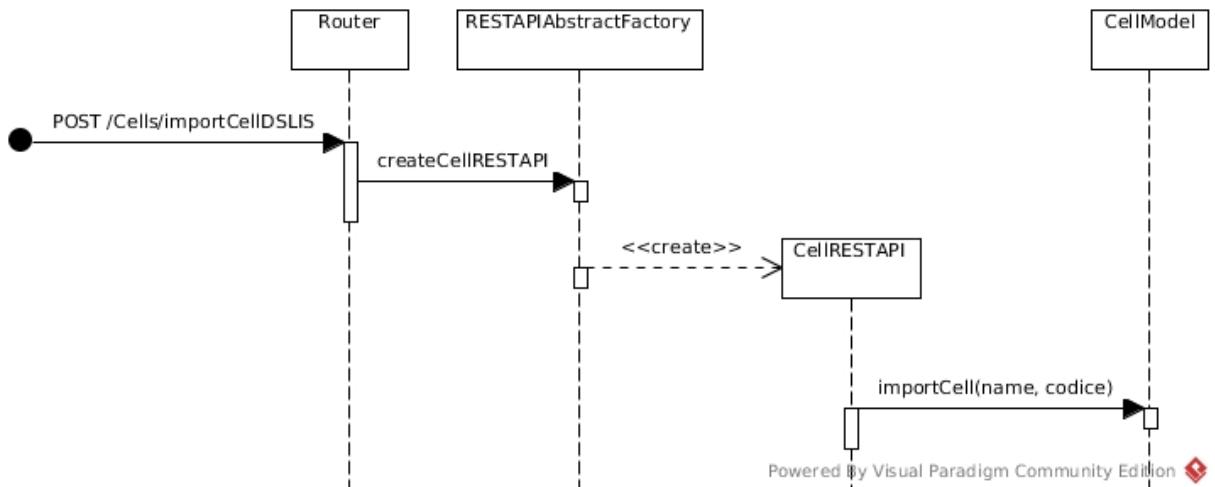


Figura 79: Diagramma di sequenza: POST /Cells/importCellDSLIS

Front-end

Descrizione packages e classi

Front-end

Informazioni sul package

5.1.1.1.1 Descrizione

Package che racchiude tutte le componenti del Front-end. Comprende tutte le classi che rappresentano la parte client dell'applicativo.

Le componenti sono organizzate secondo il design pattern Flux.

5.1.1.1.2 Package contenuti

- ActionCreators
- WebAPIUtils
- Stores
- Views

5.1.1.1.3 Framework esterni

- React

Classi

5.1.1.2.4 Dispatcher

Descrizione

Classe che rappresenta il componente Dispatcher, ovvero colui che ha il compito di inoltrare le action alle relative store. Viene implementata con il design pattern Singleton.

Utilizzo

Viene utilizzato per trasferire le action, create dalle classi contenute nel package Front-end::Actions, alle classi rappresentative della store, contenute nel package Front-end::Stores. Questa classe fornisce un metodo, alle classi ActionCreator, per inviare le action alle relative store. Inoltre, fornisce un metodo, alle classi Store, per ricevere e registrare tali action.

Classi ereditate

- React::React-dispatcher

Relazioni con altre classi

- Front-end::Stores::DSLISStores::DocumentStore
- Front-end::Stores::DSLISStores::DashboardStore
- Front-end::Stores::DSLISStores::CellStore
- Front-end::Stores::DSLISStores::CollectionStore
- Front-end::Stores::UserStore
- Front-end::Stores::SessionStore
- Front-end::Stores::ExternalDatabaseStore
- Front-end::Stores::SuperAmministratoreStore
- Front-end::Stores::CompanyStore

5.1.1.2.5 Application**Descrizione**

Classe che si occupa di lanciare l'applicazione. Viene derivata dal framework React.

Utilizzo

Viene utilizzata per lanciare l'applicazione e mostrare una pagina specifica. Essa istanzia ed utilizza il Router, il quale si occupa di mostrare la pagina iniziale.

Relazioni con altre classi

- Front-end::Router

5.1.1.2.6 Router**Descrizione**

Classe che contiene tutti i percorsi dell'applicazione ed i metodi per raggiungerli.

Utilizzo

Viene utilizzata per mostrare le diverse pagine dell'applicazione ed associarle ad uno specifico URL.

Classi ereditate

- React::React-router

Relazioni con altre classi

- Front-end::Views::DashboardView

Front-end::ActionCreators

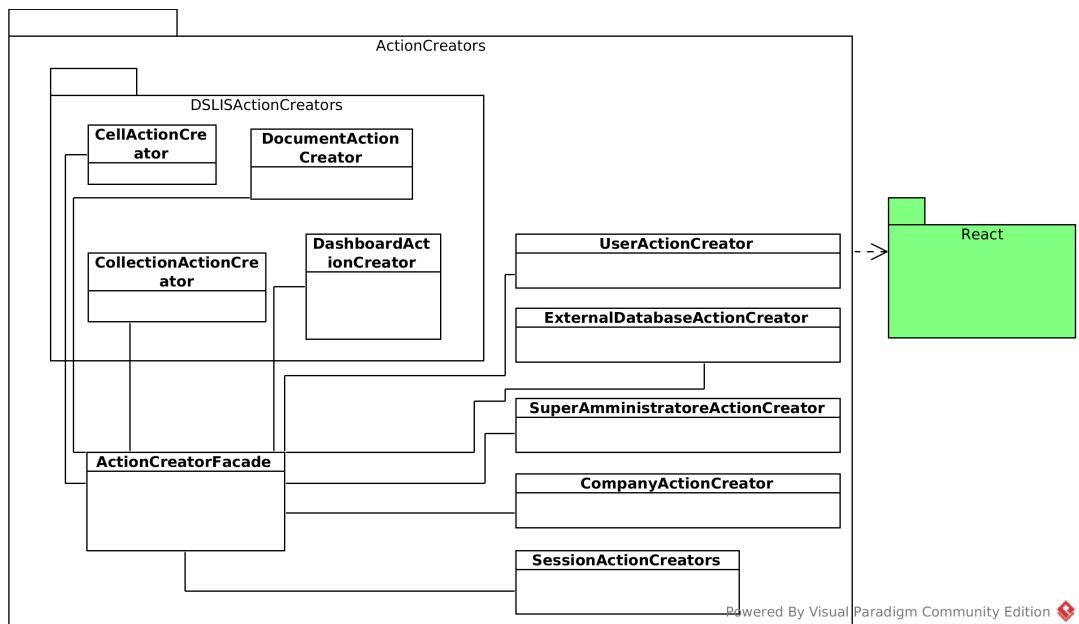


Figura 80: Diagramma delle classi del Action creators

Informazioni sul package

5.1.2.1.1 Descrizione

Package contenente le classi che si occupano di definire le liste di action. Una action è un oggetto che contiene i nuovi campi dati e il nome dell'azione da compiere. Questi oggetti rappresentano i messaggi inviati dalle diverse componenti dell'architettura del Front-end Flux.

5.1.2.1.2 Package contenuti

- **DSLISActionCreators**

5.1.2.1.3 Framework esterni

- React

Classi

5.1.2.2.4 ActionCreatorFacade

Descrizione

Classe che rappresenta l'implementazione del design pattern Facade per le classi contenute nel package Front-end::ActionCreators.

Utilizzo

Viene utilizzata per fornire un'interfaccia comune alle classi ActionCreator.

Relazioni con altre classi

- Front-end::ActionCreators::CompanyActionCreator
- Front-end::ActionCreators::SessionActionCreator
- Front-end::ActionCreators::ExternalDatabaseActionCreator
- Front-end::ActionCreators::UserActionCreator
- Front-end::ActionCreators::SuperAmministratore
- Front-end::ActionCreators::DSLISActionCreators::CollectionActionCreator
- Front-end::ActionCreators::DSLISActionCreators::DashboardActionCreator
- Front-end::ActionCreators::DSLISActionCreators::CellActionCreator
- Front-end::ActionCreators::DSLISActionCreators::DocumentActionCreator
- Front-end::Dispatcher

5.1.2.2.5 UserActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti le interazioni degli utenti. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti le azioni svolte dall'utente. Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti l'utente, l>UserActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti dall>UserWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla UserStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::UserWebAPIUtils

5.1.2.2.6 SessionActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti la sessione. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti la sessione.

Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti la sessione (login, registrazione e logout), la classe SessionActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti da SessionWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla SessionStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::SessionWebAPIUtils

5.1.2.2.7 ExternalDatabaseActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti l'insieme di azioni per gestire i database MongoDB esterni. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti l'insieme di azioni per gestire i database MongoDB esterni.

Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti i database MongoDB esterni, l'ExternalDatabaseActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti dall'ExternalDatabaseWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla ExternalDatabaseStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::ExternalDatabaseWebAPIUtils

5.1.2.2.8 SuperAmministratoreActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti le interazioni del Super Amministratore. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti le azioni svolte dal Super Amministratore.

Inizialmente, quando il Super Amministratore interagisce con la View per compiere delle operazioni, la classe SuperAmministratoreActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti da SuperAmministratoreWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla SuperAmministratoreStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::SuperAmministratoreWebAPIUtils

5.1.2.2.9 CompanyActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti le operazioni sui dati delle aziende. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti le operazioni per gestire i dati delle aziende.

Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti i dati delle aziende, la classe CompanyActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti da CompanyWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla CompanyStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::CompanyWebAPIUtils

Front-end::ActionCreators::DSLISActionCreators

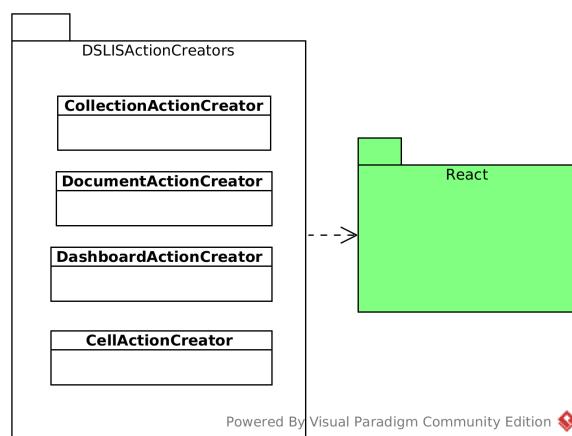


Figura 81: Diagramma delle classi del DSLISActionCreators

Informazioni sul package

5.1.3.1.1 Descrizione

Package contenente le classi che si occupano di definire le liste di action riguardanti l'ambito dei DSLIS.

5.1.3.1.2 Framework esterni

- React

Classi

5.1.3.2.3 CollectionActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti le operazioni sulle Collection, create tramite DSLIS. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti le operazioni per gestire i dati delle Collection, create tramite DSLIS.

Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti le Collection, la classe CollectionActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti da CollectionWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla CollectionStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::DSLISWebAPIUtils::CollectionWebAPIUtils

5.1.3.2.4 DocumentActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti le operazioni sui Document, creati tramite DSLIS. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti le operazioni per gestire i dati dei Document, creati tramite DSLIS.

Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti i Document, la classe DocumentActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti da DocumentWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla DocumentStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::DSLISWebAPIUtils::DocumentWebAPIUtils

5.1.3.2.5 DashboardActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti le operazioni sulle Dashboard, create tramite DSLIS. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti le operazioni per gestire i dati delle Dashboard, create tramite DSLIS.

Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti le Dashboard, la classe DashboardActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti da DashboardWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla DashboardStore.

5.1.3.2.6 CellActionCreator

Descrizione

Classe che si occupa di creare la lista di action riguardanti le operazioni sulle Cell, create tramite DSLIS. Inoltre, essa si occupa, utilizzando il dispatcher, di lanciare le action verso la componente store.

Utilizzo

Viene utilizzata per creare e lanciare le action riguardanti le operazioni per gestire i dati delle Cell, create tramite DSLIS.

Inizialmente, quando l'utente interagisce con la View per compiere delle operazioni riguardanti le Cell, la classe CellActionCreator riceve il nome dell'azione compiuta e i relativi dati. Nel caso sia necessario, la classe utilizza i metodi forniti da CellWebAPIUtils per comunicare tali dati al server e ricevere altri dati da mostrare al client.

Successivamente, la classe impacchetta i dati e il nome dell'azione in un oggetto chiamato action.

Infine, essa utilizza i metodi del dispatcher per inoltrare l'action alla relativa store, ovvero alla CellStore.

Relazioni con altre classi

- Front-end::WebAPIUtils::DSLISWebAPIUtils::CellWebAPIUtils

Front-end::WebAPIUtils

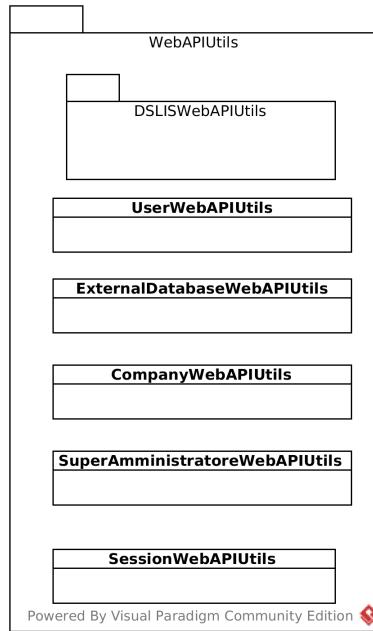


Figura 82: Diagramma delle classi del WEBAPIUtils

Informazioni sul package

5.1.4.1.1 Descrizione

Package contenente le classi che si occupano di fornire dei metodi per interraccarsi con le API fornite dal server da parte del client. Fondamentalmente, sono classi che forniscono metodi che permettono di inviare delle richieste e ricevere delle risposte, in modo da permettere al client di comunicare con il server.

5.1.4.1.2 Package contenuti

- DSLISWebAPIUtils

Classi

5.1.4.2.3 UserWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni svolte dall'utente, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni svolte da parte dell'utente. Le risposte ricevute saranno inoltrate verso l'UserActionCreator, che andrà a creare una action corrispondente.

5.1.4.2.4 SessionWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni svolte per la sessione, in modo da permettere al client di comunicare con il server. In particolare, operazioni quali login, registrazione e logout.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni svolte per la sessione. Le risposte ricevute saranno inoltrate verso la classe SessionActionCreator, che andrà a creare una action corrispondente.

5.1.4.2.5 ExternalDatabaseWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni svolte per gestire i database MongoDB esterni, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni svolte per gestire i database MongoDB esterni. Le risposte ricevute saranno inoltrate verso l'ExternalDatabaseActionCreator, che andrà a creare una action corrispondente.

5.1.4.2.6 CompanyWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni svolte per gestire i dati delle aziende, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni svolte per gestire i dati delle aziende. Le risposte ricevute saranno inoltrate verso la classe CompanyActionCreator, che andrà a creare una action corrispondente.

5.1.4.2.7 SuperAmministratoreWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni svolte dal Super Amministratore, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni svolte dal Super Amministratore. Le risposte ricevute saranno inoltrate verso la classe SuperAmministratoreActionCreator, che andrà a creare una action corrispondente.

Front-end::WebAPIUtils::DSLISWebAPIUtils

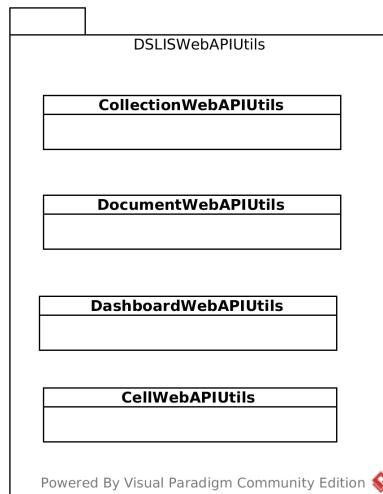


Figura 83: Diagramma delle classi del DSLISWebAPIUtils

Informazioni sul package

5.1.5.1.1 Descrizione

Package contenente le classi che si occupano di fornire dei metodi per interfacciarsi con le API fornite dal server da parte del client, nell'ambito delle operazioni riguardanti i DSLIS.

Classi

5.1.5.2.2 CollectionWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni sulle Collection create mediante i DSLIS, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni sulle Collection. Le risposte ricevute saranno inoltrate verso la classe CollectionActionCreator, che andrà a creare una action corrispondente.

5.1.5.2.3 DocumentWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni sui Document creati mediante i DSLIS, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni sui Document. Le risposte ricevute saranno inoltrate verso la classe DocumentActionCreator, che andrà a creare una action corrispondente.

5.1.5.2.4 DashboardWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni sulle Dashboard create mediante i DSLIS, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni sulle Dashboard. Le risposte ricevute saranno inoltrate verso la classe DashboardActionCreator, che andrà a creare una action corrispondente.

5.1.5.2.5 CellWebAPIUtils

Descrizione

Classe che fornisce i metodi per inviare richieste e ricevere risposte nell'ambito delle operazioni sulle Cell create mediante i DSLIS, in modo da permettere al client di comunicare con il server.

Utilizzo

Viene utilizzata per fornire dei metodi utili ad inviare le richieste HTTP verso le API fornite dal server e riceverne le risposte da parte di esso, nell'ambito delle operazioni sulle Cell. Le risposte ricevute saranno inoltrate verso la classe CellActionCreator, che andrà a creare una action corrispondente.

Front-end::Stores

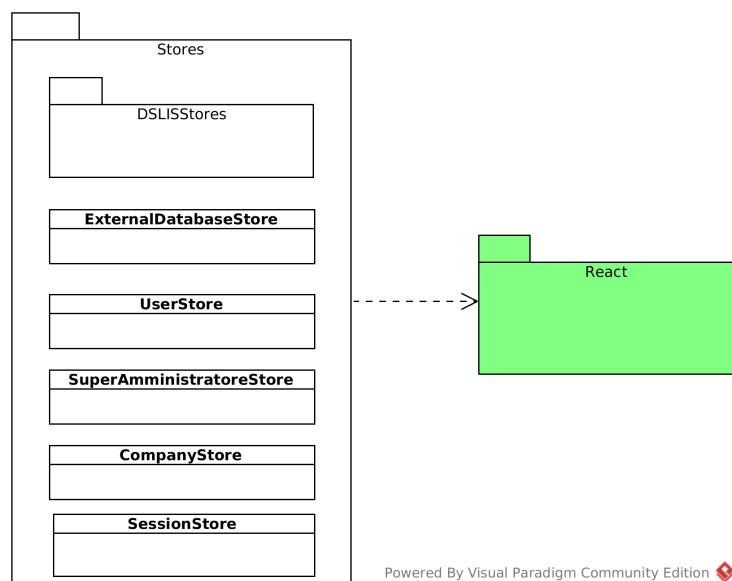


Figura 84: Diagramma delle classi dello stores

Informazioni sul package

5.1.6.1.1 Descrizione

Package contenente le classi che rappresentano i contenitori che contengono la logica dell'applicazione e il relativo stato. Esse ricevono le action inoltrate da parte del dispatcher e ne utilizzano i dati per aggiornare le View.

5.1.6.1.2 Package contenuti

- DSLISStores

5.1.6.1.3 Framework esterni

- React

Classi

5.1.6.2.4 UserStore

Descrizione

Classe che contiene i dati relativi agli utenti.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito utente. Essa riceve le action create dall'UserActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

5.1.6.2.5 SessionStore

Descrizione

Classe contenente i dati relativi alla sessione dell'utente.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante la sessione. Essa riceve le action create dalla classe SessionActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

5.1.6.2.6 SuperAmministratoreStore

Descrizione

Classe che contiene i dati relativi al Super Amministratore.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito Super Amministratore. Essa riceve le action create dalla classe SuperAmministratoreActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

5.1.6.2.7 ExternalDatabaseStore

Descrizione

Classe che contiene i dati relativi ai database MongoDB esterni.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito dei database MongoDB esterni. Essa riceve le action create dell'ExternalDatabaseActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

5.1.6.2.8 CompanyStore

Descrizione

Classe che contiene i dati relativi alle aziende.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito delle aziende. Essa riceve le action create dalla classe CompanyActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

Front-end::Stores::DSLISStores

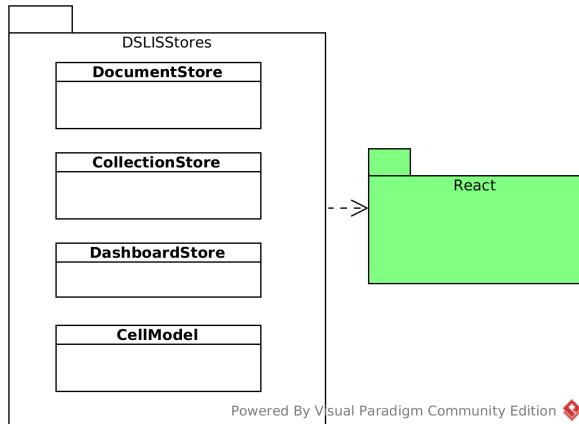


Figura 85: Diagramma delle classi del DSLISStores

Informazioni sul package

5.1.7.1.1 Descrizione

Package contenente le classi che rappresentano i contenitori che contengono la logica dell'applicazione e il relativo stato nell'ambito dei DSLIS.

5.1.7.1.2 Framework esterni

- React

Classi

5.1.7.2.3 CollectionStore

Descrizione

Classe che contiene i dati relativi alle Collection, create mediante i DSLIS.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito delle Collection. Essa riceve le action create dalla classe CollectionActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

5.1.7.2.4 DocumentStore

Descrizione

Classe che contiene i dati relativi ai Document, creati mediante i DSLIS.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito dei Document. Essa riceve le action create dalla classe DocumentActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

5.1.7.2.5 DashboardStore

Descrizione

Classe che contiene i dati relativi alle Dashboard, create mediante i DSLIS.

Utilizzo

Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito delle Dashboard. Essa riceve le action create dalla classe DashboardActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade

5.1.7.2.6 CellStore

Descrizione

Classe che contiene i dati relativi alle Cell, create mediante i DSLIS.

Utilizzo

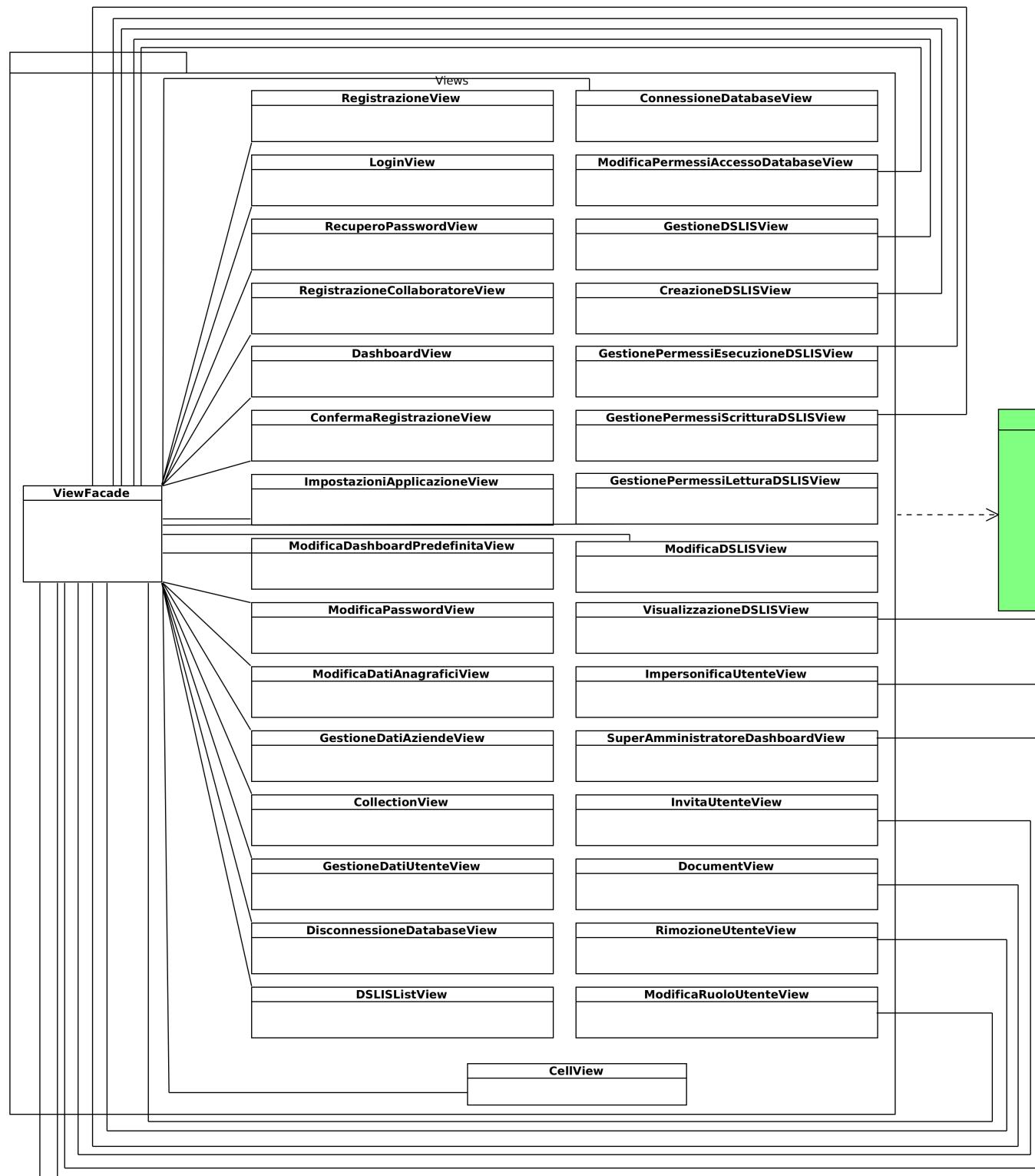
Viene utilizzata per contenere la logica dell'applicazione riguardante l'ambito delle Cell. Essa riceve le action create dalla classe CellActionCreator ed inoltrate da parte del dispatcher e, successivamente, le utilizza per aggiornare la View corrispondente.

Relazioni con altre classi

- Front-end::Views::ViewFacade



Front-end::Views



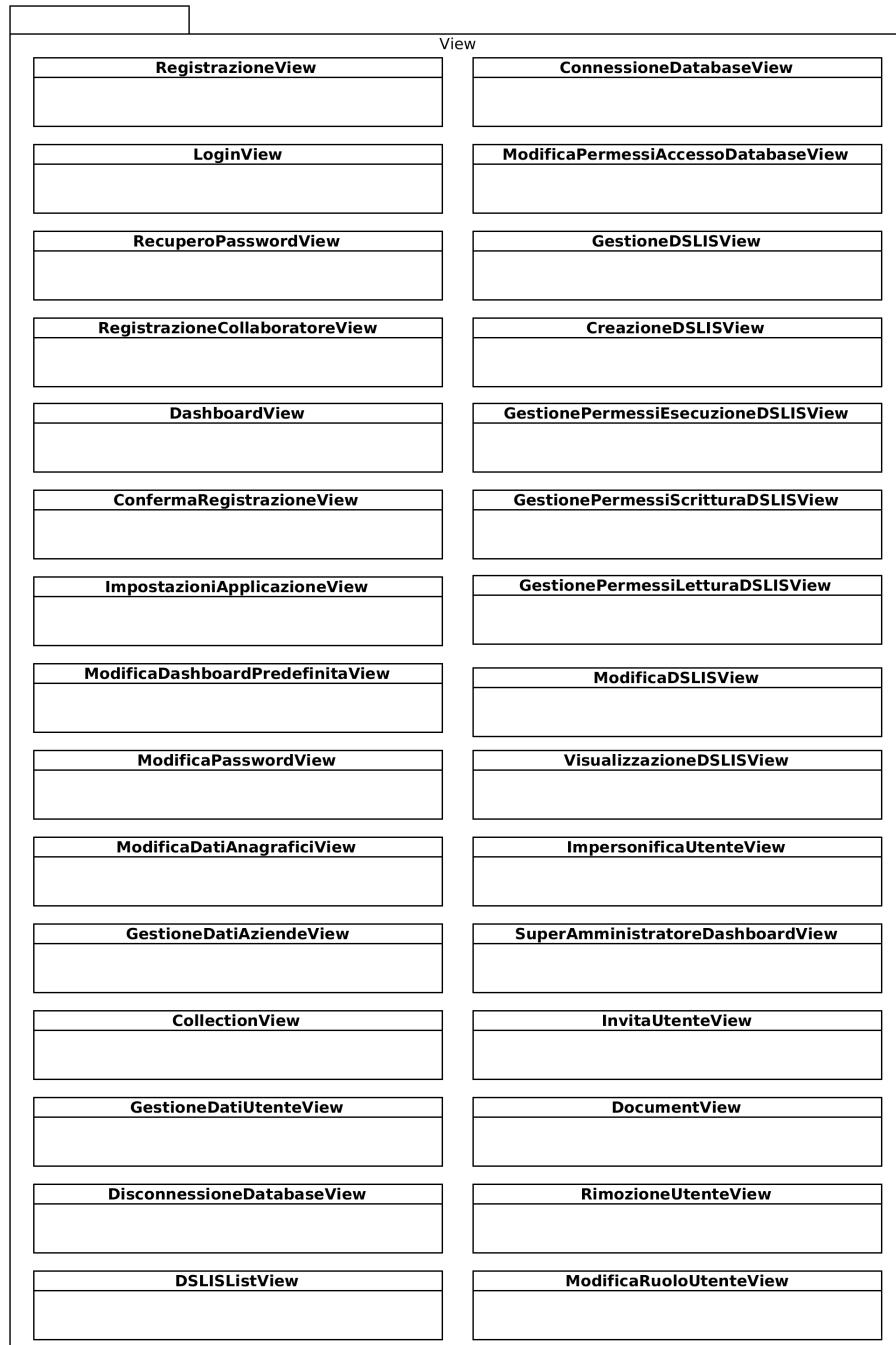


Figura 87: Diagramma delle classi del package View

Informazioni sul package

5.1.8.1.1 Descrizione

Package comprendente le classi che costituiscono la view del componente Front-end del sistema MaaS. Ogni view rappresenta una pagina html con determinati campi, i quali verranno popolati con i dati richiesti, forniti dal sistema per informare l'utente, o forniti dall'utente per compiere determinate azioni.

5.1.8.1.2 Framework esterni

- React

Classi

5.1.8.2.3 ViewFacade

Descrizione

Classe che rappresenta l'implementazione del design pattern Facade per le classi contenute nel package Front-end::Views.

Utilizzo

Viene utilizzata per fornire un'interfaccia comune alle classi Views.

Relazioni con altre classi

- Front-end::ActionCreators::ActionCreatorFacade
- Front-end::Views::RegistrazioneView
- Front-end::Views::ConnessioneDatabaseView
- Front-end::Views::LoginView
- Front-end::Views::ModificaPermessiAccessoDatabaseView
- Front-end::Views::RecuperoPasswordView
- Front-end::Views::GestioneDSLISView
- Front-end::Views::RegistrazioneCollaboratoreView
- Front-end::Views::CreazioneDSLISView
- Front-end::Views::DashboardView
- Front-end::Views::GestionePermessiEsecuzioneDSLISView
- Front-end::Views::ConfermaRegistrazioneView
- Front-end::Views::GestionePermessiScritturaDSLISView
- Front-end::Views::ImpostazioniApplicazioneView
- Front-end::Views::GestionePermessiLetturaDSLISView
- Front-end::Views::ModificaDashboardPredefinitaView

- Front-end::Views::ModificaPasswordView
- Front-end::Views::VisualizzazioneDSLISView
- Front-end::Views::ModificaDatiAnagraficiView
- Front-end::Views::GestioneDatiAziendeView
- Front-end::Views::SuperAmministratoreDashboardView
- Front-end::Views::CollectionView
- Front-end::Views::InvitaUtenteView
- Front-end::Views::GestioneDatiUtenteView
- Front-end::Views::DocumentView
- Front-end::Views::DisconnessioneDatabaseView
- Front-end::Views::RimozioneUtenteView
- Front-end::Views::DSLISListView
- Front-end::Views::ModificaRuoloUtenteView
- Front-end::Views::CellView

5.1.8.2.4 RegistroView

Descrizione

Questa classe descrive la pagina utile a eseguire la registrazione di un nuovo proprietario nel sistema MaaS inserendo negli appositi campi email e password.

Utilizzo

Viene utilizzata per mostrare la pagina di registrazione per un nuovo proprietario; quando un utente interagisce con essa, viene chiamato un metodo dell'UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dell'UserStore.

5.1.8.2.5 LoginView

Descrizione

Questa classe descrive la pagina utile a eseguire il login di un utente non autenticato mettendo a disposizione di esso un form nella quale inserire email e password; inoltre viene fornito un link grazie al quale viene data la possibilità di recuperare la propria password.

Utilizzo

Viene utilizzata per mostrare la pagina di login di un utente non ancora autenticato; quando un utente interagisce con essa, viene chiamato un metodo dell'UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dell'UserStore.

5.1.8.2.6 RecuperoPasswordView

Descrizione

Questa classe descrive la pagina utile a consentire all'utente di modificare la propria password in caso essa sia stata smarrita; al fine di eseguire questa operazione verrà fornito un form nel quale dovrà venire inserita la nuova password e la conferma della stessa.

Utilizzo

Viene utilizzata per mostrare la pagina recupero della password per un utente non ancora autenticato; quando un utente interagisce con essa, viene chiamato un metodo dell'UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dell'UserStore.

5.1.8.2.7 DashboardView

Descrizione

Questa classe descrive la pagina utile a visualizzare la dashboard, predefinita o personalizzata, propria del profilo dell'utente.

Utilizzo

Viene utilizzata per mostrare la pagina nella quale si trova la dashboard; quando un utente interagisce con essa, viene chiamato un metodo dello DashboardActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del DashboardStore.

5.1.8.2.8 DSLISListView

Descrizione

Questa classe descrive la pagina che visualizza la lista di tutti i DSLIS creati dall'utente o condivisi con lo stesso.

Utilizzo

Viene utilizzata per mostrare la pagina nella quale si trova la lista dei DSLIS; quando un utente interagisce con essa, viene chiamato un metodo dello DSLISActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del DSLISStore.

5.1.8.2.9 RegistrazioneCollaboratoreView

Descrizione

Questa classe descrive la pagina che consente ad un utente invitato di potersi registrare al sistema fornendo un form nel quale potrà inserire una password che utilizzerà per accedere a esso.

Utilizzo

Viene utilizzata per mostrare la pagina di registrazione di un nuovo collaboratore; quando un utente interagisce con essa, viene chiamato un metodo dello UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello UserStore.

5.1.8.2.10 ConfermaRegistrazioneView

Descrizione

Questa classe descrive prima pagina che l'utente appena registrato visualizza e che gli confermerà l'avvenuta registrazione; inoltre sarà presente un link che permetterà di entrare nella propria dashboard.

Utilizzo

Viene utilizzata per mostrare la pagina di conferma dell'avvenuta registrazione di un nuovo collaboratore; quando un utente interagisce con essa, viene chiamato un metodo dello UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello UserStore.

5.1.8.2.11 ImpostazioniApplicazioneView

Descrizione

Questa classe descrive la pagina che permette all'utente di modificare alcune impostazioni dell'applicazione.

Utilizzo

Viene utilizzata per mostrare la pagina che contiene le impostazioni dell'applicazione; quando un utente interagisce con essa, viene chiamato un metodo dello UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello UserStore.

5.1.8.2.12 ModificaDashboardPredefinitaView

Descrizione

Questa classe descrive la pagina che permette all'utente di modificare la preferenza riguardo alla scelta della dashboard attiva, fornendo la lista delle dashboard presenti e un bottone che permette la conferma dell'operazione.

Utilizzo

Viene utilizzata per mostrare la pagina nella quale vi sono le funzioni che consentono di modificare la dashboard che andrà a sostituire quella attualmente predefinita; quando un utente interagisce con essa, viene chiamato un metodo dello DashboardActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello DashboardStore.

5.1.8.2.13 ModificaDatiAnagraficiView

Descrizione

Questa classe descrive la pagina che consente di modificare i dati anagrafici dell'utente fornendo form che gli consentono, in particolare, di variare nome, cognome, data di nascita e sesso.

Utilizzo

Viene utilizzata per mostrare la pagina di modifica dei dati anagrafici dell'utente autenticato; quando un utente interagisce con essa, viene chiamato un metodo dello UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello UserStore.

5.1.8.2.14 ModificaPasswordView

Descrizione

Questa classe descrive la pagina che consente all'utente di modificare la propria password fornendo un form nel quale potrà inserire la password attuale, la nuova password e la conferma della stessa.

Utilizzo

Viene utilizzata per mostrare la pagina di modifica della password dell' utente autenticato; quando un utente interagisce con essa, viene chiamato un metodo dello UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello UserStore.

5.1.8.2.15 GestioneDatabaseEsterniView

Descrizione

Questa classe descrive la pagina che consente all'utente di eseguire una serie di operazioni quali: connettere un proprio database al sistema MaaS fornendo un form nel quale potrà inserire il nome e l'indirizzo del nuovo database, disconnettere un database precedentemente importato mediante selezione dello stesso e conferma dell'operazione di eliminazione, gestire i permessi di accesso ai database da parte di altri utenti, in particolare, fornendo la lista dei database presenti, degli utenti dell'azienda e dei radio button che consentono di selezionare il livello di permesso desiderato.

Utilizzo

Viene utilizzata per mostrare la pagina di gestione dei database esterni; quando un utente interagisce con essa, viene chiamato un metodo dello ExternalDataBaseActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello ExternalDatabaseStore.

5.1.8.2.16 GestioneDSLISView

Descrizione

Questa classe descrive la pagina che consente all'utente di visualizzare tutti i DLSIS prodotti dallo stesso o condivisi con lui, sui quali possiede determinati permessi, fornendo buttoni che consentono all'utente di eseguire operazioni quali: creare un nuovo DSLIS, modificare, eliminare, visualizzare il set di istruzioni che compone un DSLIS o eseguirne una esistente.

Utilizzo

Viene utilizzata per mostrare la pagina di gestione dei DSLIS; quando un utente interagisce con essa, viene chiamato un metodo dello DSLISActionCreator , utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello DSLISStore.

5.1.8.2.17 CreazioneDSLISView

Descrizione

Questa classe descrive la pagina utile a fornire strumenti che consentano all'utente di creare un proprio DSLIS, in particolare fornendo un editor di testo e bottoni che permettano ad esempio di salvare il lavoro eseguito.

Utilizzo

Viene utilizzata per mostrare la pagina di creazione di un nuovo DSLIS; quando un utente interagisce con essa, viene chiamato un metodo dello DSLISActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello DSLISStore.

5.1.8.2.18 GestioneDSLISView

Descrizione

Questa classe descrive la pagina che consente all'utente di visualizzare le opzioni utili a:

- Gestire i permessi di esecuzione dei DSLIS in particolare fornendo una lista degli stessi, degli utenti del sistema facenti parte della medesima azienda dell'utente che intende modificare i permessi e un CheckBox che consente di attribuire o meno il permesso in oggetto;
- Gestire i permessi di scrittura dei DSLIS in particolare fornendo una lista degli stessi, degli utenti del sistema facenti parte della medesima azienda dell'utente al quale si intende modificare i permessi e un CheckBox che consente di attribuire o meno il permesso in oggetto;
- Gestire i permessi di lettura dei DSLIS in particolare fornendo una lista degli stessi, degli utenti del sistema facenti parte della medesima azienda dell'utente che al quale si intende modificare i permessi e un CheckBox che consente di attribuire o meno il permesso in oggetto;

Utilizzo Viene utilizzata per mostrare la pagina di gestione dei DSLIS; quando un utente interagisce con essa, viene chiamato un metodo dello DSLISActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello DSLISStore.

5.1.8.2.19 ModificaDSLISView

Descrizione

Questa classe descrive la pagina che consente all'utente di modificare un DSLIS in suo possesso, in particolare fornendo un editor testuale nel quale si troverà il DSLIS.

Utilizzo

Viene utilizzata per mostrare la pagina di modifica di un DSLIS; quando un utente interagisce con essa, viene chiamato un metodo dello DSLISActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello DSLISStore.

5.1.8.2.20 VisualizzazioneDSLISView

Descrizione

Questa classe descrive la pagina utile a visualizzare il DSLIS in oggetto senza che essa possa venire modificata.

Utilizzo

Viene utilizzata per mostrare la pagina di visualizzazione di un DSLIS; quando un utente interagisce con essa, viene chiamato un metodo dello DSLISActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello DSLISStore.

5.1.8.2.21 CollectionView

Descrizione

La classe descrivere la pagina che visualizza i documenti della collection selezionata.

Utilizzo

Viene utilizzata per mostrare la pagina di visualizzazione di una collection; quando un utente interagisce con essa, viene chiamato un metodo del CollectionActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del CollectionStore.

5.1.8.2.22 DocumentView

Descrizione

Questa classe descrive la pagina utile a visualizzare il document selezionata dall'utente attualmente autenticato.

Utilizzo

Viene utilizzata per mostrare la pagina di visualizzazione di un documento; quando un utente interagisce con essa, viene chiamato un metodo del DocumentActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del DocumentStore.

5.1.8.2.23 CellView

Descrizione

Questa classe descrive la pagina utile a visualizzare la Cell selezionata dall'utente attualmente autenticato.

Utilizzo

Viene utilizzata per mostrare la pagina di visualizzazione di una Cell; quando un utente interagisce con essa, viene chiamato un metodo della CellActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del CellStore.

5.1.8.2.24 GestioneUtentiAziendaView

Descrizione

Questa classe descrive la pagina che consente di visualizzare una serie di opzioni utili all'utente quali:

- Rimuovere un utente dal sistema e in particolare dall'azienda cui fa parte;
- Visualizzarare un form nel quale inserire l'email dell'utente da invitare e il ruolo che avrà all'interno dell'azienda.
- Modificare il ruolo di un utente, permettendo la sua selezione e la selezione del tipo di modifica che si vuole attuare, in particolare, promozione o retrocessione.

Utilizzo

Viene utilizzata per mostrare la pagina di gestione degli utenti appartenenti ad una specifica azienda; quando un utente interagisce con essa, viene chiamato un metodo dello UserActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte dello UserStore.

5.1.8.2.25 SuperAmministratoreDashboardView

Descrizione

Questa classe descrive la pagina utile a visualizzare la dashboard, predefinita o personalizzata, propria del profilo del super amministratore.

Utilizzo

Viene utilizzata per mostrare la pagina nella quale si trova la dashboard del super amministratore; quando un utente interagisce con essa, viene chiamato un metodo dello DashboardActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del DashboardStore.

5.1.8.2.26 GestioneDatiAziendeView

Descrizione

Questa classe descrive la pagina che consente al super amministratore di gestire i dati relativi ad una specifica azienda registrato nel sistema.

Utilizzo

Viene utilizzata per mostrare la pagina di gestione dei dati delle aziendali; quando un utente interagisce con essa, viene chiamato un metodo del SuperAmministratoreActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del CompanyStore.

5.1.8.2.27 GestioneDatiUtenteView

Descrizione

Questa classe descrive la pagina che consente al super amministratore di gestire i dati relativi a uno specifico utente registrato nel sistema MaaS e di fornire gli strumenti che gli consentono di impersonificarsi in esso.

Utilizzo

Viene utilizzata per mostrare la pagina di gestione dei dati di un utente; quando il super amministratore interagisce con essa, viene chiamato un metodo del SuperAmministratoreActionCreator, utile a creare una action corrispondente all'azione eseguita. Inoltre, riceve i dati utili ad aggiornare il proprio stato da parte del SuperAmministratoreStore.

Diagrammi di Attività

In questa sezione vengono illustrati i diagrammi di attività sviluppati durante la progettazione architettonale, che descrivono i modi in cui l'utente interagisce con il sistema MaaS. Si parte illustrando le attività principali eseguibili sul sistema, sia prima che dopo l'autenticazione, per poi scendere analizzando in dettaglio attività più specifiche, disponibili soltanto ad alcuni tipi di utente.

Abbiamo inserito all'interno di alcuni diagrammi, in modo da renderne più chiara la lettura, qualche specifica sugli attori delle transizioni.

Sistema MaaS - Attività principali

Vengono di seguito illustrati i modi in cui un utente possa interagire con il sistema MaaS.

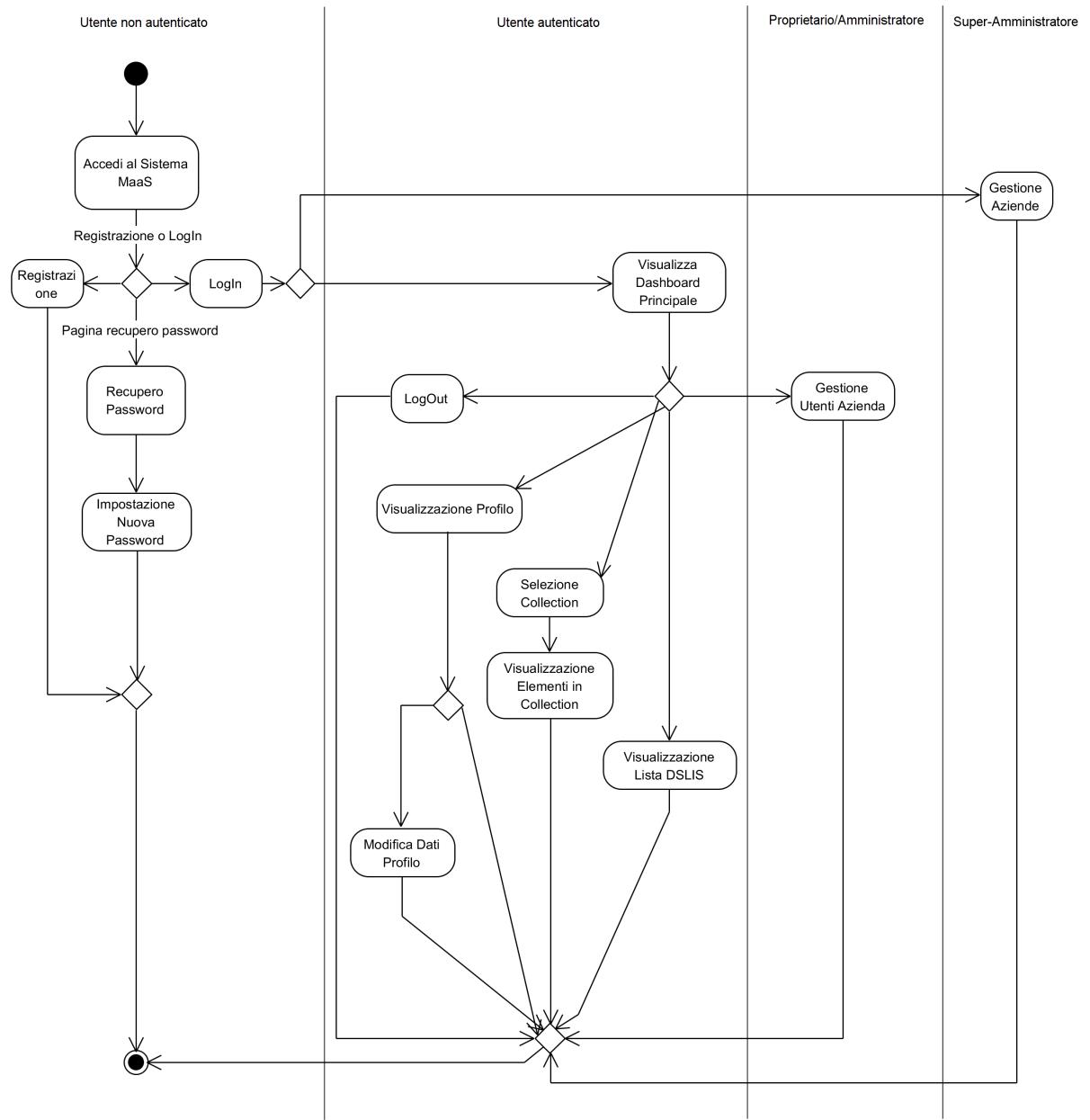


Figura 88: Attività principali del sistema MaaS

Il front-end del sistema è composto da una serie di pagine web interconnesse, attraverso le quali l'utente deve navigare per effettuare le varie operazioni. Il primo impatto con MaaS avviene attraverso una pagina statica, nella quale è possibile effettuare tre operazioni:

- **Registrazione;**
- **LogIn;**

- **Recupero password.**

Eseguito con successo il LogIn, l'utente verrà indirizzato alla pagina della Dashboard principale, dalla quale sarà possibile interagire con le funzioni del sistema. In particolare sarà possibile:

- **Effettuare il LogOut** (funzione disponibile in qualunque pagina del sistema visualizzata dopo il LogIn);
- **Visualizzare e modificare i dati del proprio profilo;**
- **Selezionare una Collection esistente;**
- **Visualizzare la lista dei DSLIS disponibili.**

Inoltre, se l'utente ha privilegi amministrativi, sarà visualizzata anche una sezione dalla quale sarà possibile modificare i dati degli utenti dell'azienda, mentre se lo stesso ha privilegi super-amministrativi, sarà visualizzata una sezione per la gestione delle aziende.

Utente generico

Questa è la categoria di attività disponibili a tutti gli utenti, dall'Ospite al Proprietario.

Registrazione nuovo proprietario

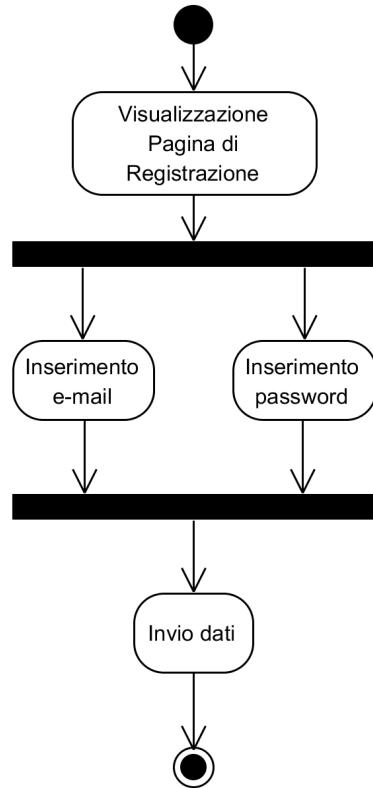


Figura 89: Attività di Registrazione al sistema MaaS

L'utente si trova nella pagina di registrazione, nella quale deve inserire l'indirizzo email, la password e cliccare sul pulsante di invio dei dati. Il sistema effettuerà dunque un controllo delle credenziali e, se corrette (indirizzo email disponibile) registrerà l'utente nel database come nuovo proprietario di un'azienda.

Autenticazione

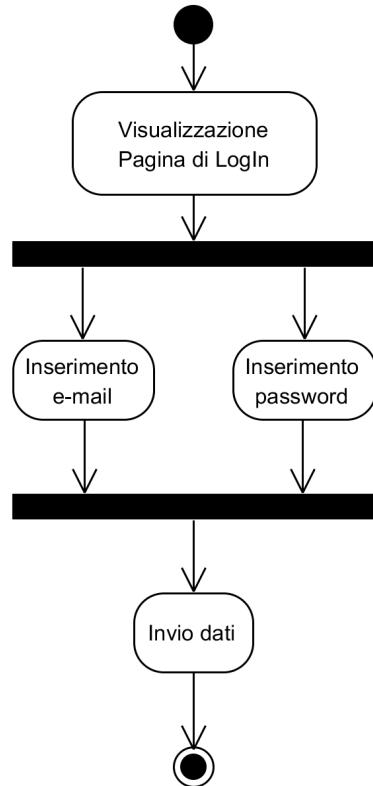


Figura 90: Attività di Autenticazione

L'utente si trova nella pagina di LogIn, nella quale deve inserire le proprie credenziali di accesso (email e password) e premere il pulsante di invio dei dati. Il sistema provvederà a controllare che le credenziali siano corrette (comparandole con quelle salvate nel database degli utenti) e, se verificate, attiverà l'account dell'utente e lo porterà alla pagina di visualizzazione della Dashboard principale.

Recupero password



Figura 91: Attività di Recupero Password

L'utente si trova nella pagina di recupero password, in cui troverà un campo in cui dovrà inserire il proprio indirizzo email, per poi premere il pulsante di invio. Il sistema verificherà l'indirizzo e, se presente nel database degli utenti, manderà allo stesso un messaggio contenente un link per effettuare il reset della password.

Reset password

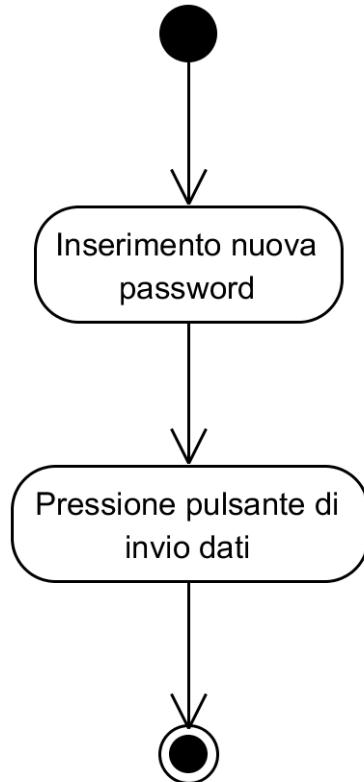


Figura 92: Attività di Reset Password

L'utente arriva a questa pagina attraverso il link ricevuto dallo strumento di recupero password. Ora dovrà inserire la nuova password nell'apposito campo e premere il pulsante di invio dei dati, il sistema sostituirà la password presente nel database con quella nuova.

LogOut

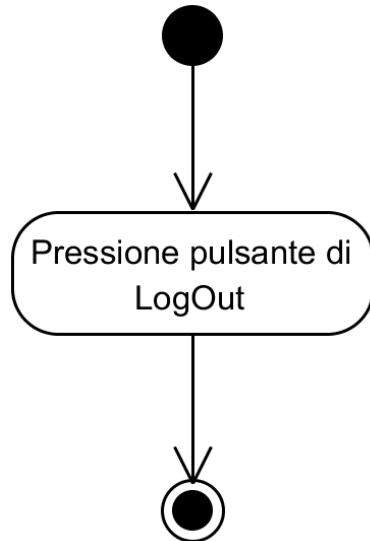


Figura 93: Attività di LogIn

L'utente, dopo aver effettuato l'autenticazione, potrà vedere in qualsiasi punto del sistema MaaS un pulsante per la disconnessione. Tutti i tipi di utente potranno affrontare questa procedura che, qualora andasse a buon fine, chiuderà l'account attivo eliminando tutti i dati di sessione. La pagina che si presenterà subito dopo sarà quella di introduzione al sistema, dal quale sarà possibile effettuare nuovamente il login o una registrazione.

Modifica impostazioni personali del sistema

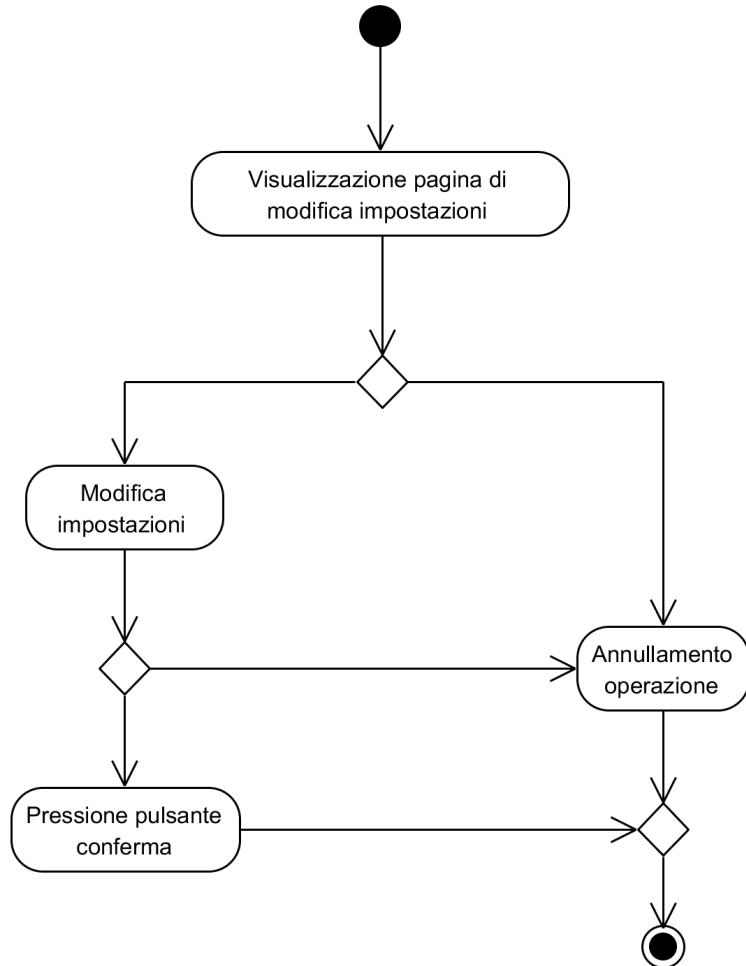


Figura 94: Attività di Modifica delle impostazioni personali

L'utente autenticato, dopo esser entrato nella pagina di visualizzazione del proprio profilo, può accedere a quella che gli permetterà di modificare impostazioni personali (proprie di ogni utente) del sistema. L'operazione può essere annullata in qualsiasi momento.

Modifica profilo

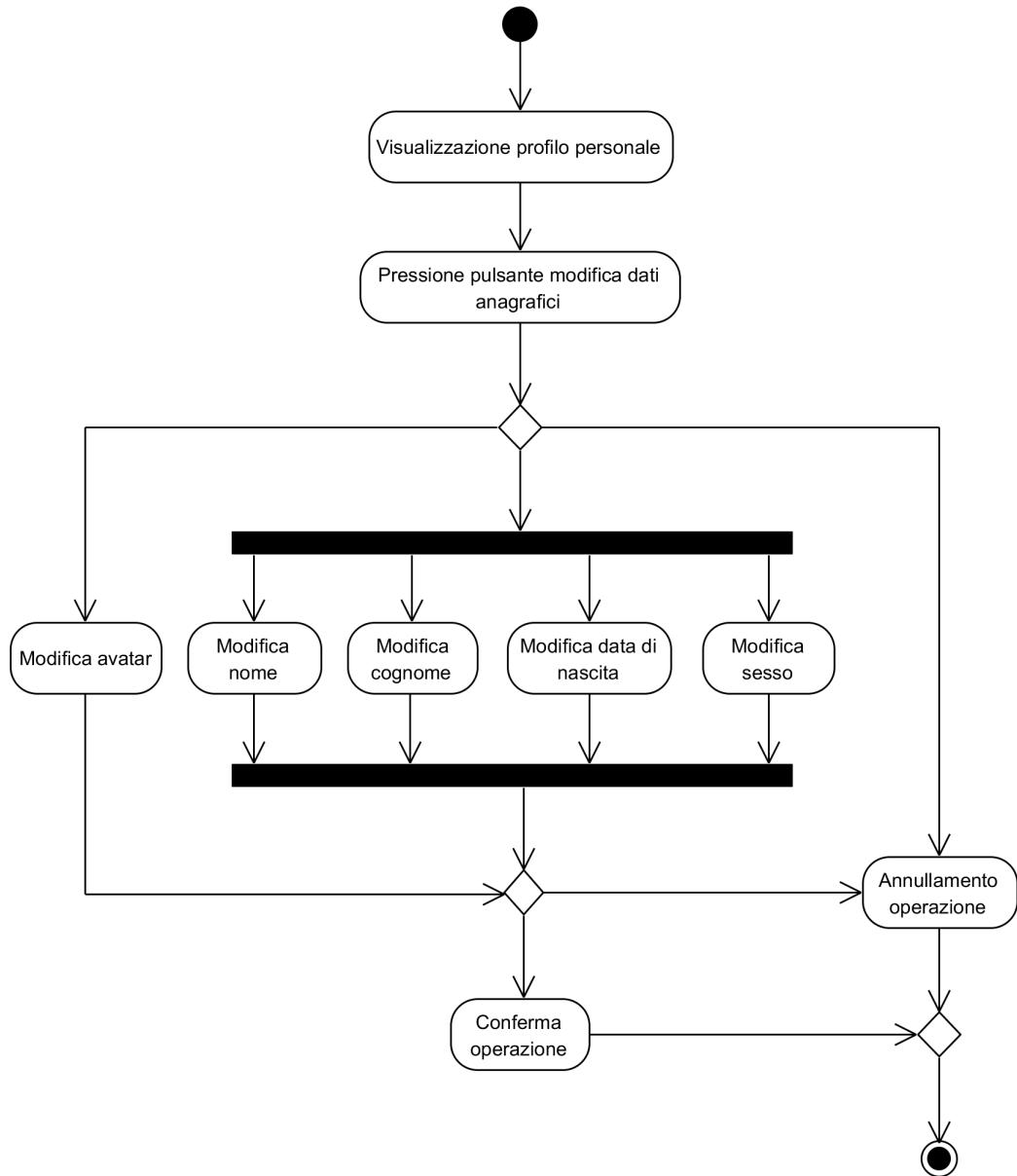


Figura 95: Attività di Modifica dei dati personali

L'utente autenticato, dopo esser entrato nella pagina di visualizzazione del proprio profilo, può accedere a quella che gli permetterà di modificare i dati personali (come ad es. anagrafiche e avatar). L'operazione può essere annullata in qualsiasi momento.

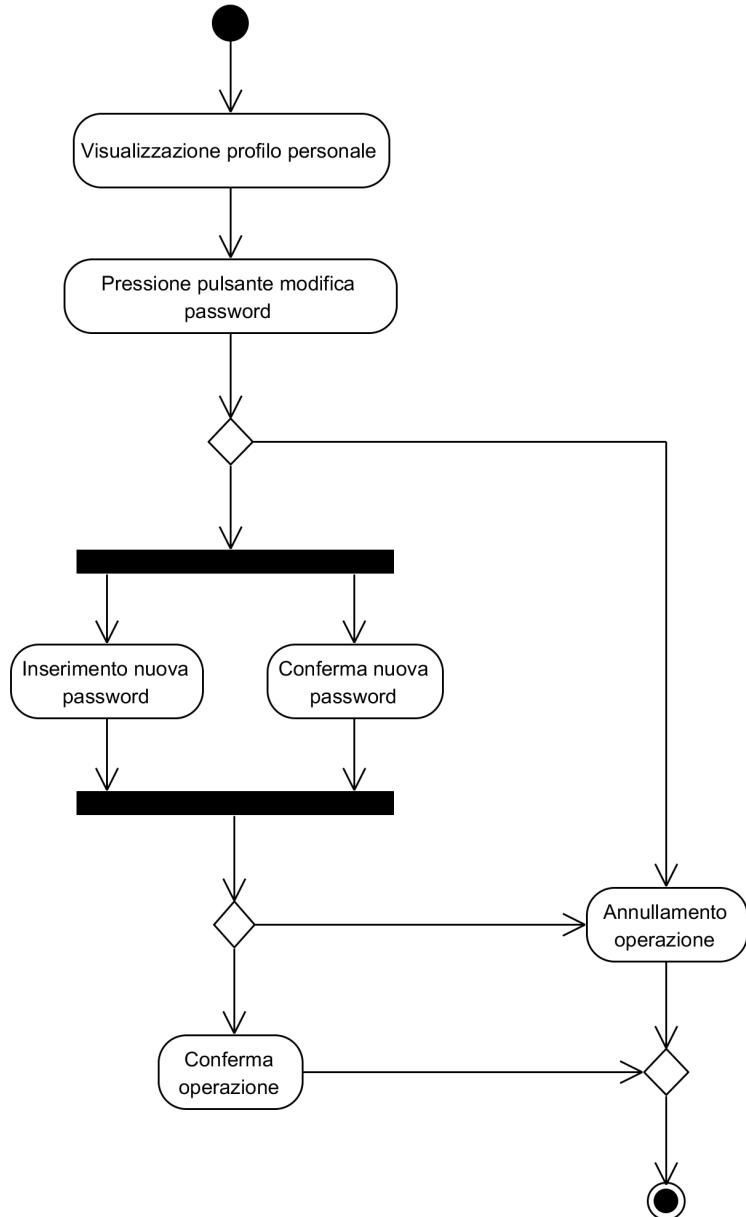
Modifica password da profilo


Figura 96: Attività di Modifica della password dopo l'accesso al profilo

L'utente autenticato, dopo esser entrato nella pagina di visualizzazione del proprio profilo, può accedere a quella che gli permetterà di modificare la password. In questa troverà due campi, uno per l'inserimento della nuova password, l'altro per la conferma della stessa. Oltre

a questi sarà presente il pulsante di conferma delle modifiche effettuate. L'operazione può essere annullata in qualsiasi momento.

Modifica Dashboard principale

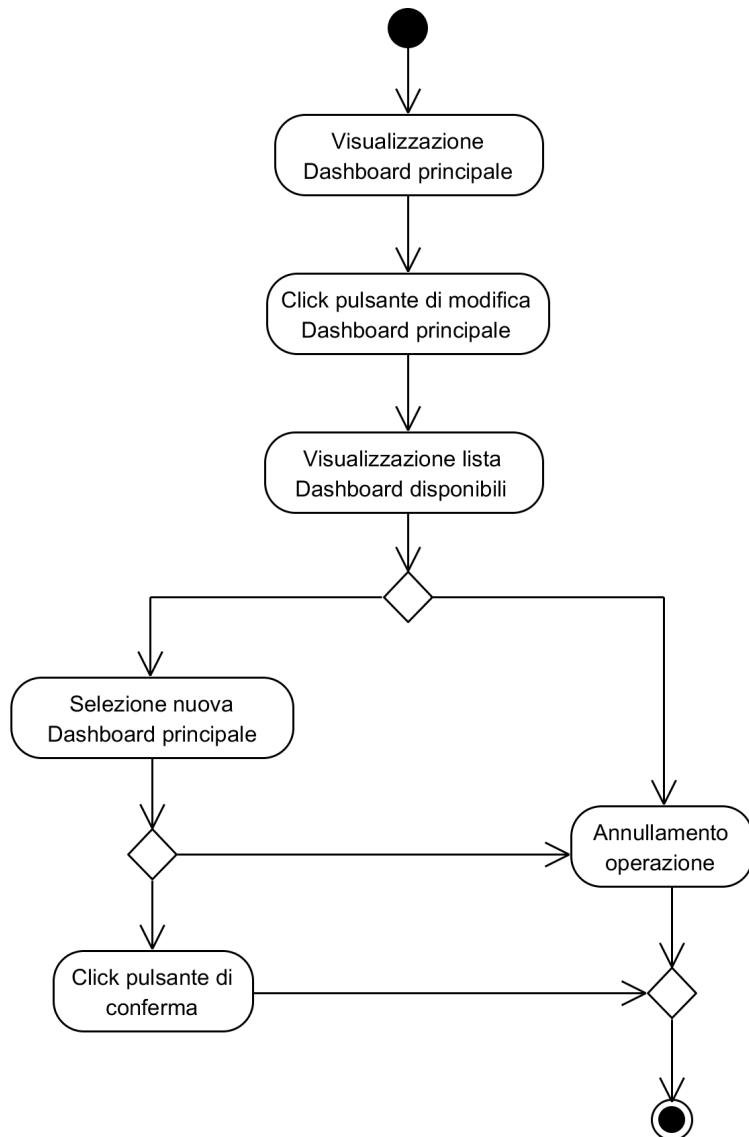


Figura 97: Attività di Modifica della Dashboard principale dell'account

L'utente, dopo aver effettuato il login al sistema, visualizza la pagina contenente Dashboard principale. Da qua, premendo l'apposito pulsante, può accedere a quella per la modifica di tale Dashboard, nella quale sarà visualizzata la lista di quelle disponibili (create

precedentemente tramite DSLIS). Dopo aver effettuato la scelta, l'utente potrà confermare la modifica attraverso un pulsante di accettazione. L'operazione può essere annullata in qualsiasi momento.

Creazione DSLIS

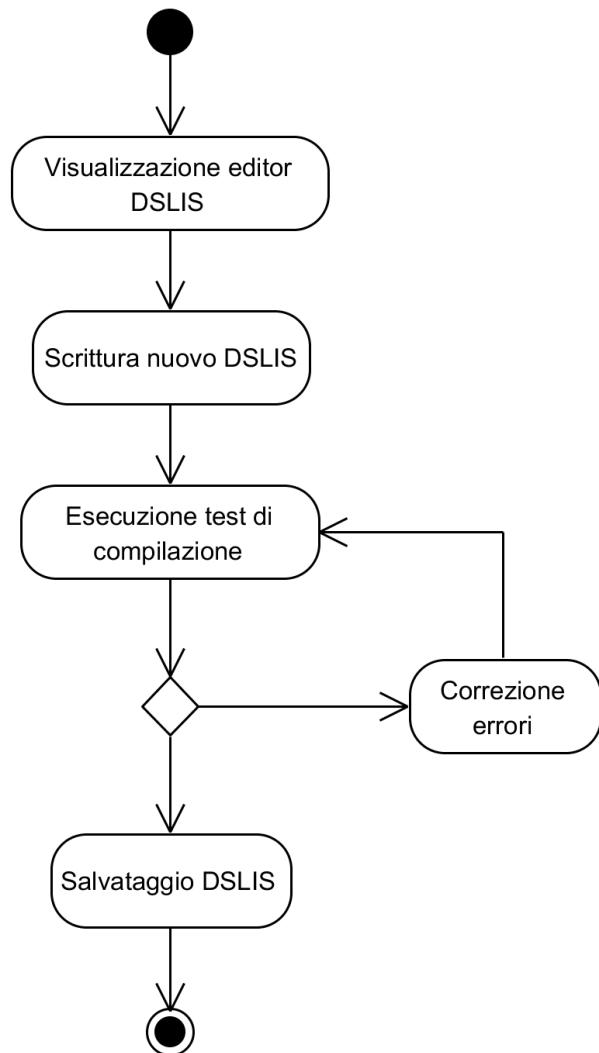


Figura 98: Attività di Creazione di un nuovo DSLIS

L'utente si trova nella pagina di visualizzazione della lista dei DSLIS presenti nel sistema. Cliccando sul pulsante apposito, qualora avesse i permessi per farlo, potrà accedere a quella per la creazione di un nuovo DSLIS. Visualizzerà a questo punto l'editor di testo per la scrittura, verifica e salvataggio del codice. L'operazione potrà essere annullata in qualsiasi momento.

Modifica DSLIS

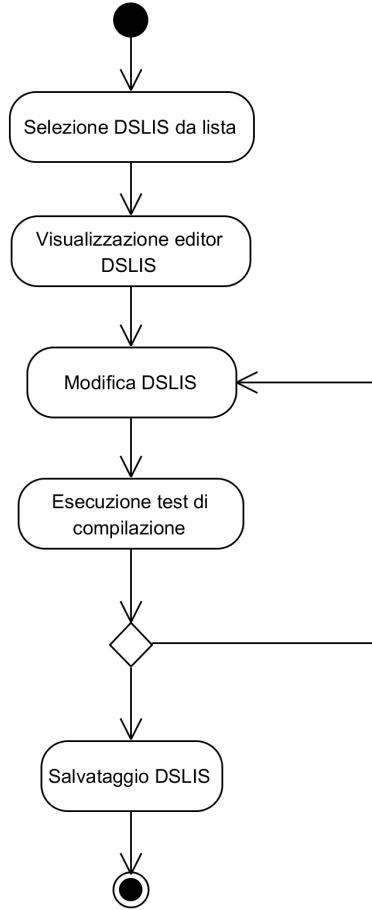


Figura 99: Attività di Modifica di un DSLIS esistente

L'utente si trova nella pagina di visualizzazione della lista dei DSLIS presenti nel sistema. Dopo aver selezionato uno specifico DSLIS potrà, qualora avesse i permessi per farlo, cliccare sul pulsante per la modifica ed accedere alla pagina corrispondente. Visualizzerà a questo punto l'editor di testo per la scrittura, verifica e salvataggio del codice. L'operazione potrà essere annullata in qualsiasi momento.

Esecuzione DSLIS

Figura 100: Attività di Esecuzione di un DSLIS esistente

L'utente si trova nella pagina di visualizzazione della lista dei DSLIS presenti nel sistema. Dopo aver selezionato uno specifico DSLIS potrà, qualora avesse i permessi per farlo, cliccare sul pulsante per la sua esecuzione. Visualizzerà a questo punto una pagina contenente la vista risultato del DSLIS eseguito.

Eliminazione DSLIS

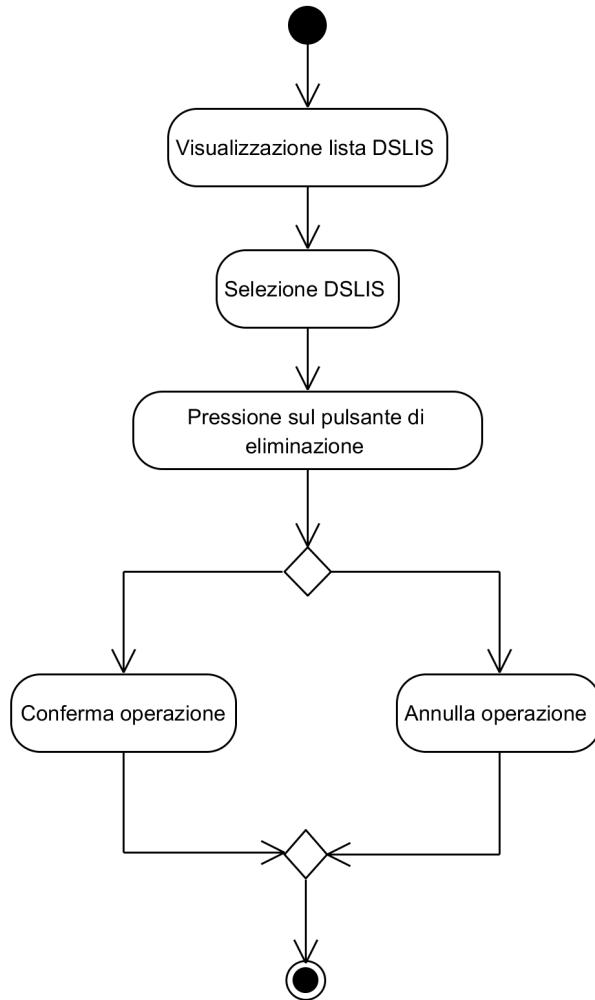


Figura 101: Attività di Eliminazione di un DSLIS esistente

L'utente si trova nella pagina di visualizzazione della lista dei DSLIS presenti nel sistema. Dopo aver selezionato uno specifico DSLIS potrà, qualora avesse i permessi per farlo (permessi di scrittura), cliccare sul pulsante per la sua eliminazione. Verrà quindi mostrata una *Dialog Box_G* con un'avvertenza riguardante l'azione che si sta per svolgere, da questa l'utente potrà confermare o annullare l'operazione.

Gestione Utenti

Questa è la categoria di attività disponibili al Proprietario e all' Amministratore.

Invito di un nuovo utente

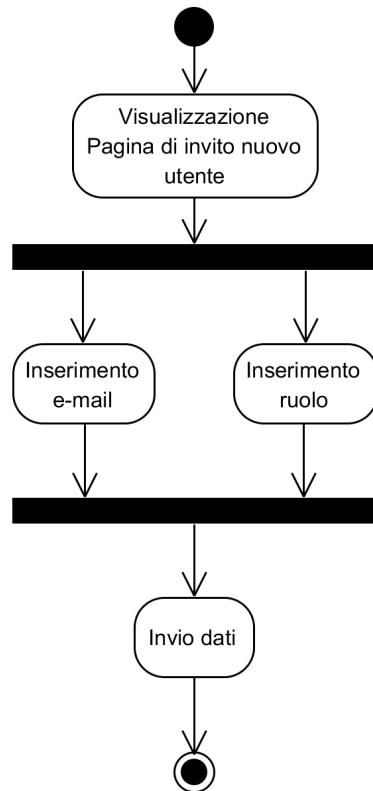


Figura 102: Attività di invito di un nuovo utente

Il Proprietario o un Amministratore può inserire un nuovo utente all'interno del sistema, tramite un'apposita procedura di invito. Dopo essere entrato nella pagina di invito, gli basterà inserire nel form in essa contenuta la mail di colui che sarà il nuovo utente ed il ruolo che si intende fargli avere (disponibile tra Ospite, Membro e Amministratore). Dovrà poi cliccare sul pulsante di invio dei dati per convalidare l'operazione.

Modifica del ruolo di un utente

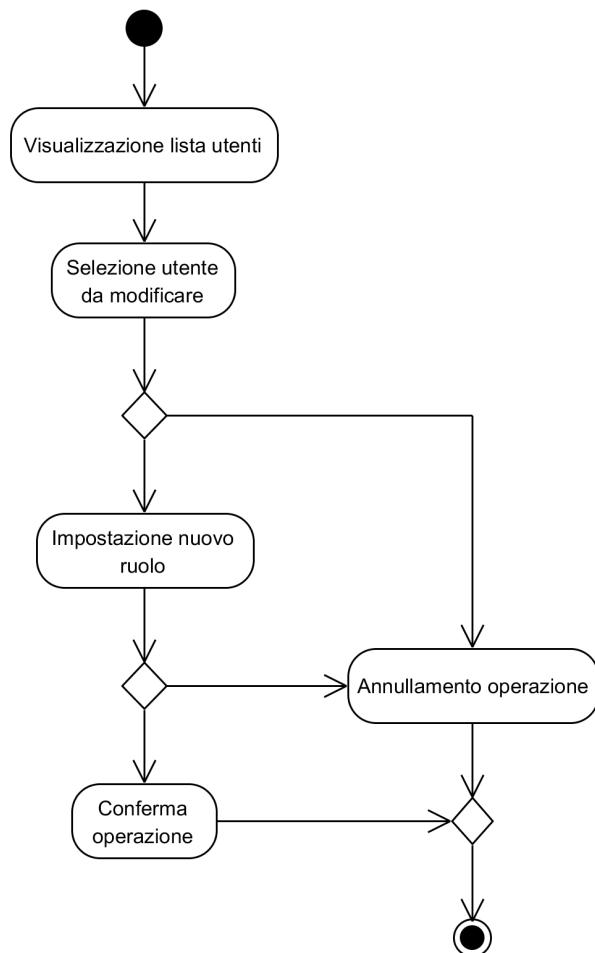


Figura 103: Attività di modifica del ruolo di un utente

Il Proprietario o un Amministratore può cambiare il ruolo di un utente già iscritto al sistema. Dopo aver visualizzato la lista degli utenti della propria azienda può, selezionandone uno, cambiare il suo ruolo. Premendo il pulsante apposito sarà infatti portato ad una pagina dalla quale potrà scegliere il nuovo ruolo. L'operazione può essere annullata in qualsiasi momento.

Eliminazione di un utente

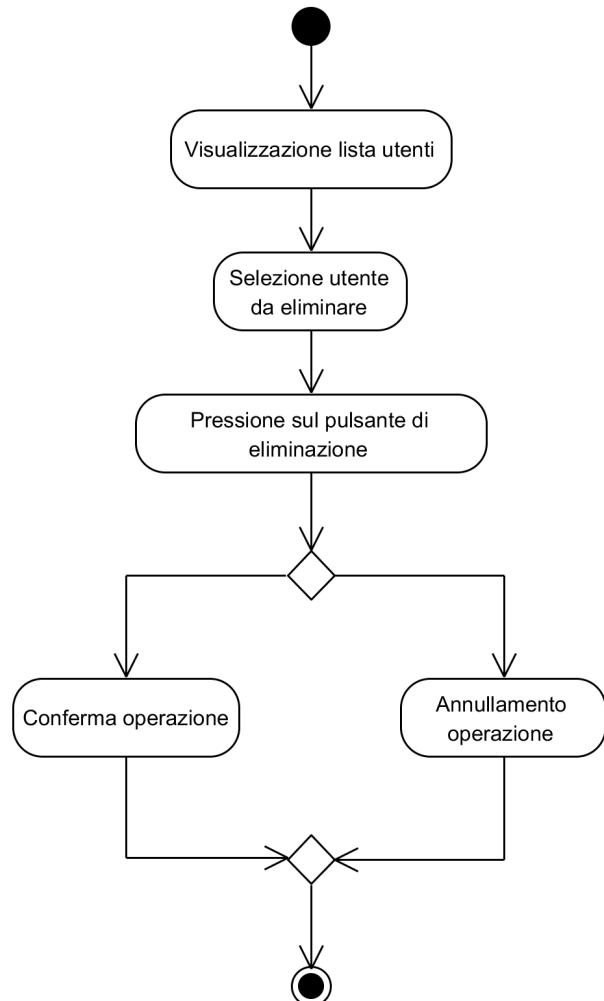


Figura 104: Attività di eliminazione di un utente

Il Proprietario o un Amministratore può eliminare un utente iscritto al sistema. Dopo aver visualizzato la lista degli utenti della propria azienda basterà, selezionandone uno, cliccare sul pulsante di eliminazione. Verrà quindi mostrata una Dialog Box con un'avvertenza riguardante l'azione che si sta per svolgere, da questa l'utente potrà confermare o annullare l'operazione.

Modifica dei permessi per un DSLIS

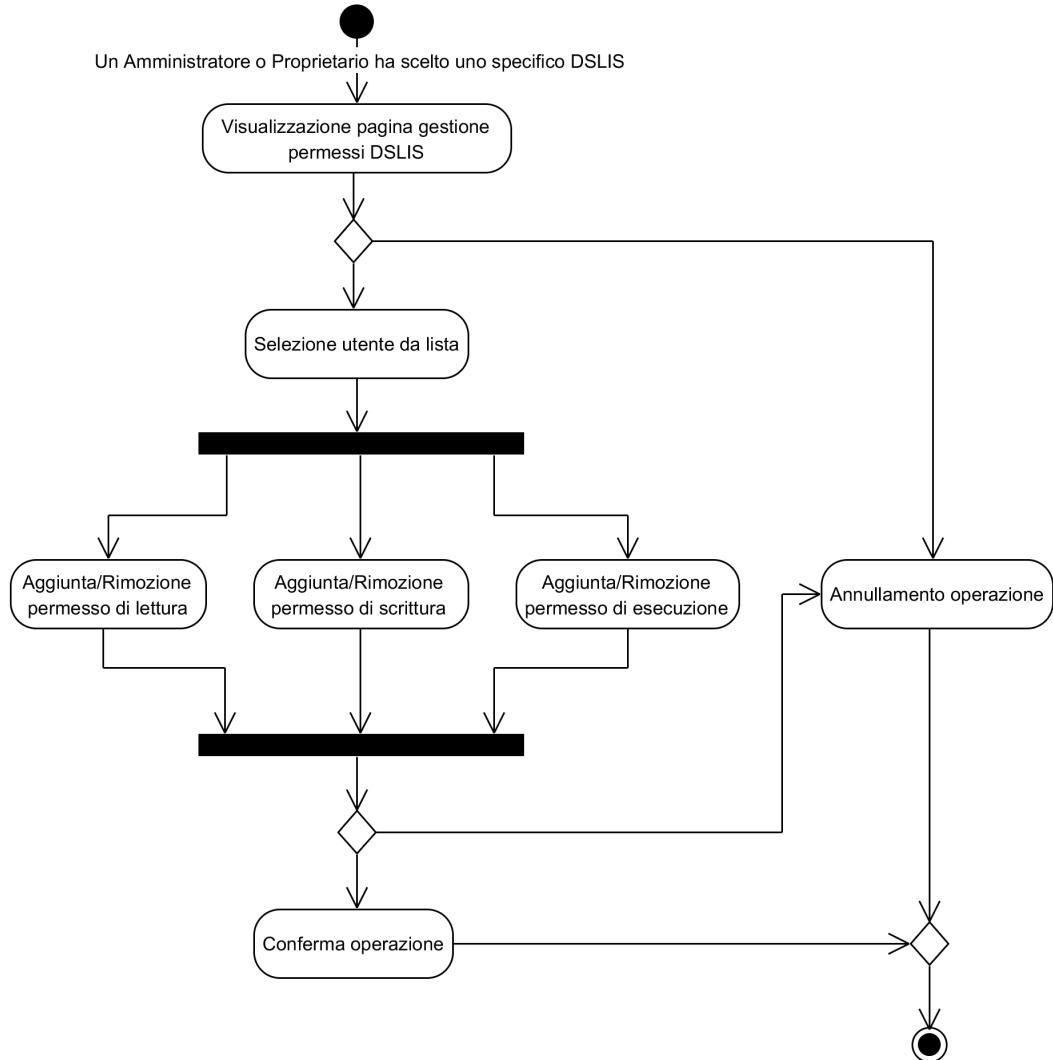


Figura 105: Attività di modifica dei permessi per specifici DSLIS

Il Proprietario o un Amministratore può cambiare i permessi che un utente può avere su uno specifico DSLIS, tra lettura, scrittura ed esecuzione. Una volta entrato nella pagina contenente la lista dei DSLIS, può sceglierne uno e cliccare sul pulsante di modifica dei permessi. A questo punto verrà mostrata la pagina con la lista degli utenti, di fianco ai quali saranno però visualizzate tre *checkbox*, relative ai tre tipi di permessi (una selezione corrisponderà all'attribuzione, mentre una deselezione corrisponderà alla deprivazione del relativo permesso). L'utente potrà poi premere il pulsante di conferma per salvare i cambiamenti. L'operazione può essere annullata in qualsiasi momento.

Modifica dei permessi di accesso al database

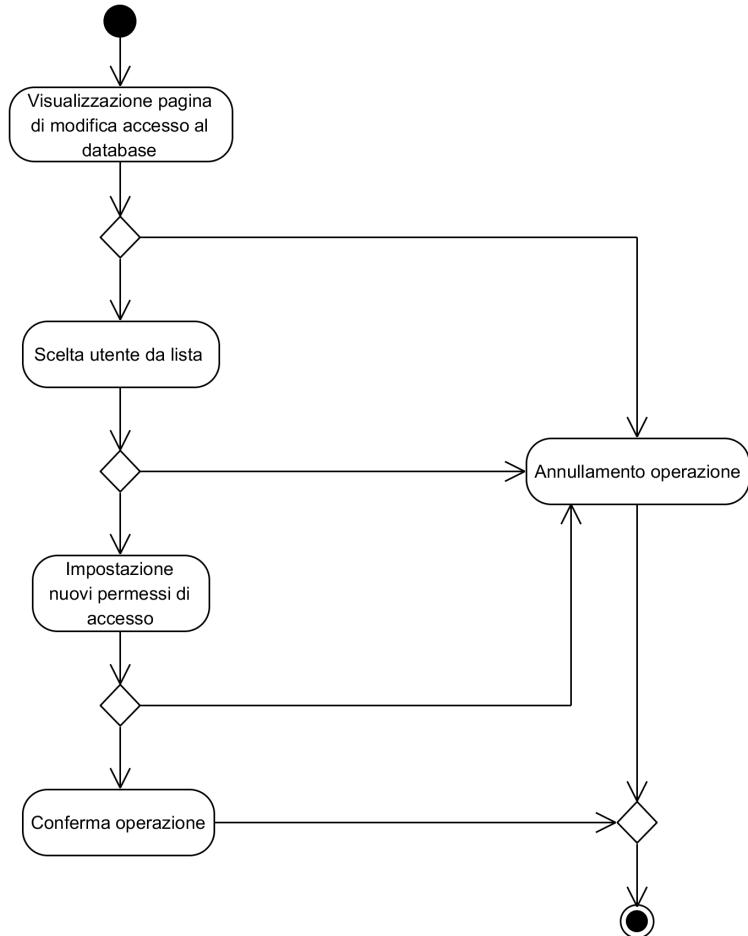


Figura 106: Attività di modifica dei permessi di accesso al database

Il Proprietario o un Amministratore può cambiare i permessi che un utente può avere sull'accesso al database. Una volta entrato nella pagina di modifica delle impostazioni di accesso, può scegliere un utente dalla lista degli utenti, di fianco ai quali sarà visualizzata una checkbox relativa al permesso di accesso (una selezione corrisponderà all'attribuzione, mentre una deselezione corrisponderà alla depravazione del relativo permesso). L'utente potrà poi premere il pulsante di conferma per salvare i cambiamenti. L'operazione può essere annullata in qualsiasi momento.

Gestione Aziende

Questa è la categoria di attività disponibili al Proprietario e al Super-Amministratore.

Connessione del database

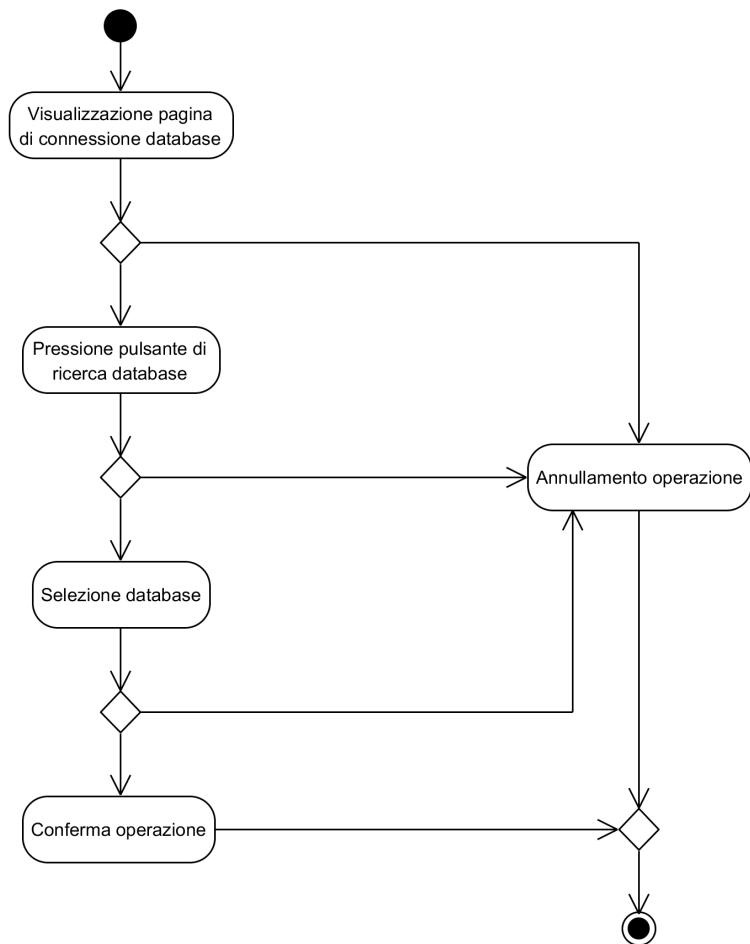


Figura 107: Attività di connessione del database aziendale

Il Proprietario può connettere l’azienda al proprio database (tipicamente una volta effettuata la registrazione). Dopo essere entrato nella pagina di connessione ad un data, dovrà cliccare sul pulsante di ricerca, il quale mostrerà la lista dei database disponibili per la sua azienda (normalmente uno solo). Successivamente gli basterà selezionare la base di dati e confermare attraverso l’apposito pulsante. L’operazione può essere annullata in qualsiasi momento.

Impersonificazione di un utente

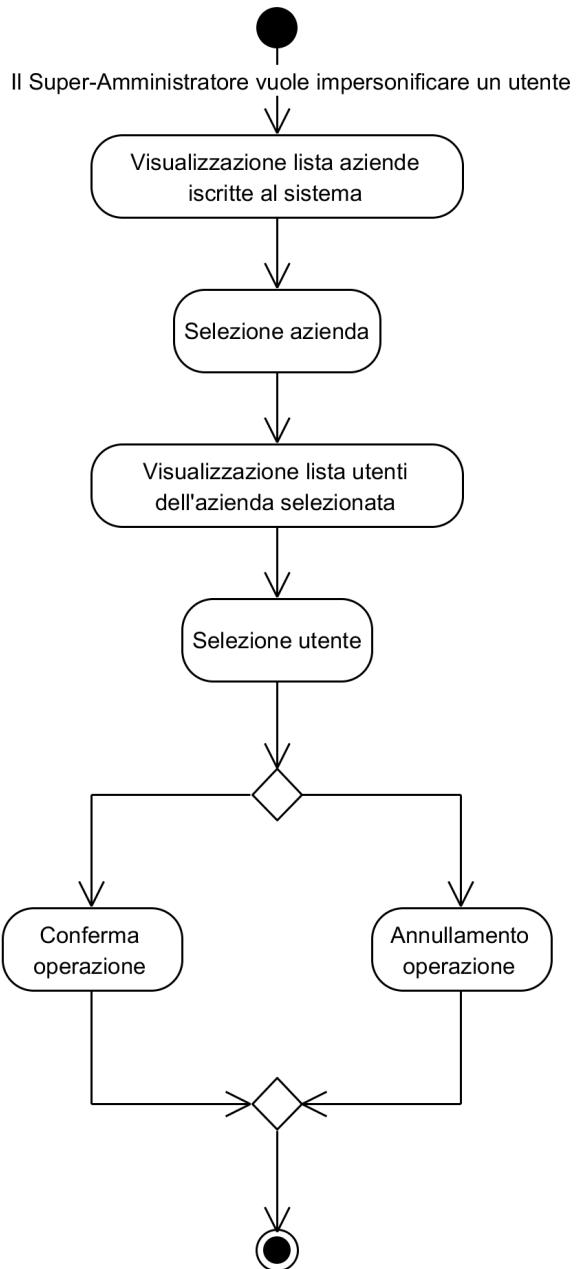


Figura 108: Attività di impersonificazione di un utente

Il Super-Amministratore può assumere nel sistema le sembianze di un qualsiasi utente iscritto, il quale può appartenere a qualsiasi azienda, per risolvere problemi di vario genere. Gli basterà entrare nella pagina di visualizzazione della lista di aziende e sceglierne una, per

poi scegliere un utente ad essa appartenente nella pagina successiva. Verrà quindi mostrata una Dialog Box con un'avvertenza riguardante l'azione che si sta per svolgere, da questa il Super-Amministratore potrà confermare o annullare l'operazione. In caso di conferma, comunque, perderà (momentaneamente) lo status di Super-Amministratore, per assumere l'identità (quindi anche il ruolo) dell'utente selezionato.

Descrizione Design Pattern

Design Pattern Architetturali

Flux

Flux è un pattern utilizzato per costruire applicazioni web client-server. Flux ha 3 componenti principali:

- **Dispatcher:** questo componente è unico nell'architettura e costituisce l'hub centrale che gestisce tutto il flusso di dati in un'applicazione flux; essenzialmente esso è il componente che si occupa di distribuire le azioni agli store;
- **Stores:** questo componente contiene lo stato logico dell'applicazione e il loro ruolo è quello di gestire lo stato delle componenti della view;
- **Views:** rappresenta l'interfaccia grafica dell'applicazione, riceve dati dallo store e interazioni con l'utente;
- **Actions:** è un oggetto Javascript che descrive l'azione che si vuole compiere.

Le conseguenze derivate dall'utilizzo di questo pattern sono:

- Permette di tracciare con facilità i cambiamenti che avvengono durante lo sviluppo;
- Semplifica individuazione e risoluzione di bug;
- Rende esplicito e facilmente comprensibile il flusso dei dati;
- Aiuta a produrre software scalabile.

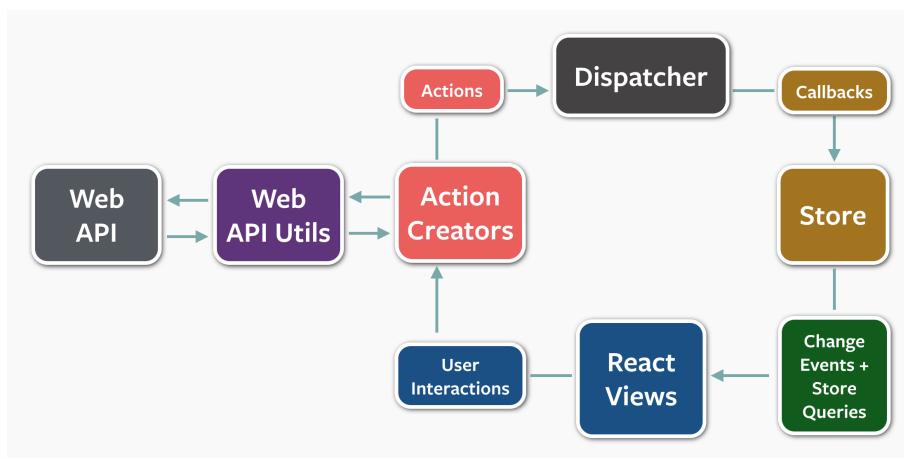


Figura 109: Struttura logica del pattern Flux

Design Pattern Creazionali

Singleton

Il Singleton è un design pattern che permette di avere un'unica istanza di una classe con un unico punto di accesso globale e noto; tale condizione trova risvolti pratici in svariate applicazioni. Per permettere l'implementazione di questo pattern è sufficiente che la classe interessata si occupi di tracciare la propria istanziazione e di bloccarla qualora sia già avvenuta almeno una volta. Il Singleton dovrebbe essere estensibile usando il subclassing quindi il client può utilizzarne l'estensione senza quindi modificarne il codice. Le conseguenze derivate dall'utilizzo di questo pattern sono:

- **Controllo d'accesso all'istanza:** la classe Singleton incapsula la sua unica istanza perciò è in grado di controllare quando e come i client vi accedono;
- **Pulizia del Namespace:** usando questo pattern si evita l'utilizzo delle variabili globali quindi evitando l'inquinamento dello spazio dei nomi;
- **Flessibilità:** in alcuni linguaggi come ad esempio il C++ è possibile ricondursi al singleton utilizzando delle operazioni sulle classi, in particolare la keyword static; in questo modo però è difficile controllarne il design e inoltre non è possibile l'utilizzo polimorfo delle sottoclassi che ridefiniscono perciò questo pattern consente di dare più flessibilità.

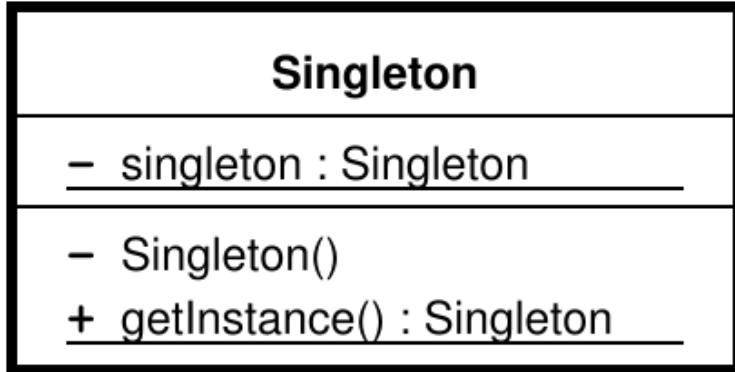


Figura 110: Struttura logica del pattern Singleton

Factory method

Questo pattern definisce un'interfaccia per la creazione di un oggetto, lasciando alle sottoclassi la decisione sul tipo che dovrà avere; inoltre consente di deferire l'istanziazione di una classe alle sottoclassi. Può venire utilizzato nei seguenti casi:

- Quando una classe non sa in anticipo di che tipo sarà l'oggetto da creare;
- Quando una classe vuole che le sue sottoclassi scelgano gli oggetti da creare;
- Quando le classi delegano la responsabilità a una o più classi di supporto e si vuole localizzare in un punto preciso la conoscenza di quale o quali classi di supporto.

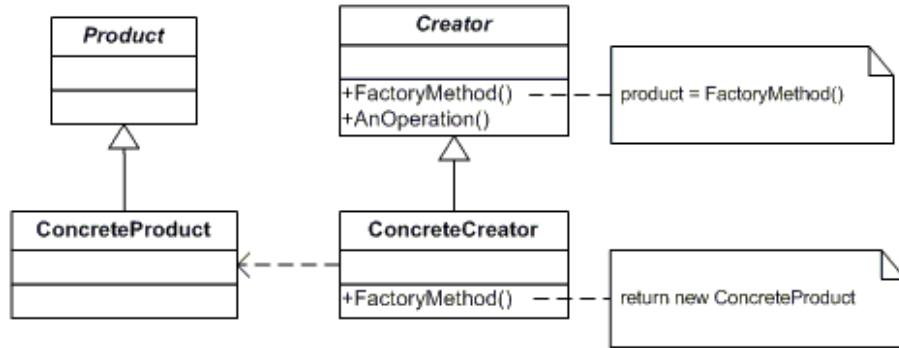


Figura 111: Struttura logica del pattern Factory Method

Abstract Factory

L'Abstract Factory fornisce un'interfaccia utile a creare delle famiglie di oggetti connessi o dipendenti tra loro, in modo che non ci sia necessità da parte dei client di specificare i nomi delle classi concrete all'interno del proprio codice; in questo modo si permette che un sistema sia indipendente dall'implementazione degli oggetti concreti e che il client, attraverso l'interfaccia, utilizzi diverse famiglie di prodotti. Il pattern può venire applicato nel caso in cui si voglia:

- un sistema indipendente da come gli oggetti vengono creati, composti e rappresentati;
- permettere la configurazione del sistema come scelta tra diverse famiglie di prodotti;
- che i prodotti, organizzati in famiglie, siano vincolati ad essere utilizzati con prodotti della stessa famiglia;
- fornire una libreria di classi mostrando solo le interfacce e nascondendo le implementazioni.

L'utilizzo del pattern Abstract Factory comporta poi una serie di conseguenze:

- **Isolamento:** la factory viene utilizzata solamente attraverso la sua interfaccia, per cui nei client non c'è traccia del codice per istanziare gli oggetti.
- **Coerenza nell'utilizzo dei prodotti:** Se i prodotti di una famiglia sono stati esplicitamente progettati per lavorare insieme, l'interfaccia AbstractFactory permette di rispettare questo vincolo;
- **Difficoltà nell'aggiungere supporto per le nuove tipologie di prodotti:** dato che AbstractFactory definisce tutte le tipologie di prodotti che è possibile istanziare, aggiungere una famiglia significa modificare l'interfaccia della factory; questa modifica si

ripercuerterà a cascata nelle factory concrete e in tutte le sottoclassi, rendendo laboriosa l'operazione;

- **Semplicità di modifica:** la factory viene istanziata una volta sola nel codice, quindi è sufficiente cambiare il tipo di factory istanziato in quel punto per utilizzare un diverso tipo di prodotti; la coerenza col resto del codice è assicurata dall'utilizzo delle interfacce astratte e non delle classi concrete.

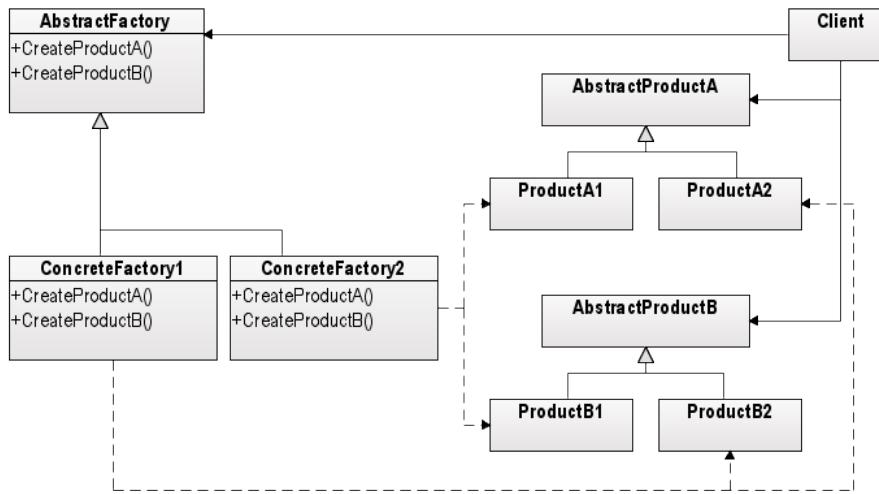


Figura 112: Struttura logica del pattern Abstract Factory

Design Pattern Strutturali

Facade

Facade definisce un'interfaccia di alto livello e unificata che rende il sottosistema più semplice da utilizzare. Può essere utilizzato nei seguenti casi:

- Quando si vuole fornire un'interfaccia semplice a un sottosistema complesso. La complessità dei sottosistemi tende ad aumentare con la loro evoluzione e molti pattern, quando applicati, portano a un aumento nel numero di classi piccole quindi a un aumento della complessità del sistema; l'utilizzo del Facade può fornire una vista semplice di base su un sottosistema che un client può utilizzare, rendendo trasparente la complessità reale di esso.
- Nel caso in cui ci siano molte dipendenze fra i client e le classi che implementano un'astrazione; introducendo un facade si disaccoppia il sottosistema dai client e dagli altri sottosistemi, promuovendo quindi la portabilità e l'indipendenza di sottosistemi;
- Quando si vogliono organizzare i sottosistemi in una struttura a livelli un facade può venire utilizzato per definire un punto di ingresso ad ogni livello; nel caso in cui i sottosistemi non siano indipendenti e le dipendenze esistenti possano essere semplificate facendo comunicare tra loro i sottosistemi soltanto attraverso i rispettivi facade.

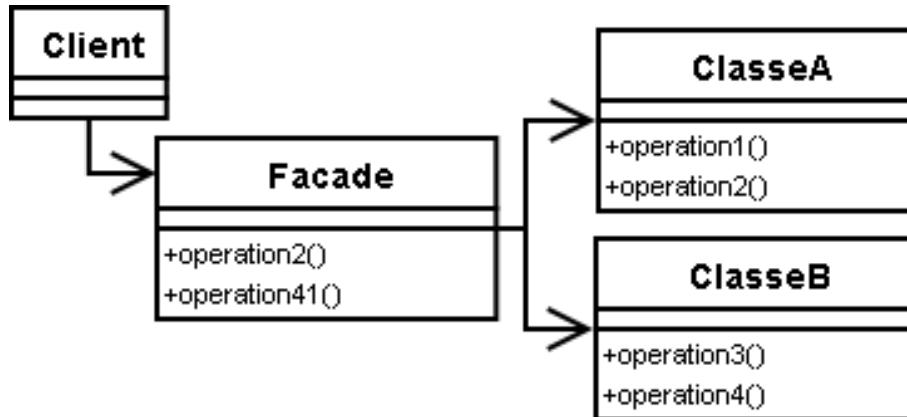


Figura 113: Struttura logica del pattern Facade

Design Pattern Comportamentali

Chain of Responsibility

Il Chain of Responsibility è un pattern comportamentale che permette di separare i sender delle richieste dai receiver delle stesse. In particolare, la richiesta attraversa una catena di oggetti per poi essere intercettata solo quando raggiunge il proprio gestore. Viene utilizzato in due casi:

- Quando non è possibile determinare staticamente il receiver;
- Quando l'insieme degli oggetti gestori cambia dinamicamente a runtime.

Le richieste vengono dette implicite poiché il sender non ha alcuna conoscenza sull'identità del ricevente; per permettere alla richiesta di attraversare la catena e per rimanere implicita, ogni receiver condivide un'interfaccia comune per gestire le richieste ed accedere al proprio successore. La gerarchia che vorrà inviare richieste dovrà avere una superclasse che dichiara un metodo handler generico. Le conseguenze derivate dall'utilizzo del pattern sono:

- **Riduzione dell'accoppiamento:** gli oggetti non sono a conoscenza di chi gestirà la richiesta ma sanno solo che verrà gestita in modo appropriato; inoltre non bisognerà manutenere i riferimenti a tutti i possibili riceventi;
- **Maggiore flessibilità:** in particolare nell'assegnamento delle responsabilità degli oggetti; è possibile distribuire le responsabilità tra gli oggetti a runtime modificandone la gerarchia mentre asaticamente è possibile usare il subclassing per specializzare i gestori;
- **Affidabilità:** non c'è garanzia che la request venga gestita, in particolare nel momento in cui la catena non è stata costruita correttamente.

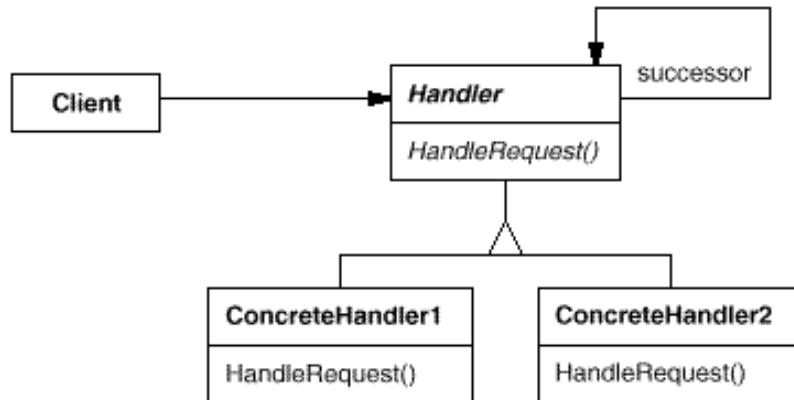


Figura 114: Struttura logica del pattern Chain of Responsibility

Template Method

Questo pattern permette di definire la struttura di un algoritmo lasciando alle sottoclassi il compito di implementarne alcuni passi come preferiscono; in questo modo si può ridefinire e personalizzare parte del comportamento nelle varie sottoclassi senza dover riscrivere più volte il codice in comune. Il template method viene applicato nei seguenti casi:

- Quando si vuole implementare la parte invariante di un algoritmo una volta sola e lasciare che le sottoclassi implementino il un comportamento diverso utilizzando come base la parte di codice in comune;
- Quando il comportamento comune di più classi può essere fattorizzato in una classe a parte per evitare di scrivere più volte lo stesso codice;
- per avere modo di controllare come le sottoclassi ereditano dalla superclasse, facendo in modo che i metodi template chiamino dei metodi "gancio" impostandoli come unici metodi sovrascrivibili.

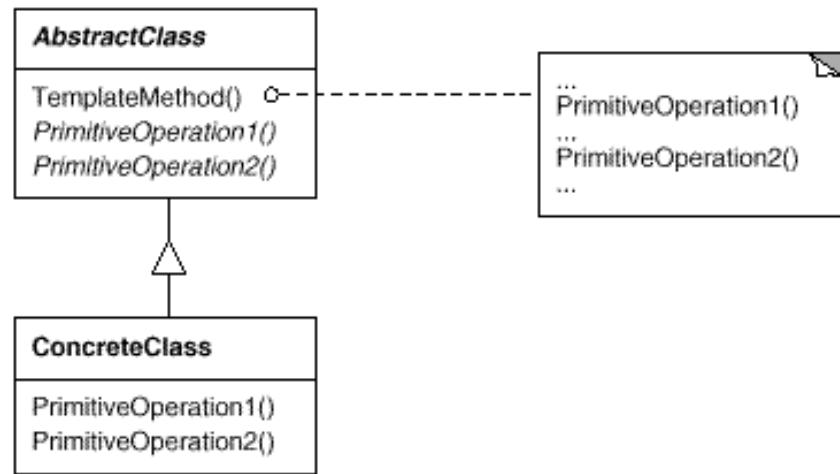


Figura 115: Struttura logica del pattern Template Method