MaaS - MongoDB as an admin Service

matrioska.io.go@gmail.com

Specifica Tecnica V3.0.0

Nome del documento	Specifica Tecnica
Versione del Documento	3.0.0
Data Creazione	05/05/2016
Redazione	D'Amico Roberto, Santi Guido
Verifica	Zamberlan Sebastiano
Approvazione	Berselli Marco
Uso	*Esterno*
Destinatari	*RedBabel*, Prof. Tullio Vardanega, Prof. Riccardo Cardin.

Registro delle modifiche

Versione	Autore	Ruolo	Data	Descrizione
3.0.0	Berselli Marco	Responsabile	10/09/2016	Approvazione del documento
2.2.0	Zamberlan Sebastiano	Verificatore	10/09/2016	Verifica Documento
2.2.0	D'Amico Roberto	Programmatore	09/09/2016	Miglioramento paragrafo 3.2
2.0.0	Santi Guido	Responsabile	15/08/2016	Approvazione
1.6.0	Berselli Marco	Verificatore	15/08/2016	Verifica finale
1.5.0	D'Amico Roberto	Analista	14/08/2016	Aggiornato tracciamento requisiti
1.4.1	D'Amico Roberto	Analista	14/08/2016	Aggiornati diagrammi
1.4.0	D'Amico Roberto	Analista	14/08/2016	Dettagliate le relazioni fra classi
1.3.0	D'Amico Roberto	Analista	14/08/2016	Eliminata sezione sull'architettura del framework loopback
1.2.4	Santi Guido	Progettista	12/08/2016	Aggiornate tecnologie utilizzate (sezione 2)
1.2.3	D'Amico Roberto	Analista	11/08/2016	Corretta formattazione delle tabelle di tracciamento
1.2.2	D'Amico Roberto	Analista	11/08/2016	Aggiunti pedici di glossario a pag.17
1.2.1	D'Amico Roberto	Analista	10/08/2016	Rivista formattazione delle utlime pagine
1.2.0	D'Amico Roberto	Analista	10/08/2016	Rimosso pattern MVC dal documento
1.1.0	D'Amico Roberto	Analista	9/08/2016	Inserito descrizione testuale API, rimosse immagini
1.0.0	D'Amico Roberto	Responsabile	8/06/2016	Approvazione
0.9.0	Berselli Marco	Verificatore	8/06/2016	Verifica finale
0.8.0	Petrov Andrei, D'Amico Roberto	Progettista	7/06/2016	Stesura delle sezioni relative ai Pattern di Progettazione ed Architetturali

Versione	Autore	Ruolo	Data	Descrizione
0.7.2	D'Amico Roberto	Progettista	7/06/2016	Correzione errori segnalati da Berselli Marco
0.7.1	Berselli Marco	Verificatore	6/06/2016	Verifica e segnalazione degli errori di battitura, punteggiatura e anomalie di tempi verbali
0.7.1	Berselli Marco	Verificatore	6/06/2016	Verifica semantica contenuti delle sezioni in stesura fino a questa data. Sono stati segnalate delle incongruenze di progettazione tra cliente e server
0.7.0	Santi Guido	Progettista	5/06/2016	Stesura delle stime di fattibilità e delle risorse
0.6.0	Petrov Andrei, Zamberlan Sebastiano	Progettista	3/06/2016	Stesura progettazione server
0.5.0	Santi Guido	Progettista	30/05/2016	Inserimento di contenuto riguardante il client
0.4.0	Petrov Andrei	Progettista	30/05/2016	Descrizione architettura
0.3.0	Petrov Andrei	Progettista	27/05/2016	Stesura Tecnologie Utilizzate
0.2.0	Petrov Andrei	Progettista	28/04/2016	Stesura Introduzione
0.1.0	Zamberlan Sebastiano	Progettista	28/04/2016	Creazione Struttura del Documento

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Il presente documento ha come obiettivo la descrizione dell'architettura ad alto livello per il prodotto. Vengono definite le componenti e la loro organizzazione attraverso la scelta oculata dei design pattern utilizzati. Inoltre vengono tracciate accuratamente le relazioni tra i requisiti e i package che li implementano.

1.2 Scopo del prodotto

L'obiettivo del *prodotto software*$_g$ è adattare il *framework*$_g$ *MongoDB*$_g$ as an admin Platform (*MaaP*$_g$), sviluppato dal gruppo SteakHolders durante l'attività accademica di *progetto*$_g$ nel corso di Ingegneria del Software dell'a.a. 2013/2014, per offrire il prodotto come *servizio web*$_g$.

1.3 Glossario

Con il presente viene consegnato anche un *"Glossario v3.0.0"* lo scopo di facilitare la lettura dei documenti formali. Ogni acronimo o termine tecnico accompagnato con una g a pedice sarà quindi integrato con ulteriori informazioni da ricercarsi nel suddetto Glossario.

1.4 Riferimenti

1.4.1 Normativi

- Norme di Progetto: *"Norme di Progetto v3.0.0"*;
- Capitolato d'appalto C4: MaaS "MongoDB as an admin Service": http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C4.pdf.

1.4.2 Informativi

- Capitolato d'appalto C1: MaaP: MongoDB as an admin Platform: http://www.math.unipd.it/~tullio/IS-1/2013/Progetto/C1.pdf;
- Software Engineering, Ian Sommerville, 9th edition (2010):
  - Part 1: Introduction to Software Engineering:
    - Chapter 5: System modeling;
    - Chapter 6: Architectural design.
- Software Architecture Patterns, Richards M. (2014):
  - Chapter 1: Layered Architecture.
- Web Development with MongoDB and NodeJS, M. Satheesh, J. Krol, B.J. D'mello, 2nd edition (2015):

  - Chapter 8: Creating a RESTful API;
  - Chapter 11: Single Page Applications with Popular Frontend Frameworks.
- Web Development with Node and Express, E. Brown (2014):
  - Chapter 11: Sending Email;
  - Chapter 12: Production Concerns;
  - Chapter 13: Routing.
- Express in Action *Writing, building, and testing Node.js applications*, E.M. Hahn (2016):
  - Part 1 Intro:
    * Chapter 3: Foundations of Express.
  - Part 2 Core:
    * Middleware;
    * Rounting.
  - Part 3 Express in Context:
    * Chapter 9: Testing Express applications;
    * Chapter 10: Security;
    * Chapter 11: Deployment: assets and Heroku;
    * Chapter 12: Best practices.
- F. Haupt, D. Karastoyanova, F. Leyman, B. Schroth, *A model-driven approach for REST compliant services*, IEEE International Conference on Web Services, 2014 ;
- S. Schreier, *Modeling RESTful applications*, Proceedings of the Second International Workshop on RESTful Design WSREST (15-21), 2011 ;
- H. Koziolek, *The SPOSAD architectural style for multi-tenant software applications*, Proceedings - 9th Working IEEE/IFIP Conference on Software Architecture, WICSA (320-327), 2011;
- S.J. Yang, P.C. Lai, J. Lin, *Design role-based multi-tenancy access control scheme for cloud services*, Proceedings - 2013 International Symposium on Biometrics and Security Technologies, ISBAST (273-279), 2013;
- *Service Oriented Front End Architecture*, D. Nelson (2013), http://www.sei.cmu.edu/library/assets/presentations/nelson-saturn2013.pdf;
- Hexagonal architecture http://alistair.cockburn.us/Hexagonal+architecture, Alistair Cockburn;
- React: https://facebook.github.io/react/;
- Redux: https://github.com/reactjs/redux;
- React-Router: https://github.com/reactjs/react-router.

2 Tecnologie Utilizzate

In questa sezione vengono descritte le tecnologie su cui si basa lo sviluppo del progetto. Per ognuna di esse, viene indicato l'ambito di utilizzo della tecnologia ed i vantaggi che ne derivano.

2.1 ECMAScript6

Come strumento di codifica il gruppo ha scelto di utilizzare la versione standard di Javascript, ECMAScript6 (ES6). Questa scelta è guidata dalle novità introdotte dal linguaggio e dal contesto applicativo del capitolato.

2.1.1 Vantaggi

Il linguaggio ES6 offre il supporto nativo delle senguenti *features*$_g$:

- Promesse, costrutto del linguaggio per il supporto dello stile di programmazione asincrono;
- Viene migliorato il problema dello *hoisting*$_g$ delle variabili grazie alla introduzione del concetto di *block scope*$_g$;
- Viene migliorata la leggibilità del codice tramite l'introduzione di nuovi costrutti sintattici come ad esempio le *arrow functions*.

2.1.2 Svantaggi

Essendo l'ultima versione uscita non possiede ancora lo stesso supporto offerto dai suoi predecessori.

2.2 HTML5

Per la strutturazione delle pagine web il gruppo ha deciso di utilizzare *HTML5*$_g$.

2.2.1 Vantaggi

- Linguaggio standard;
- Linguaggio flessibile, che funziona con diversi sistemi operativi, browser e dispositivi;
- Viene supportato dalla maggior parte dei browser;
- Richiede meno manutenzione avendo un solo codice sorgente.

2.2.2 Svantaggi

- Più lento rispetto ad una applicazione scritta in codice nativo;
- Meno stabile rispetto ad applicazioni scritte in codice nativo;
- Offre meno funzionalità offline.

2.3 CSS3

Per la formattazione delle pagine web lato client il gruppo ha deciso di utilizzare la versione 3 del linguaggio CSS.

2.3.1 Vantaggi

- Linguaggio standard;
- Viene supportato dalla maggior parte dei browser;
- Supporta a livello di linguaggio gli effetti grafici come le transazioni;
- È retro-compatibile.

2.3.2 Svantaggi

- Rallenta il caricamento della pagina, una pagina html senza css ci metterà meno a caricare;
- A differenza del normale CSS varie funzionalità non sono disponibili in tutti i browser;
- Gli effetti grafici non sono allo stesso livello di quelli realizzabili mediante javascript.

2.4 JSON

Per l' interscambio di dati tra client e server viene scelto JavaScript Object Notation (*JSON*$_g$).

2.4.1 Vantaggi

- Il client, il web server ed il database server parlano lo stesso linguaggio;
- JSON è leggero, rendendo la comunicazione e la lettura/scrittura più veloci;
- Facilità di utilizzo;
- JSON è nativo di Javascript, quindi non bisogna effettuare particolari operazioni per estrarre i dati.

2.4.2 Svantaggi

- JSON non è largamente utilizzato come altri linguaggi come ad esempio XML;
- Non è semplice come altri linguaggi;
- Non è possibile validare i dati attraverso uno schema/modello.

2.5 MongoDB

Il gruppo non ha avuto modo di decidere sulla tecnologia di persistenza dei dati. MongoDB è un vincolo tecnologico imposto dal Proponente.

MongoDB è un database Not only Search Query Language (NoSQL), Open Source, scalabile e performante di tipologia orientata al documento. Una tecnologia di persistenza dati semi-strutturati, una realtà sempre più comune nel contesto del Cloud Computing.

2.5.1 Vantaggi

- **Schema-less** : Una collezione può contenere documenti diversi. Numero di campi, contenuti e dimensioni del documento possono differire da un documento all'altro;
- La struttura di un singolo oggetto è chiara;
- **Profondità query** : MongoDB supporta le query dinamiche su documenti ;
- **Facilità di scale-out**: MongoDB è facile da scalare;
- Conversione/Mappatura di oggetti di applicazione al database non necessaria.

2.5.2 Svantaggi

I database NoSQL impongono un approccio diverso alla organizzazione dei dati. Un approccio che richiede al gruppo di mutare il proprio pensiero relazionale, dovuto a precedenti esperienze con database relazionali come ad esempio *MySQL*$_g$.

2.6 Node.js

Il gruppo non ha avuto modo di decidere sulla tecnologia di *run-time* server-side. Node.js è un vincolo tecnologico imposto dal Proponente. Il gruppo, tuttavia, ha avuto piena libertà sulla versione da utilizzare, versione maggiore a 4 *Long Term Support* (LTS). Il gruppo ha deciso di utilizzare la versione v6.1.0.

2.6.1 Vantaggi

- **Approccio asincrono**: l'approccio asincrono orientato ad eventi migliora la gestione delle richieste concorrenti;
- **Singolo linguaggio**: Node.js permette la scrittura del back-end in javascript, senza dover utilizzare nessun linguaggio di programmazione server-side;
- **Facilmente estendibile e supporta strumenti esterni**: oltre a permettere l'utilizzo di una vasta gamma di strumenti dedicati, essendo il codice open-source, è possibile estendere o modificare il codice di node in base alle proprie esigenze;
- **Facilita la realizzazione di applicazioni real-time**: quest'ambiente gestisce facilmente molte connessioni contemporanee, e tramite l'utilizzo di websockets semplifica e velocizza la comunicazione bidirezionale tra il server e il client.

2.6.2 Svantaggi

- **Mancanza di librerie robuste**: a differenza di molti linguaggi di programmazione moderni javascript non possiede delle librerie standard soddisfacenti;
- **Non adatto ad applicazioni larghe e complesse**: al momento node non supporta programmazione multi-thread, di conseguenza mette in coda le varie richieste e le esegue in maniera sequenziale, questo processo incrementa il tempo richiesto dall'applicazione per rispondere alle richieste dell'utente.

2.7 JWT

Per garantire la sicurezza dei dati e facilitare la comunicazione tra client e server il gruppo ha deciso di utilizzare JSON Web Token (JWT).

2.7.1 Vantaggi

- Autenticazione basata su token:
  - Rende la comunicazione insensibile allo stato;
  - Sicurezza extra, grazie agli algoritmi di criptografia dei dati multi livello.
- Il token è compatto ed auto contenuto;
- Facile da trasmettere tra client e server.

2.7.2 Svantaggi

Al momento non ne sono stati ricontratti rispetto ad altre tecnologie, in quanto offrono una sicurezza maggiore rispetto ad altre soluzioni e molti framework sono compatibili con questa tecnologia.

2.8 LoopBack

Come framework di alto livello e basato su Node.js, su consiglio del Proponente, il gruppo ha deciso di utilizzare il framework LoopBack.

2.8.1 Vantaggi

- Permette lo sviluppo di applicazioni web e API complesse;
- Offre funzionalità di generazione automatica di codice da riga di comando;
- Offre funzionalità di composizione visuale di API;
- Un ecosistema di *Connector*$_g$ molto vasta ed estende elegantemente il framework Express;
- *Swagger*$_g$ UI e generatore di codice automatico di API REST;
- Offre funzionalità di Autorizzazione mediante una matura implementazione di Access Control Lists.

2.8.2 Svantaggi

- Documentazione non soddisfacente;
- La difficoltà nell'apprendere inizialmente ad utilizzare questo framework.

2.9 React.js

Come libreria per l'implementazione dell'interfaccia grafica di MaaS il gruppo ha deciso di far uso di React.js assieme a Redux.

2.9.1 Vantaggi

- React.js pone l'attenzione sulla interfaccia utente e non sul framework;
- Semplicità di gestione dei dati tra le componenti grafiche dovuto al *single data binding*$_g$;
- Manipolazione virtuale del *DOM*$_g$.

2.9.2 Svantaggi

- Le conoscenze necessarie per l'utilizzo di questa libreria non sono minime;
- Non è semplice come utilizzare direttamente html5 o javascript;
- Non è un framework completo, è necessario utilizzare librerie esterne in vari ambiti.

2.9.3 Redux

Per agevolare lo sviluppo con React.js, il gruppo assieme al Proponente ha deciso di fare uso di Redux un framework sviluppato sulla base di React.js.

2.9.4 Svantaggi

- Redux è un contenitore di stato predicidibile;
- Si basa sui seguenti principi:
  - Lo stato della applicazione viene memorizzato in un'unico *store*$_g$;
  - Unico modo per effettuare una mutazione di stato è mediante l'emissione di una *action*$_g$;
  - Le modifiche vengono effettuate mediante *funzioni pure*$_g$ chiamate *reducers*$_g$.

2.9.5 Svantaggi

- Le conoscenze necessarie per l'utilizzo di questa libreria non sono minime;
- Non è semplice come utilizzare direttamente react.

3 Descrizione Architettura

3.1 Metodo e formalismo di specifica

La descrizione dell' architettura segue un approccio di decomposizione top-down; dove possibile il gruppo ha cercato di seguire anche un approccio bottom-up. In questa fase il gruppo ha svolto una attenta ricerca di librerie esterne da integrare. Successivamente vengono descritti i package, le componenti e le classi costituenti la struttura del prodotto software.

Per ogni entità software vengono delineate le relazioni di legame con altre entità, ai fini di delineare le dipendenze tra le componenti interni del sistema e dipendenze da componenti software esterne al prodotto software che il gruppo si impegna di sviluppare.

L'architettura generale non specifica l'intero insieme delle sottoclassi del sistema. Di conseguenza viene specificato lo scopo di una gerarchia e vengono individuate le relazioni con le varie componenti del sistema. L'individuazione di tutti gli attributi, metodi e sottoclassi viene rimandata alla *Progettazione di Dettaglio*.

I pattern di progettazione e i pattern architetturali, invece, vengono descritti in appendice. Come strumento di formalismo viene utilizzato il linguaggio grafico UML 2.x. Infine, i termini *componente* e *package* sono da intendere come sinonimi; e nei diagrammi delle componenti lato back-end vengono utilizzati dei colori con una precisa semantica:

- **Colore rosso**: indica una componente appartenente al framework Redux;
- **Colore verde**: indica una componente appartenente al framework LoopBack;
- **Colore arancio**: indica una componente esterna che viene integrata internamente al prodotto software.

3.2 Architettura generale

Nella fase di progettazione i progettisti hanno preso spunto dal progetto MaaP (MongoDB as an Admin Platform) nella forma del prodotto. La forma è delineata da tre componenti interconnesse che comunicano tra di loro. Queste componenti sono: Client, Web Server ed il Database Server. La relazione di connessione tra le componenti è la seguente: il Database server ed il Web server vengono interconnessi tramite connettori driver; il Client ed il Web server vengono interconnessi da una API REST-like.

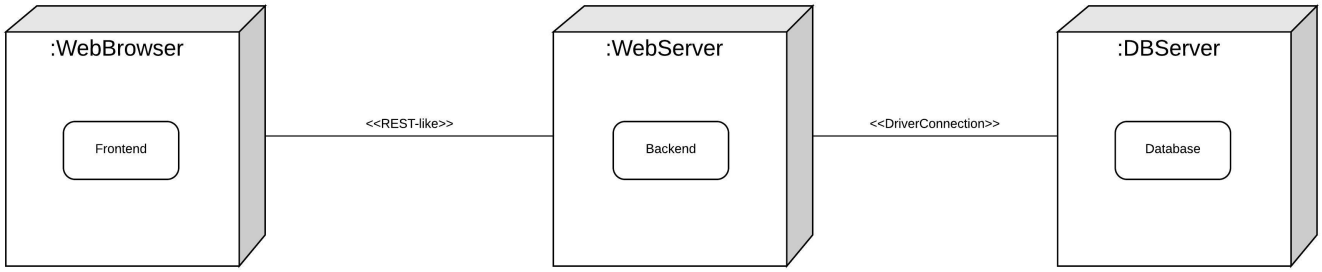

Fig 1: Diagramma di deployment

I framework scelti per il frontend sono: Redux e SuperAgent, che consentono con facilità di connettersi alle api lato server esposte da LoopBack.io, basato su Node.js. LoopBack.io a sua volta utilizza una base di dati non relazionale di tipo MongoDB.

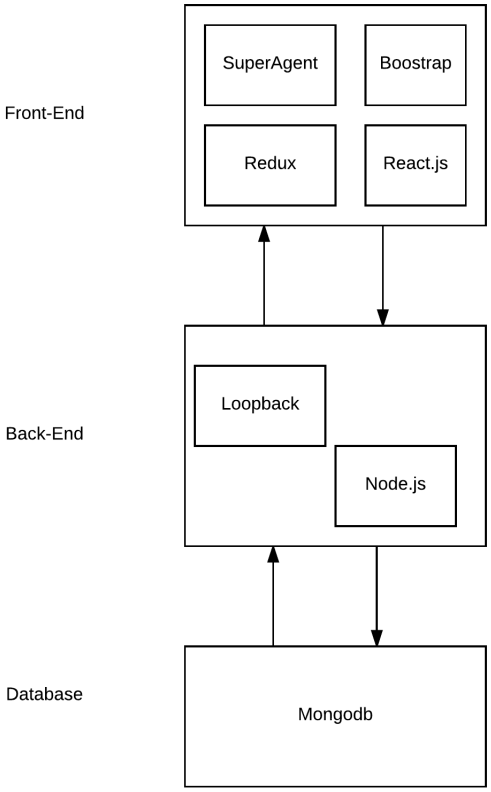

Fig 2: Diagramma dei framework utilizzati

3.3 Interfaccia REST-like

La comunicazione tra Client e Web server segue un protocollo REST-like. La comunicazione tra le due parti viene criptata da un token JWT. Un tale approccio permette di alleggerire il carico del server che altrimenti deve mantenere sempre in memoria lo stato del contesto dell'utente autenticato all'applicazione. Inoltre, il token JWT non deve essere memorizzato da alcuna parte internamente al sistema.

La scelta di utilizzo dello stile architetturale REST è dovuto a:

- Facile integrazione con la generazione automatica di API REST mediante il framework Loopback;
- Un insieme di principi e vincoli da rispettare ai fini di offrire un boun servizio web;
- Progettazione orientata ai dati.

I vincoli architetturali REST sono i seguenti:

- Modello di interazione client-server;

- Comunicazione trasparente nel contesto del tempo e dello spazio (Stateless Communication);
- Deve essere possibile memorizzare una risorsa in un contesto provvisorio;
- Organizzazione a livelli del sistema;
- Opzionalmente una richiesta può essere marcata da un codice da ritornare al client;
- Interfaccia di accesso alle risorse uniforme.

4 Definizione di alto livello dell'interfaccia REST

Ad ogni richiesta il server può rispondere con un messaggio di errore nel formato *JSON*$_g$ G e inviato con un codice *HTTP*$_g$ G della tipologia 4xx o 5xx. Il formato JSON G del messaggio di errore sarà:

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

Di seguito sono elencate le risorse *REST*$_g$ associate al tipo di metodo che è possibile richiedere su esse. Non vengono inclusi i permessi associati al tipo di accesso, in quanto le *API*$_g$ sono state generate automaticamente tramite modelli con lo strumento automatico **Swagger UI**; uno strumento incluso in loopback alla base dello strumento loopback-explorer. Tuttavia i tipi di permessi possibili sono:

- **Utente Guest**: Visualizza tramite link l'esecuzione di di una specifica DSL;
- **Utente Membro**: Utente comune MaaS ha accesso al proprio spazione di lavoro;
- **Utente Admin**: Utente con funzionalità di gestione dell'azienda e colui che ha la capacità di inviare inviti ad utenti esterni;
- **Utente Owner**: Utente con super permessi con il vincolo di non poter essere cancellato;
- **Super Admin**: Utente root con tutti i possibili permessi per la gestione dell'intera applicazione.

4.1 API esposte

Di seguito sono riportate tutte le API pubblicamente esposte dal server. Molte di queste richiedono tuttavia di essere autenticati per poter essere utilizzate. Ci si riserva la possibilità di variare questa lista in base alle necessità di sviluppo future.

4.1.1 GET /accounts/id/exists

Questa API è necessaria per conoscere preventivamente se un indirizzo e-mail è già stato utilizzato per la creazione di un account. In particolare questa informazione torna utile quando si intende registrare un'azienda a proprio nome o quando è necessario invitare i propri collaboratori.

```
{
" exist ": [ valore booleano ]
}
```

4.1.2 GET /accounts/confirm

Questa API viene utilizzata dai link recapitati agli utenti invitati. Un token di validazione viene infatti passato come parametro al fine di attivare l'account.

4.1.3 POST /accounts/login

Questa classica API accetta come parametro email e password dell'utente che intende autenticarsi. Questi dati vengono verificati e se corretti, si procede con la generazione di un token di autenticazione che accompagnerà l'utente durante l'uso del prodotto. Si fa nota che occorre accettare un invito tramite link spedito via mail per poter effettuare il login.

```
{
" _id ": [ access token dell'utente ] ,
" ttl ": [ tempo di validità ] ,
" accountId ": [ indirizzo email dell'utente ] ,
" createdAt ": [ data e ora di creazione utente]
}
```

4.1.4 POST /accounts/logut

API molto semplice che senza dover ricevere nessun parametro particolare rende invalido il token con il quale è stata chiamata, rendendo di fatto l'utente de-autenticato.

4.1.5 POST /companies

Questa API viene utilizzata per creare una nuova istanza aziendale. Sono necessari tre dati fondamentali:

- **Organization**: Il nome utilizzato come codice univoco, è necessario che sia unico all'interno di MaaS;
- **SubscribedAt**: Data di iscrizione a MaaS, utile per capire quando eliminare l'istanza aziendale dopo il limite di tempo senza utenti registrati;
- **OwnerId**: Indirizzo e-mail (nonché codice univoco) dell'utente proprietario.

4.1.6 GET /companies/id/exists

Questa API è necessaria per conoscere preventivamente se il nome di una nuova azienda è già stato utilizzato, infatti questa informazione torna utile durante la procedura di creazione di una nuova istanza aziendale.

4.1.7 GET /companies/id/databases

API che recupera tutti i database di proprietà dell'azienda e li restituisce sotto forma di JSON.

```
{
   [
     {
```

```
      " _id ": [ id del database ] ,
      " tag ": [ nome del database ]
      },
      {
      " _id ": [ id del database ] ,
      " tag ": [ nome del database ]
      }, ...
    ]
}
```

4.1.8 POST /companies/id/databases

Questa API viene utilizzata per creare un nuovo collegamento con un database non relazionale al quale il sistema attinge per eseguire le DSLI. Sono necessari due dati fondamentali:

- **Uri**: Stringa di connessione contente le credenziali necessarie per accedere in lettura al database prescelto;
- **Tag**: Un nome mnemonico e arbitrario per consentire agli utenti di identificare i vari database fra loro, in caso siano più di uno.

4.1.9 GET /companies/id/databases/fk

API che recupera tra i database aziendali uno di cui si conosce il codice univoco.

```
{
  " _id ": [ id del database ] ,
  " tag ": [ nome del database ]
}
```

4.1.10 PUT /companies/id/databases/fk

API che permette di aggiornare i dati di un database aziendale di cui si conosce il codice univoco.

4.1.11 DELETE /companies/id/databases/fk

API che permette di eliminare un database aziendale di cui si conosce il codice univoco. Si fa nota che va assegnato manualmente un nuovo database a tutte le DSLI legate ad un database eliminato, al fine di non renderle ineseguibile.

4.1.12 GET /companies/id/dsls

API che recupera tutte le DSLI di proprietà dell'azienda e le restituisce sotto forma di JSON.

```
[
  {
  " name ": [ nome della DSLI ],
  " lastModifiedDate ": [ ultima modifica della DSLI ],
```

```
  " permits ": [ permessi della DSLI ],
  " id ": [ id della DSLI ],
  " accountId ": [ email dell'autore ],
  " companyId ": [ company correlata ],
  " databaseId ": [ id del database correlato ],
  },
  {
  " name ": [ nome della DSLI ],
  " lastModifiedDate ": [ ultima modifica della DSLI ],
  " permits ": [ permessi della DSLI ],
  " id ": [ id della DSLI ],
  " accountId ": [ email dell'autore ],
  " companyId ": [ company correlata ],
  " databaseId ": [ id del database correlato ],
  },...
]
```

4.1.13 POST /companies/id/dsls

Questa API viene utilizzata per creare una nuova DSLI appartenente all'azienda. Sono necessari alcuni dati fondamentali:

- **Name**: Un nome mnemonico e arbitrario per consentire agli utenti di identificare in modo chiaro i dati trattati dalla DSLI;
- **Code**: Il codice scritto in DSL;
- **LastModifiedDate**: La data di ultima modifica, necessaria a fini di ordinamento;
- **AccountId**: Indirizzo e-mail (nonché codice univoco) dell'utente autore;
- **DatabaseId**: Il codice univoco del database aziendale al quale attingere per i dati necessari all'esecuzione della DSLI stessa.

4.1.14 GET /companies/id/dsls/fk

API che recupera tra le DSLI aziendali una di cui si conosce il codice univoco.

```
{
" name ": [ nome della DSLI ],
" lastModifiedDate ": [ ultima modifica della DSLI ],
" permits ": [ permessi della DSLI ],
" id ": [ id della DSLI ],
" accountId ": [ email dell'autore ],
" companyId ": [ company correlata ],
" databaseId ": [ id del database correlato ],
" DSLCode ": [ codice della DSLI ]
}
```

4.1.15 PUT /companies/id/dsls/fk

API che permette di aggiornare i dati di una DSLI aziendale di cui si conosce il codice univoco.

4.1.16 DELETE /companies/id/dsls/fk

API che permette di eliminare una DSLI aziendale di cui si conosce il codice univoco.

4.1.17 GET /companies/id/users

API che recupera tutti gli utenti appartenenti all'azienda e li restituisce sotto forma di JSON.

```
[
  {
  " email ": [ indirizzo email univoco ],
  " subscribedAt ": [ momento di iscrizione all'azienda ],
  " emailVerified ": [ booleano indica se l'invito è stato accettato ],
  " companyId ": [ compagnia dell'utente ],
  " dutyId ": [ ruolo dell'utente ]
  },
  {
  " email ": [ indirizzo email univoco ],
  " subscribedAt ": [ momento di iscrizione all'azienda ],
  " emailVerified ": [ booleano indica se l'invito è stato accettato ],
  " companyId ": [ compagnia dell'utente ],
  " dutyId ": [ ruolo dell'utente ]
  },...
]
```

4.1.18 POST /companies/id/users

Questa API viene utilizzata per creare un nuovo utente appartenente all'azienda. Si fa nota che invitare un utente implica la creazione di un istanza non confermata fino ad accettazione del rispettivo invito. Sono necessari alcuni dati fondamentali:

- **E-mail**: Indirizzo e-mail, utilizzato come codice univoco;
- **SubscribedAt**: Data di iscrizione a MaaS;
- **Password**: Parola segreta necessaria per autenticarsi, in principio viene generata una password provvisoria consegnata con l'invito;
- **EmailVerified**: Valore booleano che indica se l'invito è stato accettato o no;
- **DutyId**: Livello di accesso in ambito aziendale.

4.1.19 GET /companies/id/users/fk

API che recupera tra gli utenti aziendali uno di cui si conosce l'indirizzo e-mail.

```
{
" email ": [ indirizzo email univoco ],
" subscribedAt ": [ momento di iscrizione all'azienda ],
" emailVerified ": [ booleano indica se l'invito è stato accettato ],
" companyId ": [ compagnia dell'utente ],
" dutyId ": [ ruolo dell'utente ]
}
```

4.1.20 PUT /companies/id/users/fk

API che permette di aggiornare i dati di un utente di cui si conosce l'indirizzo e-mail.

4.1.21 DELETE /companies/id/users/fk

API che permette di eliminare un utente di cui si conosce l'indirizzo e-mail.

5 Stime di fattibilità e di bisogno risorse

Durante l'attività di progettazione il gruppo ha dedicato del tempo allo studio delle possibilità offerte dal framework loopback. Inoltre, sono state fatte ricerche di componenti esterne di ausilio per agevolare la futura implementazione del front-end e del back-end.

Le tecnologie adottate sono principalmente componenti non facenti parte del nucleo di loopback, ma sono componenti sviluppate dalla comunità di sviluppo open-source appositamente per loopback e successivamente integrate come plug-in.

Nell'ultimo anno LoopBack è maturato molto. Con ogni release, il framework diventa sempre più affidabile. In rete si trovano molti strumenti per iniziare a comprendere l'ecosistema loopback. Infatti, la repository loopback se da un lato è molto ricca di discussioni e forum di supporto, dall'altro offre un ampio spettro di esempi modello.

Per la parte di Front-End la situazione è molto simile. La comunità di sviluppo è molto attiva e i forum di supporto in continua crescita. React.js e Redux fanno emozionare molti sviluppatori di front-end. Le soluzioni che le tecnologie offrono ai problemi comuni di gestione delle pagine web sono innovative e questo permette un notevole incremento di nuovi entusiasti.

Quanto detto ha l'obiettivo di evidenziare che nonostante le tecnologie siano recenti il gruppo ha approfondito le proprie conoscenze via tutorial, punti di incontro orientati alla discussione come i forum, articoli, demo interne sia back-end sia front-end.

I membri del gruppo continuano ad approffondire le proprie conoscenze nel proprio dominio applicativo.

Per quanto riguarda la soddisfacibilità delle risorse di progettazione il gruppo è soddisfatto delle risorse fissate in momento pre-progettazione.

6 Componenti

In questa sezione vengono delineate le componenti di alto livello dell'applicazione MaaS (MongoDB as an Admin Service).

L'applicazione è organizzata in due componenti principali:

- **Front-End**:
  componente responsabile della gestione del client (lato utente) dell'applicazione;
- **Back-End**:
  componente responsabile della gestione del server dell'applicazione.

6.1 BackEnd

6.1.1 Informazioni sul package

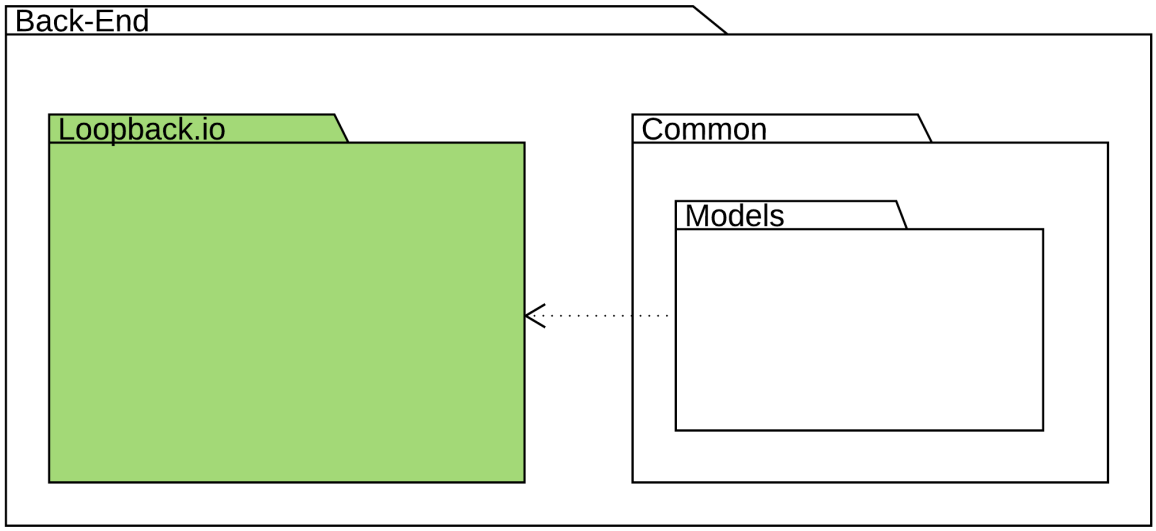

Fig 3: Componente BackEnd

6.1.2 Descrizione

Raccoglie tutti i package del back-end. Questa sezione è molto breve perchè Loopback.io è un framework molto complesso sul quale non abbiamo effettuato modifiche. Per altri dettagli su questa tecnologia visionare i riferimenti informativi.

6.1.3 Package contenuti

- BackEnd::Common::Models

6.2 BackEnd :: Common :: Models

6.2.1 Informazioni sul package

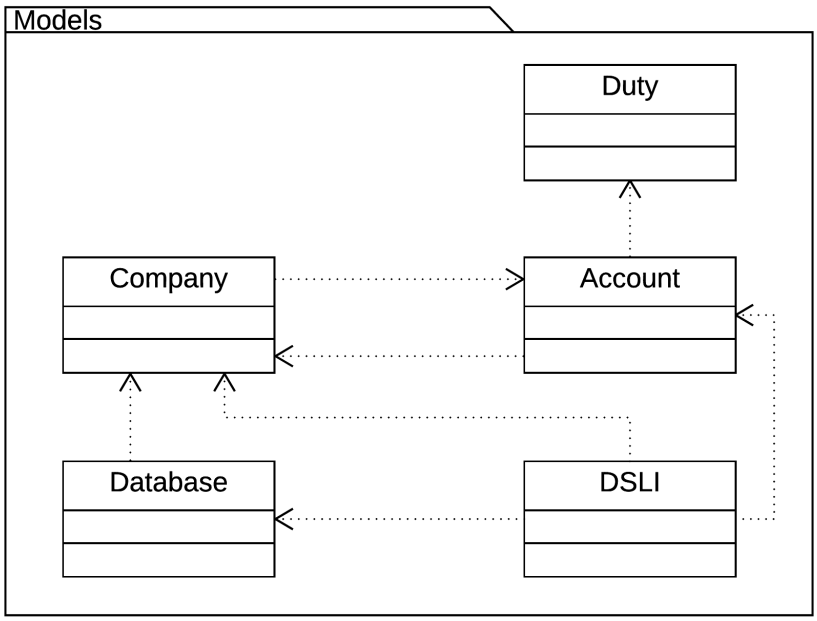

Fig 4: Componente BackEnd::Common::Models

6.2.2 Descrizione

Package che racchiude le classi che rappresentano la business logic dell'applicazione. Queste classi sono definite come modelli in loopback il quale concede l'utilizzo di alcuni modelli già implementati da poter estendere.

6.2.3 Classi

hypertargetBackEnd::Common::Models::Account

6.2.3.1 BackEnd :: Common :: Models :: Account

Descrizione

Classe che racchiude la business logic legata agli utenti. Implementa modello e schema definiti da Loopback.

Utilizzo

Il modello viene utilizzato sia per la rappresentazione di un utente nell'applicazione, sia per l'autenticazione nel sistema.

Relazioni con altre classi

Le relazioni seguenti sono di fatto riferimenti fra modelli di dati

BackEnd::Common::Models::Company: l'azienda di appartenenza.
BackEnd::Common::Models::Duty: il ruolo dell'utente.

6.2.3.2 BackEnd :: Common :: Models :: Company

Descrizione

Racchiude la business logic legata alle aziende. Implementa modello e schema definiti da Loopback.

Utilizzo

Il modello rappresenta un'azienda nel sistema.

Relazioni con altre classi

Le relazioni seguenti sono di fatto riferimenti fra modelli di dati

BackEnd::Common::Models::Account: l'utente proprietario.

6.2.3.3 BackEnd :: Common :: Models :: Database

Descrizione

Racchiude la business logic legata al collegamento con i database delle Aziende. Implementa modello e schema definiti da LoopBack.

Utilizzo

Il modello rappresenta la connessione ad un database aziendale di una Azienda. Contiene i dati per effettuare l'accesso al database e il riferimento alle collections definite su tale database, permettendo così all'utente di poter definire per ciascuna collection la possibilità di interagirvi da parte di tutti i membri della propria Azienda o solo degli Admin.

Relazioni con altre classi

Le relazioni seguenti sono di fatto riferimenti fra modelli di dati

BackEnd::Common::Models::Company: l'azienda di appartenenza.

6.2.3.4 BackEnd :: Common :: Models :: Duty

Descrizione

Racchiude la business logic legata ai ruoli di un utente. Implementa modello e schema definiti da Loopback.

Utilizzo

Il modello rappresenta dei ruoli che può avere un utente nel sistema.

6.2.3.5 BackEnd :: Common :: Models :: DSLI

Descrizione

Racchiude la business logic legata alle DSLI. Implementa modello e schema definiti da Loopback.

Utilizzo

Il modello rappresenta una DSLI nel sistema.

Relazioni con altre classi

Le relazioni seguenti sono di fatto riferimenti fra modelli di dati

BackEnd::Common::Models::Account: l'utente autore.
BackEnd::Common::Models::Company: l'azienda di appartenenza.
BackEnd::Common::Models::Database: il database a cui la DSLI fa riferimento.

6.3 FrontEnd

6.3.1 Informazioni sul package

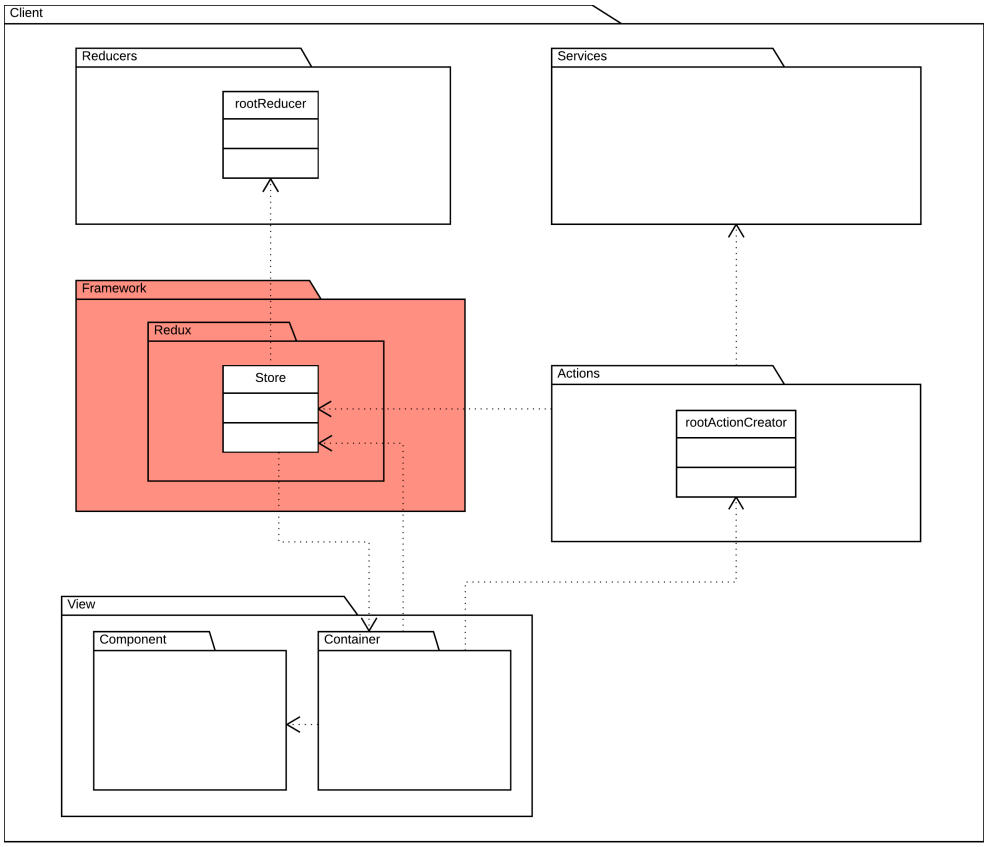

Fig 5: Componente FrontEnd

6.3.2 Descrizione

Raccoglie tutti i package del client.

6.3.3 Package contenuti

- FrontEnd::Actions
- FrontEnd::Reducers
- FrontEnd::Services
- FrontEnd::View

6.4 FrontEnd::Actions

6.4.1 Informazioni sul package

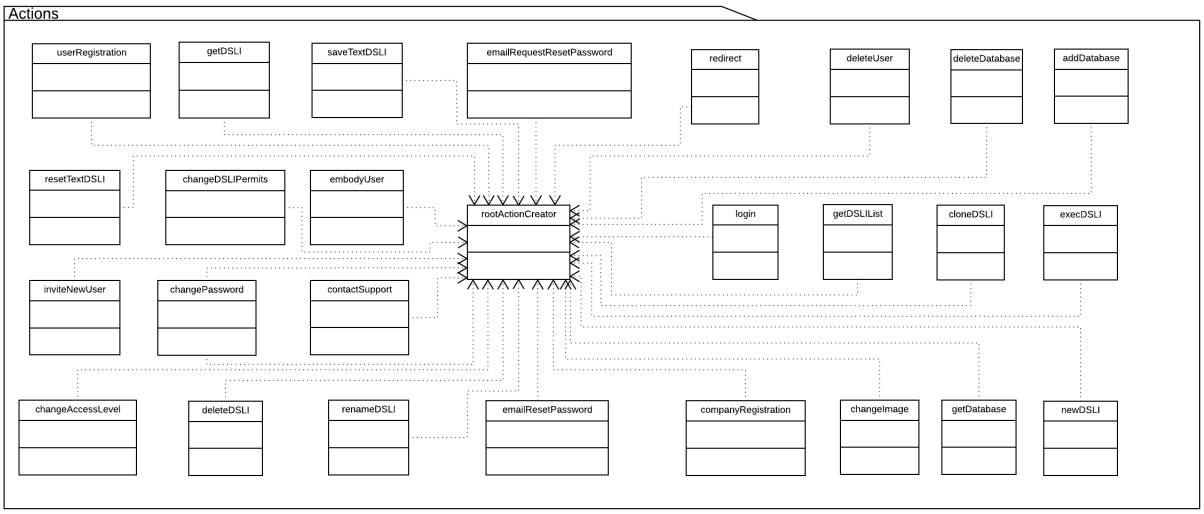

Fig 6: Componente FrontEnd::Actions

6.4.2 Descrizione

Il client comprende le Action creator, ovvero classi che hanno come scopo quello di interagire con il back-end dell'applicazione e al contempo generare le action, ovvero oggetti che contengono le eventuali informazioni ottenute dalle richieste al sistema.

6.4.3 Classi

6.4.3.1 FrontEnd :: Actions :: companyRegistration

Descrizione

Action creator che ha il compito di registrare un'azienda all'interno del sistema.

Utilizzo

Questa action creator ha il compito di verificare la validità dei dati inseriti e, se i dati sono validi, manda una richiesta di inserimento dati al sistema e genera una action contente le nuove informazioni, altrimenti genera una action di errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.
FrontEnd::Actions::userRegistration: se la registrazione dell'azienda va a buon fine essa termina chiamando la procedura per la registrazione del proprietario.

6.4.3.2 FrontEnd :: Actions :: userRegistration

Descrizione

Action creator che ha il compito di registrare un utente all'interno del sistema.

Utilizzo

Questa action creator ha il compito di verificare la validità dei dati inseriti e, se i dati sono validi, manda una richiesta di inserimento dati al sistema e genera una action contente le nuove informazioni, altrimenti genera una action di errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.3 FrontEnd :: Actions :: login

Descrizione

Action creator che ha il compito di autenticare un utente all'interno del sistema.

Utilizzo

Questa action creator ha il compito di verificare la validità dei dati inseriti mediante una richiesta al sistema e, se i dati sono validi, contatta il sistema per effettuare l'autenticazione e genera una action per descrivere l'operazione effettuata, altrimenti genera un'opportuna action d'errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.4 FrontEnd :: Actions :: emailRequestResetPassword

Descrizione

Action creator che ha il compito di iniziare la procedura di reset password.

Utilizzo

Questa action creator ha il compito di verificare la validità dei dati inseriti mediante una richiesta al sistema e, se i dati sono validi, effettua una chiamata alla classe ... del back-end e genera una action per descrivere l'operazione compiuta, altrimenti genera un'opportuna action d'errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.5 FrontEnd :: Actions :: emailResetPassword

Descrizione

Action creator che ha il compito di concludere la procedura di reset password.

Utilizzo

Questa action creator ha il compito di verificare la validità dei dati inseriti mediante una richiesta al sistema e, se i dati sono validi, manda una richiesta di aggiornamento dati al sistema e genera una action per descrivere l'operazione compiuta, altrimenti genera un'opportuna action d'errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.6 FrontEnd :: Actions :: redirect

Descrizione

Action creator che ha il compito di far reindirizzare la pagina.

Utilizzo

Questa action creator ha il compito di generare una action contente la pagina a cui si deve venire reindirizzati.

6.4.3.7 FrontEnd :: Actions :: changePassword

Descrizione

Action creator che ha il compito di far modificare la password dell'utente.

Utilizzo

Questa action creator ha il compito di verificare la validità della password precedente fornita e, in caso positivo, manda una richiesta di aggiornamento dati al sistema e genera una action contente la nuova password inserita nel sistema, altrimenti generare una action di errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.8 FrontEnd :: Actions :: changeImage

Descrizione

Action creator che ha il compito di far modificare l'immagine dell'utente.

Utilizzo

Questa action creator ha il compito di verificare la validità dell'immagine fornita e, in caso positivo, manda una richiesta di aggiornamento dati al sistema e genera una action contente la nuova immagine da inserire nel sistema, altrimenti genera una action di errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.9 FrontEnd :: Actions :: execDSLI

Descrizione

Action creator che ha il compito di far eseguire il codice della DSLI.

Utilizzo

Questa action creator ha il compito di effettuare una chiamata al sistema per ottenere l'esito dell'esecuzione della DSLI selezionata. In caso non ci siano problemi nei dati ottenuti la classe genera una action contente tali dati, altrimenti genera una action di errore.

Relazioni con altre classi

FrontEnd::Actions::redirect
FrontEnd::Services::PageBuilder

6.4.3.10 FrontEnd :: Actions :: newDSLI

Descrizione

Action creator che ha il compito di far aggiungere una nuova DSLI.

Utilizzo

Questa action creator ha il compito di mandare una richiesta per l'inserimento della nuova DSLI al sistema e generare una action contente il nome della nuova DSLI.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.11 FrontEnd :: Actions :: deleteDSLI

Descrizione

Action creator che ha il compito di far cancellare una DSLI dal sistema.

Utilizzo

Questa action creator ha il compito di mandare una richiesta di rimozione per la DSLI selezionata al sistema e generare una action che descrive l'operazione effettuata.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.12 FrontEnd :: Actions :: cloneDSLI

Descrizione

Action creator che ha il compito di far clonare una DSLI.

Utilizzo

Questa action creator ha il compito di mandare una richiesta per l'inserimento della nuova DSLI al sistema e generare una action contente il nome della nuova DSLI.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.13 FrontEnd :: Actions :: saveTextDSLI

Descrizione

Action creator che ha il compito di far salvare il testo modificato di una DSLI.

Utilizzo

Questa action creator ha il compito di mandare una richiesta di aggiornamento dati al sistema e genera una action contente le informazioni modificate.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.14 FrontEnd :: Actions :: inviteNewUser

Descrizione

Action creator che ha il compito di far inviare una email di invito.

Utilizzo

Questa action creator ha il compito di effettuare una chiamata al server remoto e generare una action che descrive l'operazione effettuata.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.15 FrontEnd :: Actions :: changeAccessLevel

Descrizione

Action creator che ha il compito di far modificare il livello d'accesso di un utente.

Utilizzo

Questa action creator ha il compito di mandare una richiesta di aggiornamento per il livello d'accesso dell'utente al sistema e genera una action contente le informazioni modificate.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.16 FrontEnd :: Actions :: deleteUser

Descrizione

Action creator che ha il compito di far eliminare un utente dal sistema.

Utilizzo

Questa action creator ha il compito di mandare una richiesta di rimozione per l'utente selezionato al sistema e generare una action che descrive l'operazione effettuata.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.17 FrontEnd :: Actions :: changeDSLIPermits

Descrizione

Action creator che ha il compito di far modificare i diritti d'accesso ad una DSLI.

Utilizzo

Questa action creator ha il compito di mandare una richiesta di aggiornamento per i diritti d'accesso alla DSLI selezionata al sistema e genera una action contente le informazioni modificate.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.18 FrontEnd :: Actions :: embodyUser

Descrizione

Action creator che ha il compito di far impersonare al superadmin un utente di una data azienda all'interno del sistema.

Utilizzo

Questa action creator ha il compito di verificare la validità dei dati inseriti mediante una richiesta al sistema e, se i dati sono validi, contatta il sistema per effettuare l'impersonificazione e genera una action per descrivere l'operazione effettuata, altrimenti genera un'opportuna action d'errore.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.19 FrontEnd :: Actions :: contactSupport

Descrizione

Action creator che ha il compito di recapitare al supporto del sito un messaggio dell'utente.

Utilizzo

Questa action creator ha il compito di effettuare una chiamata al server remoto e generare una action che descrive l'operazione effettuata.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.20 FrontEnd :: Actions :: rootAction

Descrizione

Classe che ha come compito quello di dare un semplice accesso ai vari action creators.

Utilizzo

Questa classe mette a disposizione i vari action creator importandoli al proprio interno e contemporaneamente esportandoli verso l'esterno.

Relazioni con altre classi

Le relazioni seguenti esistono con il puro scopo di centralizzare l'accesso ad esse. Non vengono utilizzate internamente a questa classe.

FrontEnd::Actions::companyRegistration
FrontEnd::Actions::userRegistration
FrontEnd::Actions::login
FrontEnd::Actions::emailRequestResetPassword
FrontEnd::Actions::emailResetPassword
FrontEnd::Actions::redirect
FrontEnd::Actions::changePassword
FrontEnd::Actions::changeImage
FrontEnd::Actions::execDSLI
FrontEnd::Actions::newDSLI
FrontEnd::Actions::deleteDSLI
FrontEnd::Actions::cloneDSLI
FrontEnd::Actions::renameDSLI
FrontEnd::Actions::saveTextDSLI
FrontEnd::Actions::resetTextDSLI
FrontEnd::Actions::inviteNewUser
FrontEnd::Actions::changeAccessLevel
FrontEnd::Actions::deleteUser

FrontEnd::Actions::changeDSLIPermits
FrontEnd::Actions::embodyUser
FrontEnd::Actions::contactSupport
FrontEnd::Actions::getDSLI
FrontEnd::Actions::getDSLIList

6.4.3.21 FrontEnd :: Actions :: getDSLI

Descrizione

Action creator che ha il compito di prendere le informazioni di una DSLI dal sistema.

Utilizzo

Questa action creator ha il compito di mandare una richiesta all'API per ottenere le informazioni complete relative ad una DSLI determinata tramite il proprio id.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.22 FrontEnd :: Actions :: getDSLIList

Descrizione

Action creator che ha il compito di prendere dal sistema la lista delle DSLI accessibili dall'utente.

Utilizzo

Questo action creator ha il compito di mandare una richiesta all'API per ottenere la lista delle DSLI personali dell'utente e di quelle pubbliche all'interno dell'azienda.

Relazioni con altre classi

FrontEnd::Actions::redirect: redirect viene utilizzato per permettere a questa action creator di reindirizzare l'utente ad operazione avvenuta.

6.4.3.23 FrontEnd :: Actions :: addDatabase

Descrizione

Action creator che ha il compito di memorizzare i database di un'azienda all'interno del sistema.

Utilizzo

Questa action creator ha il compito di mandare una richiesta per l'inserimento delle stringhe d'accesso ai database dell'azienda al sistema e genera una action contente le nuove informazioni, altrimenti genera una action di errore.

6.4.3.24 FrontEnd :: Actions :: getDatabase

Descrizione

Action creator che ha il compito di recuperare l'elenco dei database aziendali necessari all'esecuzione delle DSLI.

Utilizzo

Questo action creator ha il compito di mandare una richiesta all'API per ottenere la lista dei database aziendali.

6.4.3.25 FrontEnd :: Actions :: deleteData

Descrizione

Action creator che ha il compito di eliminare il collegamento ad un database aziendale.

Utilizzo

Questa action creator ha il compito di mandare una richiesta di rimozione per il database selezionato al sistema e generare una action che descrive l'operazione effettuata.

6.5 FrontEnd::Reducers

6.5.1 Informazioni sul package

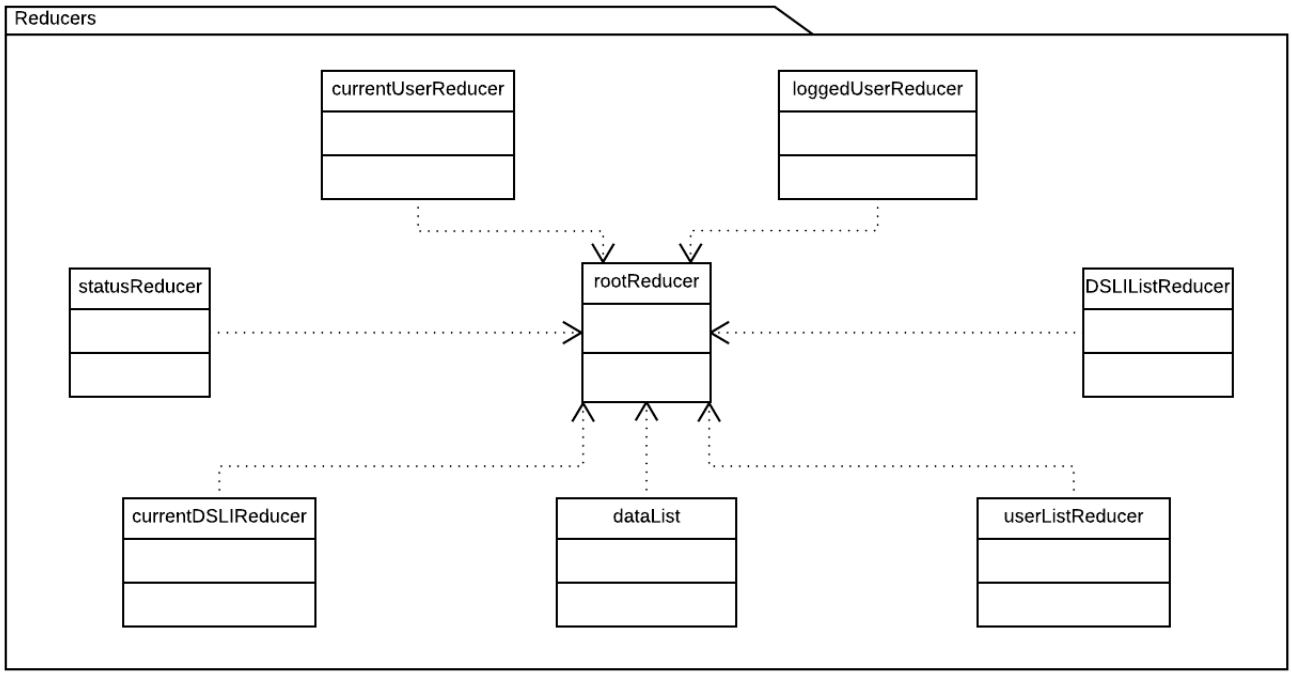

Fig 7: Componente FrontEnd::Reducers

6.5.2 Descrizione

Mentre le action descrivono ciò che è accaduto, il compito dei reducer è quello di cambiare lo stato del sistema in base all'action che hanno ricevuto. I reducer sono delle funzioni pure che prendono in input lo stato del sistema e una action e generano un nuovo stato.

6.5.3 Classi

6.5.3.1 FrontEnd :: Reducers :: rootReducer

Descrizione

Reducer che gestisce l'accesso agli altri reducers.

Utilizzo

Questo reducer gestisce le action ricevute reindirizzandole all'apposito reducer.

Relazioni con altre classi

FrontEnd::Reducers::statusReducer
FrontEnd::Reducers::loggedUserReducer
FrontEnd::Reducers::currentUserReducer
FrontEnd::Reducers::currentDSLIReducer

FrontEnd::Reducers::DSLIListReducer
FrontEnd::Reducers::userListReducer

6.5.3.2 FrontEnd :: Reducers :: statusReducer

Descrizione

Reducer che gestisce lo status del sistema.

Utilizzo

Questo reducer gestisce le informazioni, ricevute tramite apposite action, relative alle operazioni avvenute nel sistema e aggiorna lo status in base all'azione effettuata.

6.5.3.3 FrontEnd :: Reducers :: loggedUser

Descrizione

Reducer che gestisce l'autenticazione al sistema e le informazioni dell'utente autenticato.

Utilizzo

Questo reducer gestisce le informazioni, ricevute tramite apposite action, relative all'autenticazione e alle informazioni dell'utente autenticato e genera un nuovo stato aggiornando il precedente con i dati ottenuti.

6.5.3.4 FrontEnd :: Reducers :: currentDSLI

Descrizione

Reducer che gestisce la DSLI attualmente selezionata.

Utilizzo

Questo reducer gestisce le informazioni, ricevute tramite apposite action, relative alla DSLI correntemente selezionata permettendo di cambiare la DSLI selezionata e modificare le informazioni di tale DSLI.

6.5.3.5 FrontEnd :: Reducers :: currentUser

Descrizione

Reducer che gestisce l'utente attualmente selezionato.

Utilizzo

Questo reducer gestisce le informazioni, ricevute tramite apposite action, relative all'utente selezionato permettendo di cambiare l'utente correntemente selezionato e modificare le informazioni di tale utente.

6.5.3.6 FrontEnd :: Reducers :: DSLIList

Descrizione

Reducer che gestisce la lista delle DSLI disponibili per l'utente.

Utilizzo

Questo reducer gestisce le informazioni, ricevute tramite apposite action, relative alla lista delle DSLI disponibili permettendo di aggiornarla.

6.5.3.7 FrontEnd :: Reducers :: userList

Descrizione

Reducer che gestisce la lista degli utenti registrati all'interno dell'azienda.

Utilizzo

Questo reducer gestisce le informazioni, ricevute tramite apposite action, relative alla lista degli utenti registrati permettendo di aggiornarla.

6.5.3.8 FrontEnd :: Reducers :: dataList

Descrizione

Reducer che gestisce l'elenco di database non relazionali appartenenti all'azienda e le informazioni per accederci.

Utilizzo

Questo reducer gestisce le informazioni, ricevute tramite apposite action, relative ai database aziendali e genera un nuovo stato aggiornando il precedente con i dati ottenuti.

6.6 FrontEnd::Services

6.6.1 Informazioni sul package

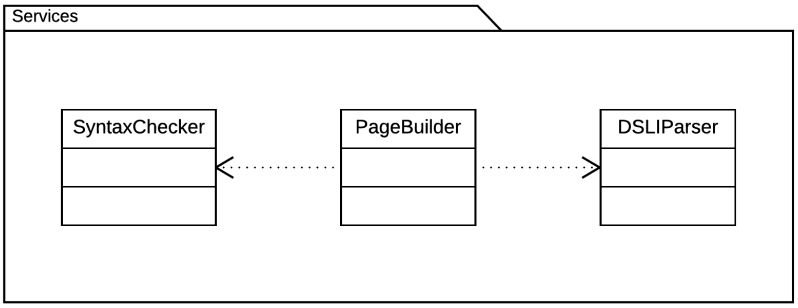

Fig 8: Componente FrontEnd::Services

6.6.2 Descrizione

Questo pacchetto contiene per la maggior parte classi statiche con funzioni utilizzabili quando è necessario. Data questa capacità di eseguire calcoli e controlli su richiesta abbiamo definito il nome Services.

6.6.3 Classi

6.7 Services

6.7.0.1 FrontEnd :: Services :: PageBuilder

Descrizione

Questa classe contiene metodi e procedure necessarie al fine di comporre la pagina web che dovrà mostrare i risultati delle DSLI.

Utilizzo

I metodi statici di questa classe accetteranno in input i file json creati dal server remoto e produrranno un componente che FrontEnd::View::Containers::ViewDSLI utilizzerà.

6.7.0.2 FrontEnd :: Services :: SyntaxChecker

Descrizione

Questa classe contiene metodi e procedure necessarie al fine di verificare che la sintassi di una DSLI sia esatta.

Utilizzo

I metodi statici di questa classe accetteranno in input una stringa e restituiranno un booleano e una stringa di eventuali errori.

6.7.0.3 FrontEnd :: Services :: DSLIParser

Descrizione

Questa classe contiene metodi e procedure necessarie al fine di trasformare una DSLI in query NoSQL.

Utilizzo

I metodi statici di questa classe accetteranno in input una stringa e restituiranno un json pronto per essere inviato al server remoto.

Relazioni con altre classi

FrontEnd::Serivces::SyntaxChecker: utilizzato per controllare che la sintassi sia effettivamente corretta.

6.8 FrontEnd::View

6.8.1 Informazioni sul package

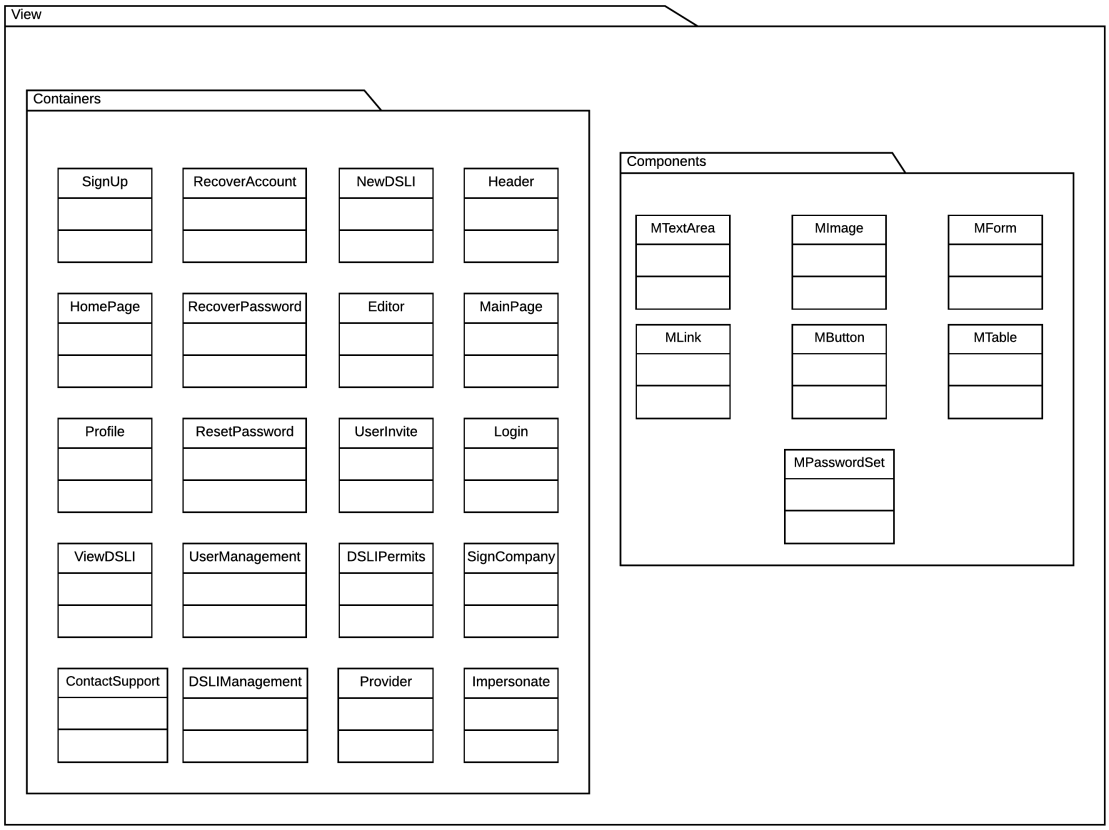

Fig 9: Componente FrontEnd::View

6.8.2 Descrizione

Il pacchetto View si divide in due parti, componenti semplici e contenitori. I contenitori possiedono della logica e sono in grado di svolgere il proprio compito con l'ausilio delle componenti semplici che lo compongono.

6.8.3 Package contenuti

- FrontEnd::View::Containers
- FrontEnd::View::Components

6.9 FrontEnd::View::Containers

6.9.1 Descrizione

Pacchetto dei Containers, componenti con logica integrata.

6.9.2 Classi

6.9.2.1 FrontEnd :: View :: Containers :: Provider

Descrizione

Questa classe è un container per tutti i componenti dell'applicazione, avrà quindi la funzione di radice della gerarchia di viste e può essere intesa come il foglio bianco della Single-Page Application.

Utilizzo

Il suo scopo principale è quello di possedere degli attributi che definiremo contesto. In questo modo, ogni componente figlio potrà accederci.

6.9.2.2 FrontEnd :: View :: Containers :: Header

Descrizione

Questa classe è un container dalla presenza costante. Esso infatti rappresenta e contiene tutte le componenti della parte alta della pagina.

Utilizzo

In base allo stato dell'applicazione, l'Header mostrerà contenuti diversi, come i pulsanti per le funzionalità dedicate agli utenti.

Relazioni con altre classi

FrontEnd::View::Containers::MainPage: pagina lanciata di default.
FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.3 FrontEnd :: View :: Containers :: MainPage

Descrizione

Questa classe è un container per la pagina iniziale dell'applicazione.

Utilizzo

Il suo scopo è di accogliere eventuali visitatori fornendo alcune informazioni di base sul nostro servizio.

6.9.2.4 FrontEnd :: View :: Containers :: Login

Descrizione

Questa classe è un container per la sezione di autenticazione degli utenti già registrati.

Utilizzo

Il suo scopo è di raccogliere le credenziali degli utenti ed effettuare l'autenticazione contattando il server remoto. In caso di procedura riuscita, reindirizza alla HomePage attraverso l'uso di un token specifico generato dal server remoto.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.5 FrontEnd :: View :: Containers :: SignIn

Descrizione

Questa classe è un container per la sezione di registrazione aziendale.

Utilizzo

Il suo scopo è di raccogliere i dati di una nuova azienda e inviarli al server remoto. In caso di procedura riuscita, reindirizza alla MainPage.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.6 FrontEnd :: View :: Containers :: MailVerified

Descrizione

Questa classe è un container per la visualizzazione di avvenuta registrazione in seguito al ricevimento di un invito.

Utilizzo

Il suo scopo è di accogliere l'utente attraverso il link di conferma spedito via mail.

6.9.2.7 FrontEnd :: View :: Containers :: RecoverAccount

Descrizione

Questa classe è un container per la prima fase di recupero password, necessaria per spedire un e-mail all'utente che ne ha bisogno.

Utilizzo

Il suo scopo è di raccogliere l'indirizzo e-mail dell'utente che richiede la procedura e inviarlo al server remoto. In caso di procedura riuscita, reindirizza alla MainPage.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.8 FrontEnd :: View :: Containers :: RecoverPassword

Descrizione

Questa classe è un container per la seconda fase di recupero password, necessaria per impostare una nuova chiave di accesso.

Utilizzo

Il suo scopo è di raccogliere la nuova password, controllando che l'utente l'abbia digitata correttamente, e inviarla al server remoto. In caso di procedura riuscita, reindirizza alla MainPage.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.9 FrontEnd :: View :: Containers :: HomePage

Descrizione

Questa classe è un container per la pagina che accoglie gli utenti subito dopo l'autenticazione. Dovrà mostrare le DSLI sulle quali si detengono permessi e fornire un punto di accesso per tutte le funzionalità utente.

Utilizzo

Il suo scopo è recuperare dal server remoto tutte le DSLI al quale si ha accesso ed esporle in una tabella accompagnate dalle dovute funzionalità.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.10 FrontEnd :: View :: Containers :: Profile

Descrizione

Questa classe è un container per le informazioni personali dell'utente che la visualizza. Dovrà mostrare tali informazioni e fornire un punto di accesso per le funzionalità di modifica del proprio account.

Utilizzo

Il suo scopo è di recuperare dal server remoto le informazioni necessarie ed esporle attraverso un layout chiaro e ordinato.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.11 FrontEnd :: View :: Containers :: ResetPassword

Descrizione

Questa classe è un container per impostare una nuova password conoscendo quella attuale. Attraverso il design pattern Decorator si riutilizzerà il componente NewPassword aggiungendo la richiesta della password attuale.

Utilizzo

Il suo scopo è di controllare la vecchia password e raccogliere quella nuova, controllando che l'utente l'abbia digitata correttamente. Successivamente dovrà inviarla al server remoto e in caso di procedura riuscita, reindirizzare alla pagina del profilo utente.

Relazioni con altre classi

FrontEnd::Actions::changePassword

6.9.2.12 FrontEnd :: View :: Containers :: ViewDSLI

Descrizione

Questa classe è un container dal contenuto molto dinamico. Infatti dopo aver ricevuto l'esito dell'esecuzione della DSLI, delega la costruzione del proprio contenuto a BuildDSLI. Layout e contenuto vengono quindi utilizzati fino alla riesecuzione della DSLI.

Utilizzo

Il suo scopo è principalmente quello di innescare l'esecuzione della DSLI, iniziando un processo di verifica della sintassi seguito da una richiesta di esecuzione vera e propria al server remoto. Il risultato, se consistente, viene poi organizzato da una classe esterna e visualizzato.

Relazioni con altre classi

FrontEnd::Actions::execDSLI

6.9.2.13 FrontEnd :: View :: Containers :: NewDSLI

Descrizione

Questa classe è un container per la pagina adibita alla creazione di una nuova DSLI. Espone dei metodi che consentono di decidere il nome da suggerire all'utente e l'eventuale codice che fin dall'inizio la DSLI deve avere. La presenza di questi parametri definisce se l'operazione è una creazione o una clonazione.

Utilizzo

Il suo scopo è di chiedere all'utente un nome per la DSLI, suggerendone uno di default. Se è presente anche del codice da clonare è necessario includerlo tra le informazioni da inviare al server remoto. In caso di procedura riuscita, reindirizza alla pagina precedente,

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.14 FrontEnd :: View :: Containers :: Editor

Descrizione

Questa classe è un container per l'editor DSLI. Deve visualizzare il codice della DSLI e fornire un punto di accesso per tutte le funzionalità di modifica.

Utilizzo

Il suo scopo è di recuperare dal server remoto il codice della DSLI aperta in editor e visualizzarlo nella macro area di testo. Le varie funzionalità dovranno essere abilitate in base a quale permesso l'utente detiene sulla DSLI.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.15 FrontEnd :: View :: Containers :: UserManagement

Descrizione

Questa classe è un container dedicato alla funzionalità di gestione degli utenti, riservata agli amministratori. Deve visualizzare tutti gli utenti dell' istanza aziendale e consentirne l'invito, la cancellazione e la modifica dei permessi.

Utilizzo

Il suo scopo è di recuperare dal server remoto l'elenco degli utenti e il rispettivo livello di permessi, queste informazioni devono quindi essere visualizzate in una tabella dove ogni riga possiede una funzione di eliminazione. Qualsiasi cambiamento confermato va inviato al server remoto. Sotto la tabella prenderà posto il form di invito utente.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.16 FrontEnd :: View :: Containers :: DSLIManagement

Descrizione

Questa classe è un container dedicato alla funzionalità di gestione delle DSLI, riservata agli amministratori. Deve visualizzare tutte le DSLI dell'istanza aziendale e consentirne la condivisione esterna, l'eliminazione e la modifica dei permessi.

Utilizzo

Il suo scopo è di recuperare dal server remoto l'elenco delle DSLI, queste informazioni devono quindi essere visualizzate in una tabella dove ogni riga possiede una funzione di eliminazione, di condivisione e di modifica dei permessi. Qualsiasi cambiamento confermato va inviato al server remoto.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.17 FrontEnd :: View :: Containers :: DSLIPermits

Descrizione

Questa classe è un container dedicato alla funzionalità di modifica dei permessi che ogni utente detiene sulla DSLI selezionata. Deve visualizzare tutti gli utenti dell'istanza aziendale accompagnati dal loro attuale permesso.

Utilizzo

Il suo scopo è di recuperare dal server remoto l'elenco degli utenti e tutti i permessi. Qualsiasi cambiamento confermato va inviato al server remoto.

6.9.2.18 FrontEnd :: View :: Containers :: ContactSupport

Descrizione

Questa classe è un container dedicato alla funzionalità che consente ad utenti non autenticati di poter contattare il supporto clienti.

Utilizzo

Il suo scopo è di recuperare le informazioni dall'utente e inviarle al server remoto. Successivamente la mail viene inviata al supporto clienti.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.9.2.19 FrontEnd :: View :: Containers :: Impersonate

Descrizione

Questa classe è un container dedicato alla funzionalità riservata ai SuperAdmin di impersonare un utente registrato a MaaS.

Utilizzo

Il suo scopo è di recuperare le informazioni dall'utente e inviarle al server remoto. Successivamente il SuperAdmin verrà autenticato come l'utente desiderato.

Relazioni con altre classi

FrontEnd::Actions::embodyUser

6.9.2.20 FrontEnd :: View :: Containers :: DataManagment

Descrizione

Questa classe è un container dedicato alla funzionalità di gestione dei database, riservata agli amministratori. Deve visualizzare tutti i collegamenti alle varie basi di dati e consentire l'eliminazione e la creazione di nuove istanze.

Utilizzo

Il suo scopo è di recuperare dal server remoto l'elenco dei database, queste informazioni devono quindi essere visualizzate in una tabella dove ogni riga possiede una funzione di eliminazione. Deve essere inoltre possibile creare un nuovo collegamento attraverso l'inserimento di una stringa di connessione.

Relazioni con altre classi

FrontEnd::View::Components: necessario per comporre la pagina.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.10 FrontEnd::View::Components

6.10.1 Descrizione

Pacchetto dei Components, componenti semplici.

6.10.2 Classi

6.10.2.1 FrontEnd :: View :: Components :: MTextArea

Descrizione

Questa classe è un componente rappresentante un'area di testo. Deve poter fornire il proprio valore in qualsiasi momento.

Utilizzo

Questo componente è utilizzato per chiedere agli utenti un valore testuale di elevate dimensioni e complessità.

6.10.2.2 FrontEnd :: View :: Components :: MImage

Descrizione

Questa classe è un componente rappresentante un immagine. Deve poter essere personalizzabile sia nelle dimensioni che nel contenuto.

Utilizzo

Questo componente è utilizzato per visualizzare immagini all'interno della pagina web.

6.10.2.3 FrontEnd :: View :: Components :: MForm

Descrizione

Questa classe è un componente rappresentante una casella di testo. Deve poter fornire il proprio valore in qualsiasi momento.

Utilizzo

Questo componente è utilizzato per chiedere agli utenti un valore testuale breve.

6.10.2.4 FrontEnd :: View :: Components :: MLink

Descrizione

Questa classe è un componente rappresentante un collegamento ipertestuale interno o esterno all'applicazione. Deve poter essere personalizzabile sia nell'indirizzo di destinazione sia che nel testo visualizzato dall'utente.

Utilizzo

Questo componente è utilizzato per reindirizzare gli utenti ad altre pagine web.

6.10.2.5 FrontEnd :: View :: Components :: MButton

Descrizione

Questa classe è un componente che rappresenta un pulsante. Deve poter essere personalizzabile sia nella forma che nel codice da eseguire in caso venga premuto.

Utilizzo

Questo componente è utilizzato per poter far eseguire determinate azioni agli utenti.

6.10.2.6 FrontEnd :: View :: Components :: MTable

Descrizione

Questa classe è un componente che rappresenta una tabella di valori. Deve poter accettare come input un file json di chiavi e valori.

Utilizzo

Questo componente è utilizzato per visualizzare grandi quantità di dati organizzati.

6.10.2.7 FrontEnd :: View :: Components :: MDataRow

Descrizione

Questa classe è un componente rappresentante un database all'interno di una tabella. Contiene i parametri per la visualizzazione e l'esecuzione di azioni sul dato che rappresenta.

Utilizzo

Questo componente è utilizzato visualizzare il nome del database e consentirne l'eliminazione.

Relazioni con altre classi

FrontEnd::View::Components::MButton: necessario per utilizzare le funzionalità offerte.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.10.2.8 FrontEnd :: View :: Components :: MDSLIRow

Descrizione

Questa classe è un componente rappresentante una DSLI all'interno di una tabella. Contiene i parametri per la visualizzazione e l'esecuzione di azioni sul dato che rappresenta.

Utilizzo

Questo componente è utilizzato visualizzare dati salienti sulla DSLI e consentirne eliminazione, clonazione e modifica in base ai permessi dell'utente.

Relazioni con altre classi

FrontEnd::View::Components::MButton: necessario per utilizzare le funzionalità offerte.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.10.2.9 FrontEnd :: View :: Components :: MUserRow

Descrizione

Questa classe è un componente rappresentante un utnete all'interno di una tabella. Contiene i parametri per la visualizzazione e l'esecuzione di azioni sul dato che rappresenta.

Utilizzo

Questo componente è utilizzato visualizzare dati salienti dell'utente e consentirne eliminazione e modifica dei permessi.

Relazioni con altre classi

FrontEnd::View::Components::MButton: necessario per utilizzare le funzionalità offerte.
FrontEnd::Actions::rootAction: necessario per lanciare una qualsiasi action.

6.10.2.10 FrontEnd :: View :: Components :: MTextBox

Descrizione

Questa classe è un componente rappresentante una casella di testo. Deve poter fornire il proprio valore in qualsiasi momento.

Utilizzo

Questo componente è utilizzato per chiedere agli utenti un breve valore testuale o una password.

7 Design Pattern

Un *Desing Pattern*$_g$ individua e specifica una soluzione comune ad un problema comune. Quindi i *Design Pattern* offrono fonti di conoscenze utilissime ai progettisti. Possiamo classificare i package secondo i seguenti criteri:

- **Design Pattern Architetturali**: schemi base per la organizzazione di prodotti software;
- **Design Pattern Creazionali**: astraggono il processo di creazione di oggetti;
- **Design Pattern Strutturali**: astraggono la composizione di oggetti;
- **Design Pattern Comportamentali**: astraggano la composizione comportamentale degli oggetti.

7.1 Design Pattern Architetturale

7.1.1 Middleware

- **Scopo**: Abilita la intercomunicazione tra componenti software che risiedono in contesti applicativi diversi;
- **Utilizzo**: Il pattern middleware è un pattern nativo di Express.JS. Mentre, il framework LoopBack è una estensione del framework Express. In Loopback il middleware a differenza di Espress può essere programmato in due modi: via programmatica oppure via dichiarativa. Nel contesto di MaaS il middleware viene utilizzato nel contesto della gestione del flusso di richieste REST. Il pattern middleware è una implementazione del Design Pattern *Chain of Responsability*.

7.1.2 Redux

- **Scopo**: Redux, basandosi su Flux, si dissocia da molti altri pattern architetturali basandosi su un concetto di applicazione fondato su un flusso di informazioni unidirezionale;
- **Utilizzo**: Redux si basano sul concetto di flusso di informazioni unidirezionali. Redux è composto quattro diversi tipi di componenti: actions, reducers, uno store e views. Le actions sono dei plain objects utilizzati per descrivere allo store le operazioni effettuate nel client. I reducers sono delle funzioni pure con il compito di generare un nuovo stato basandosi sullo stato precedente e aggiornandolo con le informazioni fornite dalle actions. Lo store è il componente del sistema che unisce le azioni con i reducer e ha accesso allo stato del sistema. Le view invece si occupano della parte grafica, e della corrispondente parte logica, del client.

7.2 Design Pattern Creazionali

7.2.1 Singleton

- **Scopo**: Viene utilizzato per le classi che devono essere instaziate una sola volta;

- **Utilizzo**: In ambiente Node.js ogni modulo è una implementazione del **Design Pattern Singleton**. Infatti, durante il caricamento dei moduli ciascun modulo in parte viene caricato una sola volta e ogni successiva occorrenza di caricamento riferisce la prima istanza del modulo caricata.

7.3 Design Pattern Strutturali

7.3.1 Facade

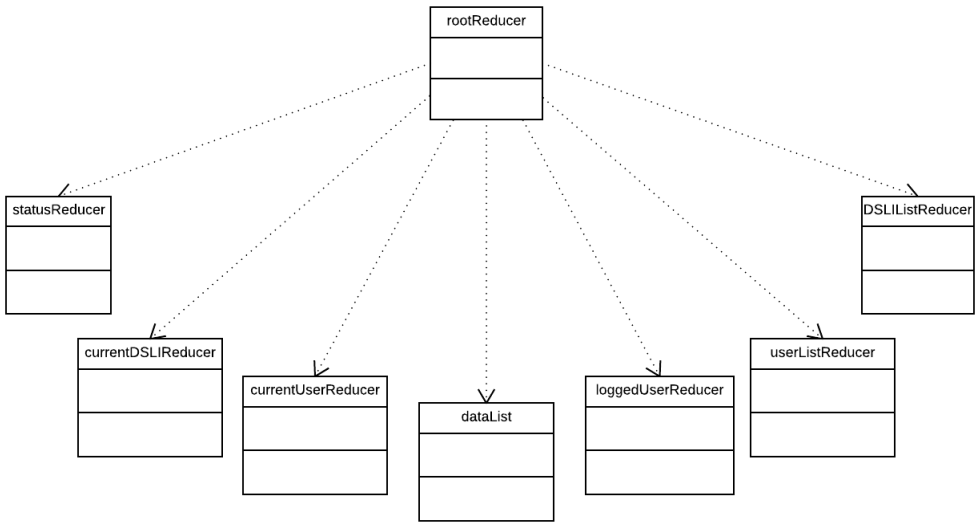

Fig 10: Organizzazione dei reducers

- **Scopo:** Viene utilizzato per rendere visibili solamente alcune cose agli altri oggetti ed avere un unico punto di accesso semplificato a un sottosistema fornendo un'interfaccia di alto livello e minimizzando dunque le comunicazioni e le dipendenze;
- **Utilizzo:** Viene utilizzato all'interno della classe Reducers::rootReducer. Mediante l'utilizzo del metodo combineReuducers() definito nella libreria Redux i vari reducers definiti negli appositi file vengono uniti per creare un unico reducer che gestisce ogni caso previsto dai vari reducers. In tal modo le richieste vengono delegate agli appositi reducers senza che la classe store conosca i reducers sottostanti.

7.3.2 Decorator

- **Scopo:** Viene usato per riutilizzare un componente consentendo di aggiungere funzionalità senza ripetere codice. Il componente base viene quindi incluso in un altro componente e integrato con decorazioni;
- **Utilizzo:** Quando viene chiesto di inserire una nuova password la prassi è sempre molto simile: è necessario inserire la nuova chiave due volte, per assicurarsi di averla digitata correttamente. Tuttavia a volte, come in Containers::ChangePassword, è

necessario fornire qualche dato in più. Proprio per questo il componente Components::MPasswordSet contiene le due caselle di testo e le funzionalità di confronto per una nuova password, così facendo indipendentemente da quali a ltre informazioni saranno necessarie, quel codice non dovrà essere ripetuto.

7.4 Design Pattern Comportamentali

7.4.1 Command

- **Scopo:** Viene utilizzato per incapsulare richieste o operazioni da eseguire all'interno di oggetti rappresentanti un comando. In questo modo chi genera l'azione non ha bisogno di conoscere i particolari su come essa viene svolta. In questo modo si crea un accoppiamento debole fra chi crea l'azione e chi la porta a termine;
- **Utilizzo:** Questo design pattern è alla base dell'architettura del client, infatti Redux si basa sul generare Action in risposta ad eventi. Queste Action vengono spedite ai Reducers che fungono da ricevente. Infine l'azione viene svolta dagli appositi Reducers in grado di interpretare l'Action e svolgere l'operazione.

7.4.2 Observer

- **Scopo:** Viene utilizzato per aggiornare le componenti View in seguito ad un cambiamento dei dati che rappresentano. In questo modo si evita un accoppiamento troppo forte e si ha la possibilità di aggiungere subscribers senza limitazioni;
- **Utilizzo:** Tutte le componenti React sono dotate del metodo render() che le aggiorna in base ai dati contenuti nello stato dell'applicazione. Attraverso un metodo di Redux::Store, Subscribe(), è possibile registrare delle componenti. Non appena lo stato varia, tutte le componenti registrate e le loro figlie eseguono render().

8 Tracciamento

8.1 Requisiti - Componenti

Requisito	Descrizione	Componenti
R-3F1	Un utente non autenticato può registrare una nuova azienda	FrontEnd::View::-Containers
R-3F1.1	L'applicazione richiede il nome dell'azienda da registrare	FrontEnd::View::-Components
R-3F1.2	L'applicazione richiede l'indirizzo email dell'utente	FrontEnd::View::-Components
R-3F1.2.1	L'email deve essere composta da una serie di caratteri alfanumerici e/o caratteri speciali seguiti dal carattere @ e infine dal dominio	FrontEnd::Actions
R-3F1.3	Alla conferma dei dati inseriti l'applicazione risponde con un esito	FrontEnd::View::-Components
R-3F1.3.1	Se l'indirizzo email dell'utente è già utilizzata nel sistema, l'applicazione visualizza un errore	BackEnd FrontEnd::Reducers
R-3F1.3.2	Se il nome dell'azienda è già utilizzato all'interno del sistema, l'applicazione visualizza un errore	BackEnd FrontEnd::Reducers
R-3F1.3.3	Se i dati inseriti sono validi, l'applicazione effettua la pre-registrazione dell'azienda	BackEnd FrontEnd::Reducers
R-3F1.3.3.1	I dati dell'azienda vengono salvati per sole 24 ore al termine delle quali vengono cancellati in assenza di utenti registrati	BackEnd::Common::-Models
R-3F1.3.3.2	Viene mandato un invito come proprietario all'indirizzo email inizialmente fornito	BackEnd::Common::-Models
R-3F1.3.3.3	Completata con successo la registrazione utente del proprietario, l'azienda viene registrata in modo definitivo	BackEnd::Common::-Models
R-3F2	Un utente invitato può usufruire della registrazione utente	FrontEnd::View::-Containers

R-3F2.1	L'applicazione risponde con un esito	FrontEnd::Reducers
R-3F2.1.1	Se si verifica un errore l'applicazione lo visualizza	BackEnd FrontEnd::Reducers
R-3F2.1.2	Se i dati sono validi l'applicazione crea l'account	BackEnd::Common::-Models
R-3F2.1.2.1	Se l'utente ha creato la sua istanza di azienda, essa diventa valida	BackEnd::Common::-Models
R-3F2.1.2.2	Se l'utente è stato invitato, esso appartiene ora all'istanza della sua azienda	BackEnd::Common::-Models
R-3F2.1.2.3	L'applicazione torna alla homepage	FrontEnd::Reducers
R-3F3	Un utente non autenticato ma in possesso di un account può procedere con il login	FrontEnd::View::-Containers
R-3F3.1	L'applicazione richiede l'email dell'utente	FrontEnd::View::-Components
R-3F3.2	L'applicazione richiede la password dell'utente	FrontEnd::View::-Components
R-3F3.3	Alla conferma delle credenziali inserite l'applicazione risponde con un esito	FrontEnd::Reducers
R-3F3.3.1	Se l'indirizzo email non è presente all'interno del database, l'applicazione visualizza un errore	BackEnd::Common::-Models FrontEnd::Reducers
R-3F3.3.2	Se la password non corrisponde a quella legata all'indirizzo email fornito, l'applicazione visualizza un errore	BackEnd::Common::-Models FrontEnd::Reducers
R-3F3.3.3	Se le credenziali inserite sono esatte, l'utente si autentica e viene rediretto alla homepage	BackEnd::Common::-Models FrontEnd::Reducers
R-3F4	Un utente registrato in possesso dell'email può reimpostare la sua password	FrontEnd::View::-Containers
R-3F4.1	L'applicazione richiede all'utente la email di accesso al sistema	FrontEnd::View::-Components
R-3F4.2	Alla conferma dei dati inseriti l'applicazione risponde con un esito	FrontEnd::Reducers

R-3F4.2.1	Se l'email digitata non è registrata nel sistema viene svuotato il form e viene visualizzato un avviso	BackEnd::Common::-Models FrontEnd::Reducers
R-3F4.2.2	Se l'email digitata è registrata nel sistema viene spedito un link per procedere nella procedura e l'utente torna alla homepage	BackEnd::Common::-Models FrontEnd::Reducers
R-3F4.3	L'applicazione richiede all'utente di fornire una nuova password	FrontEnd::View::-Components
R-3F4.3.1	La password deve essere lunga almeno 8 caratteri	FrontEnd::Actions
R-3F4.3.2	La password deve contenere almeno un numero, una lettera normale e una lettera maiuscola	FrontEnd::Actions
R-3F4.4	L'applicazione richiede all'utente di digitare per una seconda volta la password precedentemente fornita	FrontEnd::View::-Components
R-3F4.4.1	La password inserita deve essere uguale alla password precedentemente fornita	FrontEnd::View::-Components
R-3F4.5	Alla conferma dei dati inseriti l'applicazione risponde con un esito	FrontEnd::Reducers
R-3F4.5.1	L'applicazione cambia la vecchia password di accesso con quella nuova	BackEnd::Common::-Models
R-3F4.5.2	L'applicazione redirige l'utente alla homepage	FrontEnd::Reducers
R-3F5	Un utente autenticato può effettuare il logout, cioè uscire dalla piattaforma	FrontEnd::View::-Components
R-3F5.1	Un utente autenticato può cliccare il pulsante di Logout in ogni momento	FrontEnd::View::-Components
R-3F5.2	L'applicazione reindirizza l'utente alla home page	FrontEnd::Reducers
R-3F6	Un utente autenticato può visualizzare e modificare i propri dati personali	FrontEnd::View::-Containers
R-3F6.1	L'utente autenticato può visualizzare il proprio indirizzo email	FrontEnd::View::-Containers

R-3F6.2	Un utente autenticato può cambiare la password di accesso	FrontEnd::View::-Containers
R-3F6.2.1	L'applicazione richiede la password attuale	FrontEnd::View::-Components
R-3F6.2.2	L'applicazione richiede la nuova password	FrontEnd::View::-Components
R-3F6.2.2.1	La password deve essere lunga almeno 8 caratteri	FrontEnd::Actions
R-3F6.2.2.2	La password deve contenere almeno un numero, una lettera normale e una lettera maiuscola	FrontEnd::Actions
R-3F6.2.3	L'applicazione richiede di scrivere per una seconda volta la password precedentemente fornita	FrontEnd::View::-Components
R-3F6.2.3.1	La password inserita deve essere uguale alla password precedentemente fornita	FrontEnd::View::-Components
R-3F6.2.4	Alla conferma dei dati inseriti l'applicazione risponde con un esito	FrontEnd::Reducers
R-3F6.2.4.1	Se la password attuale non è corretta l'applicazione risponde con un errore	BackEnd::Common::-Models FrontEnd::Reducers
R-3F6.2.4.2	Se le password sono corrette il sistema cambia la vecchia password di accesso con quella nuova	BackEnd::Common::-Models
R-3F6.2.4.3	L'applicazione redirige l'utente al proprio profilo	FrontEnd::Reducers
R-2F6.3	Un utente autenticato può cambiare la propria immagine di profilo	FrontEnd::View::-Components
R-3F6.3.1	L'utente sceglie un immagine locale attraverso una dialog e l'applicazione risponde con un esito	FrontEnd::View::-Components
R-3F6.3.1.1	Se l'immagine è JPG o PNG e inferiore ad 1MB viene ridimensionata ad una risoluzione di 256x256 e sovrascrive l'immagine precedente	FrontEnd::Actions

R-3F6.3.1.2	Se l'immagine non può essere ridimensionata o non è valida viene visualizzato un errore	FrontEnd::Reducers
R-3F7	Un utente autenticato visualizza la homepage dell'applicazione	FrontEnd::View::-Containers
R-3F8	Un utente autenticato può eseguire una DSLI per accedere a i dati richiesti	FrontEnd::View::-Containers
R-3F8.1	L'applicazione interpreta il testo della DSLI e produce una query	FrontEnd::Services
R-3F8.1.1	L'applicazione riscontra un errore durante la interpretazione della DSLI e lo segnala	FrontEnd::Reducers
R-3F8.2	L'applicazione sottopone la query ai database aziendali	BackEnd::Common::-Models
R-3F8.2.1	Se la query provoca un errore a livello database, esso viene dunque segnalato	FrontEnd::Reducers
R-3F8.3	L'applicazione visualizza i dati raccolti dalla query creando una nuova vista	FrontEnd::Services
R-3F8.3.1	L'applicazione può produrre pagine di tipo Cell	FrontEnd::Services
R-3F8.3.1.1	Una cella può contenere un valore di tipo stringa	FrontEnd::View::-Components
R-3F8.3.1.2	Una cella può contenere un valore di tipo array	FrontEnd::View::-Components
R-3F8.3.1.3	Una cella può contenere un valore di tipo object	FrontEnd::View::-Components
R-3F8.3.1.4	Una cella può contenere un valore di tipo link	FrontEnd::View::-Components
R-3F8.3.1.5	Una cella può contenere un valore di tipo image, fornendo un'anteprima la cui taglia è definibile via DSLI	FrontEnd::View::-Components
R-3F8.3.2	L'applicazione può produrre pagine di tipo Document	FrontEnd::Services
R-3F8.3.3	L'applicazione può produrre pagine di tipo Collection	FrontEnd::Services

R-3F8.3.3.1	Le righe della collection possono essere riordinate dall'utente	FrontEnd::View::-Components
R-3F8.3.4	L'applicazione può produrre pagine di tipo Dashboard	FrontEnd::Services
R-3F8.3.4.1	L'utente autenticato può eseguire una action sui dati visualizzati	FrontEnd::Actions
R-3F8.3.4.1.1	L'applicazione genera un file contenente i dati visualizzati	FrontEnd::Actions
R-3F8.3.4.1.1.1	L'applicazione richiede il formato da utilizzare per la generazione del file	FrontEnd::View::-Components
R-3F8.3.4.1.1.2	Se l'utente sceglie JSON, l'applicazione fa scaricare il file JSON attraverso il browser	FrontEnd::Actions
R-3F8.3.4.1.1.3	Se l'utente sceglie CSV, l'applicazione fa scaricare il file CSV attraverso il browser	FrontEnd::Actions
R-3F8.3.4.1.2	L'applicazione invia una mail contenente i dati visualizzati	FrontEnd::Actions
R-3F8.3.4.1.2.1	L'applicazione richiede l'indirizzo mail del destinatario	FrontEnd::View::-Components
R-3F8.4	L'utente autenticato può cliccare su gli elementi ipertestuali per visualizzare il document di un elemento	FrontEnd::Actions
R-3F8.4.1	L'applicazione prepara una query per raccogliere tutte le informazioni sull'elemento selezionato	FrontEnd::Actions
R-3F8.4.2	L'applicazione sottopone la query al database e recupera le informazioni	BackEnd::Common::-Models
R-3F8.4.3	L'applicazione visualizza le informazioni del document in una nuova pagina	FrontEnd::View::-Containers
R-3F8.5	L'utente autenticato può riordinare le righe per colonne	FrontEnd::View::-Components

R-3F8.5.1	L'applicazione predispone nell'header di ogni colonna, un pulsante per riordinare le righe in ordine crescente secondo l'attributo scelto	FrontEnd::View::-Components
R-3F8.5.2	L'applicazione predispone nell'header di ogni colonna, un pulsante per riordinare le righe in ordine decrescente secondo l'attributo scelto	FrontEnd::View::-Components
R-3F9	Un utente membro può creare una nuova DSLI	FrontEnd::View::-Containers
R-3F9.1	L'applicazione predispone un pulsante adibito alla creazione di nuove DSLI	FrontEnd::View::-Components
R-3F9.2	L'applicazione mostra una form che richiede il nome della nuova DSLI	FrontEnd::View::-Components
R-3F9.3	L'applicazione crea una nuova DSLI vuota di cui l'utente è creatore	BackEnd::Common::-Models
R-3F10	Un utente membro può, se permesso, leggere e modificare il testo di una DSLI	FrontEnd::View::-Containers
R-3F10.1	L'applicazione recupera e visualizza il testo della DSLI	FrontEnd::Actions FrontEnd::Reducers
R-3F10.2	L'utente membro può clonare la DSLI, creandone un'altra con lo stesso testo a suo nome	FrontEnd::Actions
R-3F10.3	L'utente membro può, se permesso, modificare il testo della DSLI	FrontEnd::View::-Components
R-3F10.3.1	L'utente membro può salvare le modifiche effettuate, sovrascrivendo la vecchia DSLI	FrontEnd::Actions
R-3F10.3.2	L'utente membro può annullare le modifiche effettuate, ripristinando il testo originale	FrontEnd::Actions
R-2F10.3.3	L'editor di testo offre funzioni più complesse simili ad un ambiente di sviluppo	FrontEnd::View::-Components
R-2F10.3.3.1	L'editor di testo offre funzioni di evidenziamento delle parole chiave	FrontEnd::View::-Components

R-2F10.3.3.2	L'editor di testo offre funzioni di auto-completamento di parole chiave	FrontEnd::View::-Components
R-2F10.3.3.3	L'editor di testo offre funzioni di indentazione della codifica DSLI	FrontEnd::View::-Components
R-3F10.4	L'utente membro può, se permesso, eliminare definitivamente una DSLI	FrontEnd::Actions
R-3F10.5	L'utente membro può, se permesso, sovrascrivere il nome della DSLI con uno nuovo	FrontEnd::Actions
R-3F11	Un utente amministratore può gestire le DSLI dell'azienda	FrontEnd::View::-Containers
R-3F11.1	L'utente amministratore può visualizzare tutte le DSLI dell'azienda	FrontEnd::View::-Components
R-3F11.1.1	L'applicazione visualizza le DSLI raccolte in una lista verticale, accompagnate dalle funzioni di modifica, impostazione di permessi e disabilitazione	FrontEnd::View::-Components
R-3F11.2	L'utente amministratore può impostare i permessi della DSLI selezionata	FrontEnd::View::-Containers
R-3F11.2.1	L'applicazione visualizza un elenco di tutte le DSLI	FrontEnd::View::-Components
R-3F11.2.2	L'applicazione accompagna ogni DSLI elencata con i suoi attuali permessi sugli utenti	FrontEnd::View::-Components
R-3F11.2.3	L'utente amministratore può modificare liberamente i permessi della DSLI	FrontEnd::View::-Components
R-3F11.2.4	L'utente amministratore può confermare e rendere effettivi i cambiamenti effettuati	FrontEnd::Actions
R-3F11.2.5	L'utente amministratore può chiudere la dialog, annullando tutte le modifiche effettuate	FrontEnd::Actions
R-3F11.3	Un utente amministratore può condividere l'accesso alla pagina di esecuzione di una DSLI	FrontEnd::Actions

R-3F11.3.1	L'applicazione richiede l'indirizzo e-mail al quale mandare il link della DSLI	FrontEnd::View::-Components
R-3F11.3.1.1	L'indirizzo email deve essere composta da una serie di caratteri alfanumerici e/o caratteri speciali seguiti dal carattere @ e infine dal dominio	FrontEnd::View::-Components
R-3F11.3.1.2	Viene verificato l'indirizzo e-mail che risponde con un esito	FrontEnd::Reducers
R-3F11.3.1.2.1	Se L'indirizzo e-mail è valido si può proseguire con la stesura del messaggio	FrontEnd::View::-Components
R-3F11.3.1.2.2	Se si verifica un errore l'applicazione lo visualizza	FrontEnd::Reducers
R-3F11.3.2	L'applicazione richiedel la creazione del link d'accesso attraverso l'utilizzo di un token	FrontEnd::Actions
R-3F11.3.3	L'applicazione richiede una conferma dopo la quale invia l'email	FrontEnd::Actions
R-3F11.3.3.1	Se l'e-mail è valida si può procedere ad invitare un nuovo utente	FrontEnd::Reducers
R-3F11.4.1	L'applicazione apre una dialog in cui chiede la conferma per la eliminazione della DSLI	FrontEnd::View::-Containers
R-3F11.4.2	L'utente amministratore può confermare attraverso il bottone conferma presente nella dialog la eliminazione della DSLI	BackEnd::Common::-Models
R-3F11.4.3	L'utente amministratore può annullare attraverso il bottone annulla presente nella dialog la eliminazione della DSLI	FrontEnd::View::-Containers
R-3F12	Un utente amministratore può agire sui livelli di accesso degli utenti della sua azienda	FrontEnd::View::-Containers
R-3F12.1	L'applicazione raccoglie tutti gli utenti iscritti alla stessa azienda dell'utente amministratore, proprietario escluso	BackEnd::Common::-Models

R-3F12.2	L'applicazione visualizza i dati raccolti in una lista verticale	FrontEnd::View::-Components
R-3F12.3	L'applicazione accompagna ogni utente elencato con l'attuale livello di accesso e un pulsante di disabilitazione	FrontEnd::View::-Components
R-3F12.4	L'utente amministratore può cambiare liberamente il livello di accesso di ogni utente	FrontEnd::View::-Components
R-2F12.5	L'utente amministratore può, dopo una conferma, disabilitare un account utente	FrontEnd::Actions
R-3F12.5.1	L'utente amministratore può confermare la disabilitazione dell'utente	FrontEnd::Actions
R-3F12.5.2	L'utente amministratore può annullare la disabilitazione dell'utente	FrontEnd::Actions
R-3F23	Un superadmin autenticato può impersonificare un qualsiasi utente registrato della piattaforma	FrontEnd::View::-Containers
R-3F23.1	L'applicazione richiede l'email dell'utente che si vuole impersonare	FrontEnd::View::-Components
R-3F23.2	Un superadmin può confermare l'e-mail per l'autenticazione inserita premendo sull'apposito bottone	FrontEnd::Actions
R-3F23.2.1	L'applicazione apre una dialog in cui chiede la conferma per l'autenticazione dell'utente in modo da consentire al superadmin di poterlo impersonare	FrontEnd::View::-Components
R-3F23.2.2	Un superadmin può confermare attraverso il bottone conferma presente nella dialog l'autenticazione	FrontEnd::Actions
R-3F23.2.3	Un superadmin può annullare attraverso il bottone annulla presente nella dialog l'autenticazione	FrontEnd::Actions
R-3F26	Un utente Ospite può visualizzare l'esecuzione di una DSLI	FrontEnd::View::-Containers

R-3F26.1	L'utente ospite clicca sul link mandato per mail	BackEnd
R-3F26.2	L'applicazione apre una pagina browser contente la DSLI già eseguita	FrontEnd::View::-Containers
R-3F26.3	L'utente può visualizzare in dettaglio la DSLI	FrontEnd::View::-Components
R-3F26.4	Il link della DSLI viene rifiutato dall'applicazione perchè il token è scaduto	BackEnd
R-3F27	L'applicazione offre un servizio di Customer Service	FrontEnd::View::-Containers
R-3F27.1	L'aplicazione predispone una form per l'inserimento della email e del testo	FrontEnd::View::-Components
R-3F27.2	L' utente inserisce la propria email nella form	FrontEnd::View::-Components
R-3F27.3	L'utente scrive il messaggio da inviare al superadmin	FrontEnd::View::-Components
R-3F27.4	L'utente clicca il bottone invia e-mail	FrontEnd::Actions
R-3F27.5	L'applicazione ha verificato l'email e non ha trovato errori , quindi la invia	BackEnd::Common::-Models FrontEnd::Actions
R-3F27.6	L'applicazione ha verificato ed ha trovato un errore. L'errore viene segnalato all'utente	FrontEnd::Reducers
R-3F28	L'utente autenticato può riordinare la tabella per attributi	FrontEnd::View::-Components
R-3F28.1	L'applicazione predispone nell'header di ogni colonna, un pulsante per riordinare le righe in ordine crescente secondo l'attributo scelto	FrontEnd::Reducers
R-3F28.2	L'applicazione predispone nell'header di ogni colonna, un pulsante per riordinare le righe in ordine decrescente secondo l'attributo scelto	FrontEnd::Reducers

R-3F29	L'utente amministratore può invitare nuovi utenti ad iscriversi alla piattaforma	FrontEnd::View::-Containers
R-3F29.1	L'applicazione richiede l'indirizzo email al quale mandare l'invito	FrontEnd::View::-Components
R-3F29.1.1	L'email deve essere composta da una serie di caratteri alfanumerici e/o caratteri speciali seguiti dal carattere @ e infine dal dominio	FrontEnd::View::-Components
R-3F29.2	L'applicazione richiede di scegliere il livello di accesso per l'utente da invitare	FrontEnd::View::-Components
R-3F29.3	L'applicazione richiede una conferma dopo la quale invia la email	FrontEnd::Actions
R-3F29.3.1	Se la mail al quale recapitare l'invito è già utilizzata in MaaS la procedura si interrompe e si visualizza una dialog di errore	BackEnd::Common::-Models
R-3F29.3.2	Se la email è valida si può procedere ad invitare un nuovo utente	BackEnd::Common::-Models
R-3F34	L'admin gestisce i database aziendali quindi può visualizzarli , inserirli oppure rimuoverli	FrontEnd::View::-Containers
R-3F34.1	L'admin visualizza tutti i dati di un database tranne l'URI	BackEnd::Common::-Models
R-3F34.2	L'admin inserisce un database attraverso l'apposita form	FrontEnd::Actions FrontEnd::View::-Containers
R-3F34.3	L'admin rimuove il database dall'applicazione	BackEnd::Common::-Models

Tab 3: Tabella requisiti / componenti

8.2 Componenti - Requisiti

Componente	Requisiti
BackEnd	R-3F1.3.1 R-3F1.3.2 R-3F1.3.3 R-3F2.1.1 R-3F26.1 R-3F26.4
BackEnd::- Common::- Models	R-3F1.3.3.1 R-3F1.3.3.2 R-3F1.3.3.3 R-3F3.3.1 R-3F3.3.2 R-3F2.1.2.1 R-3F2.1.2.2 R-3F2.1.2 R-3F4.5.1 R-3F6.2.4.1 R-3F6.2.4.2 R-3F3.3.3 R-3F8.2 R-3F8.4.2 R-3F12.1 R-3F29.3.1 R-3F4.2.1 R-3F4.2.2 R-3F9.3 R-3F29.3.2 R-3F27.5 R-3F11.4.2 R-3F34.1 R-3F34.3

FrontEnd::-Actions	R-3F1.2.1 R-3F4.3.1 R-3F4.3.2 R-3F6.2.2.1 R-3F6.2.2.2 R-3F6.3.1.1 R-3F8.4 R-3F8.4.1 R-3F8.3.4.1 R-3F8.3.4.1.1 R-3F8.3.4.1.1.2 R-3F8.3.4.1.1.3 R-3F8.3.4.1.2 R-3F10.1 R-3F10.2 R-3F10.3.1 R-3F10.3.2 R-3F10.4 R-3F10.5 R-3F11.2.4 R-2F12.5 R-3F29.3 R-3F11.2.5 R-3F12.5.1 R-3F12.5.2 R-3F27.4 R-3F27.5 R-3F11.3 R-3F11.3.2 R-3F11.3.3 R-3F23.2 R-3F23.2.2 R-3F23.2.3 R-3F34.2

FrontEnd::- Reducers	R-3F1.3.1 R-3F1.3.2 R-3F1.3.3 R-3F3.3.1 R-3F3.3.2 R-3F2.1 R-3F2.1.1 R-3F3.3 R-3F4.2 R-3F4.5 R-3F5.2 R-3F6.2.4 R-3F6.2.4.1 R-3F6.3.1.2 R-3F3.3.3 R-3F28.1 R-3F28.2 R-3F10.1 R-3F8.1.1 R-3F8.2.1 R-3F2.1.2.3 R-3F4.5.2 R-3F4.2.1 R-3F4.2.2 R-3F6.2.4.3 R-3F27.6 R-3F11.3.1.2 R-3F11.3.1.2.2 R-3F11.3.3.1
FrontEnd::- Services	R-3F8.1 R-3F8.3 R-3F8.3.1 R-3F8.3.2 R-3F8.3.3 R-3F8.3.4

FrontEnd::- View::- Containers	R-3F1 R-3F2 R-3F3 R-3F4 R-3F6.1 R-3F6.2 R-3F6 R-3F7 R-3F8 R-3F8.4.3 R-3F9 R-3F10 R-3F11 R-3F11.2 R-3F12 R-3F23 R-3F26 R-3F26.2 R-3F27 R-3F11.4.1 R-3F11.4.3 R-3F29 R-3F34 R-3F34.2
FrontEnd::- View::- Components	R-3F1.1 R-3F1.2 R-3F1.3 R-3F3.1 R-3F3.2 R-3F4.1 R-3F4.3 R-3F4.4 R-3F4.4.1 R-3F5 R-3F5.1 R-3F6.2.1 R-3F6.2.2 R-3F6.2.3 R-3F6.2.3.1 R-2F6.3 R-3F6.3.1

FrontEnd::- View::- Containers	R-3F8.5 R-3F8.5.1 R-3F8.5.2 R-3F8.3.4.1.1.1 R-3F8.3.4.1.2.1 R-3F9.1 R-3F10.3 R-3F11.1 R-3F11.1.1 R-3F11.2.1 R-3F11.2.2 R-3F11.2.3 R-3F12.2 R-3F12.3 R-3F12.4 R-2F10.3.3 R-2F10.3.3.1 R-2F10.3.3.2 R-2F10.3.3.3 R-3F29.1 R-3F29.2 R-3F9.2 R-3F29.1.1 R-3F8.3.1.1 R-3F8.3.1.2 R-3F8.3.1.3 R-3F8.3.1.4 R-3F8.3.1.5 R-3F26.3 R-3F27.1 R-3F27.2 R-3F27.3 R-3F11.3.1 R-3F11.3.1.1 R-3F11.3.1.2.1 R-3F8.3.3.1 R-3F28 R-3F23.1 R-3F23.2.1

Tab 5: Tabella componenti / requisiti

A Descrizione Generale Design Pattern

A.1 Design Pattern Architetturali

A.1.1 Middleware

Il pattern Middleware vede le sue origini nel contesto dei Sistemi Operativi. In questo contesto il Middleware rappresenta uno strato di astrazione tra l'infrastruttura di basso livello della macchina e l'applicazione che deve interfacciarsi con i servizi offerti dalla macchina per il proprio corretto funzionamento. Possiamo pensare al Middleware come ad una interfaccia tra due contesti separati in cui il contesto più esterno si affida al contesto interno per ottenere dei servizi; come ad esempio tempo di esecuzione di processo, servizi di comunicazione con altri dispositivi ed ecc. Il pattern successivamente viene riutilizzato in contesto di applicazioni web. In tale contesto il middleware offre il servizio di interfaccia tra il cliente richiedente una risorsa di qualche tipologia ed il serviente che serve la richiesta del cliente. La forma di comunicazione determina la peculiarità del middleware. Infatti, esistono diverse tipologie di middleware: quelli orientati al messaggio, broker di messaggio e molti altri. Nelle immagini che seguono viene delineato la tipologia tipica del middleware in contesto web.

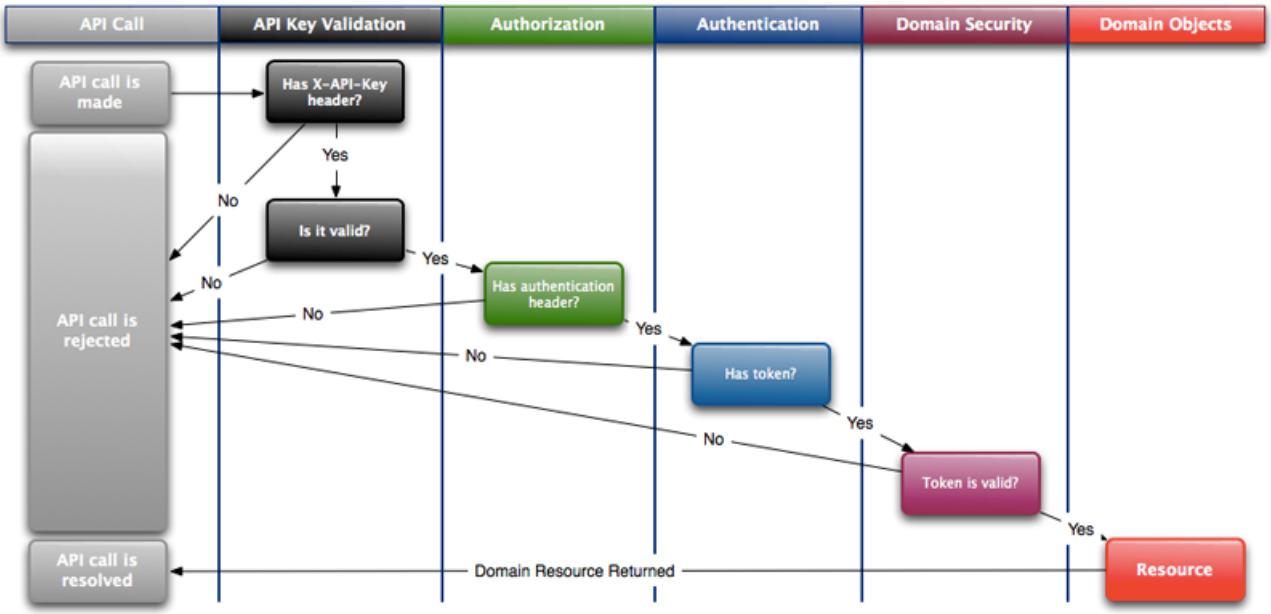

Fig 11: Vista estesa del middleware in contesto web

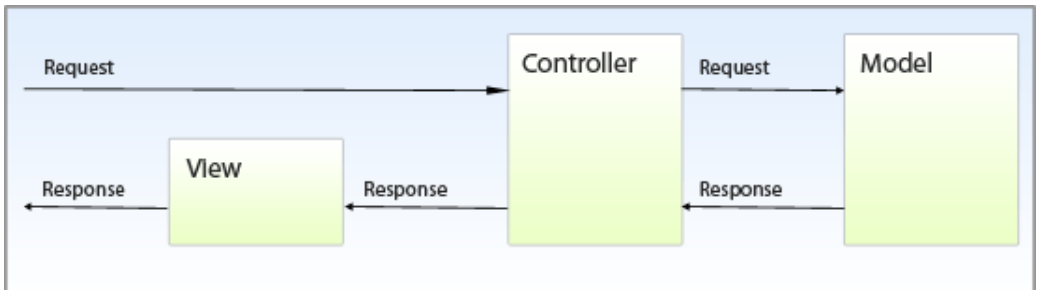

Fig 12: Vista del middleware dal punto di vista MVC

A.1.2 Flux

Flux è un pattern architetturale che complementa le componenti di React per la realizzazione di view componibili utilizzando un flusso di dati unidirezionali. Le applicazioni Flux sono divise in tre componenti principali: il dispatcher, gli stores e le views (componenti di React). Queste componenti non devono essere erroneamente paragonate con il pattern Model-View-Controller. I controllers sono presenti in un'applicazione Flux, ma sono controller-views, ovvero delle view generalmente situate in cima alla gerarchia delle view che hanno il compito di gestire le informazioni fornite loro dagli store e passarle ai loro figli. In più sono presenti anche action creators, metodi di supporto al dispatcher, che hanno il compito di descrivere tutte le operazioni possibili all'interno dell'applicazione. Questi metodi possono essere considerati come il quarto componente del ciclo di aggiornamento di Flux.
Flux è basato sul concetto di flusso di dati unidirezionali: quando un utente interagisce con una view di React, questa propaga una action, attraverso un dispatcher centrale, ai vari store che hanno accesso ai dati e alla logica dell'applicazione, i quali aggiornano di conseguenza le views interessate da tale action.
Il dispatcher è paragonabile ad un centro di smistamento del flusso di dati in un'applicazione di Flux. Si tratta essenzialmente di un registro dei callback ai vari store e non effettua nessuna operazione se non distribuire le actions agli stores. Ogni store si registra nel dispatcher fornendo il proprio callback. Quando un action creator fornisce al dispatcher una nuova azione, tutti gli store dell'applicazione ricevono l'action tramite i callback nel registro. La callback fornita dallo store riceve come parametro una action. All'interno di ogni callback è presente un'istruzione switch che collega, in base al tipo della action, al metodo interno dello store interessato. Dopo che gli stores sono stati aggiornati, trasmettono un evento per evidenziare il fatto che lo stato è stato aggiornato, in modo tale che le view possano richiedere il nuovo stato ed aggiornarsi a loro volta.

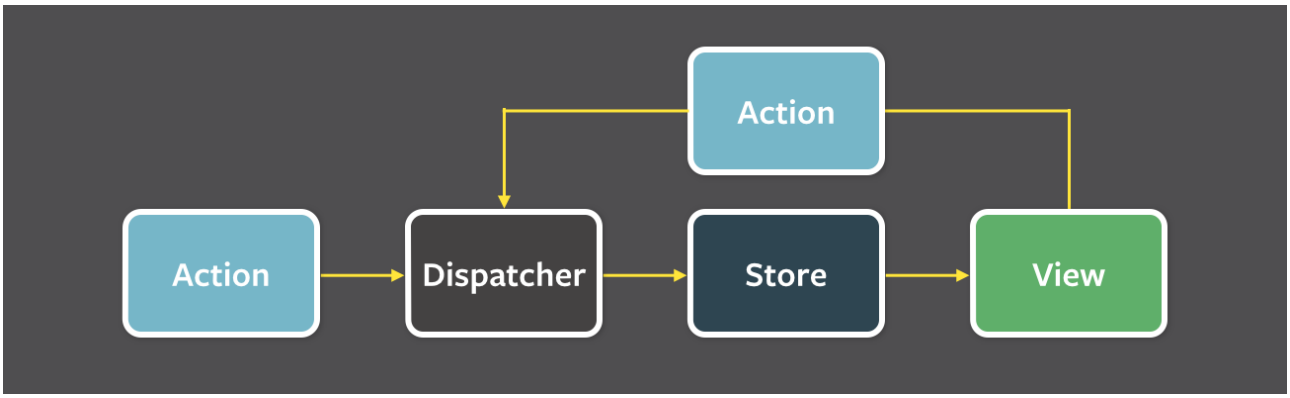

Fig 13: Flux

A.1.3 Redux

Redux è un pattern architetturale basato su vari aspetti di Flux. Come Flux, Redux impone che l'aggiornamento dei dati sia concentrato in un determinato componente del sistema (stores in Flux, reducers in Redux). Al posto di mutare direttamente i dati, entrambe le architetture richiedono che ogni operazione venga descritta un pain object denominato action. Detto ciò vi sono anche varie differenze tra i due pattern individuabili mediante la descrizione delle loro componenti. Redux è composto da: actions, reducers, store e view.
Come in Flux le actions sono dei plain objects utilizzati per mandare informazioni dall'applicazione allo store. Queste vengono generate medianti apposite funzioni denominate action creators che hanno come unico compito quello di generare specifiche actions.
Mentre è il compito delle actions descrivere cosa avviene è invece compito dei reducers specificare come lo stato dell'applicazione cambia in risposta. I reducers sono delle funzioni pure, il che significa che un reducer non dovrà mai mutare i suoi argomenti e/o effettuare operazioni che possono causare side-effects. I reducers sono dunque funzioni che prendono in input lo stato del sistema e una action e restituiscono allo store un oggetto contenente lo stato aggiornato.
Mentre le actions descrivono le operazioni avvenute e i reducer aggiornano lo stato in base alle action fornite è compito dello store di unire questi due componenti. Lo store ha le seguenti rensponsibilità:

- detiene l'accesso allo stato del sistema;
- permette allo stato di essere aggiornato chiamando l'opportuno action creator;
- permette di accedere allo stato del sistema mediante un apposito metodo;
- mediante un apposito metodo permette alle views di aggiornarsi ricavando i dati necessari dallo stato.

In una applicazione Redux sarà presente un singolo store mentre l'aspetto di gestione dei dati verrà gestito da vari reducers.

A.2 Design Pattern Creazionali

A.2.1 Singleton

Il design pattern Singleton garantisce che una classe Singleton abbia una sola istanza, e fornisce un unico punto di accesso ad esso.
I possibili contesti dove questo può essere utile sono i seguenti:

- In contesto con tante stampanti è necessario che ci sia un unico spooler di stampa;
- Un filtro digitale deve avere un unico A/D convertitore.

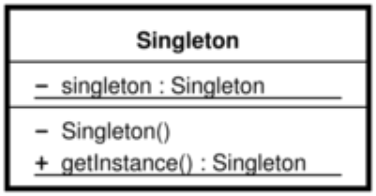

Fig 14: Design Pattern Singleton

A.3 Design Pattern Comportamentali

A.3.1 Facade

Il pattern Facade, di tipo strutturale basato sugli oggetti, permette di individuare un'interfaccia unificata per un insieme di interfacce nell'ambito di un sottosistema. Questo pattern in pratica consente di definire un'interfaccia a un livello più alto che semplifica l'accesso alle funzionalità erogate dal sottosistema e che fornisce un'entry-point unico al sottosistema stesso.
La presenza di un oggetto di facciata in un sottosistema permette di mascherare all'esterno la sua complessità interna, limitando le dipendenze dirette e l'accoppiamento. Il client comunica con il sottosistema inviando le sue richieste all'oggetto di facciata, il quale a sua volta funge da tramite verso le parti interne più idonee a fornire la risposta attesa. Il client non conosce come il sottosistema è strutturato e non ha accesso diretto ai suoi oggetti interni con i quali comunica unicamente tramite l'oggetto di facciata.

A.3.2 Decorator

Il pattern Decorator permettere di aggiungere a run-time nuove funzionalità ad un oggetto. Decorator fornisce un'alternativa flessibile all'ereditarietà per quanto riguarda l'estendere un oggetto con nuove funzionalità. Questo design pattern prevede la realizzazione di una classe decoratore che avvolge l'oggetto originale, eventualemente è anche possibile trovare varie classi decoratrici una dentro l'altra per aggiungere ulteriori funzionalità.

A.4 Design Pattern Strutturali

A.4.1 Command

Il pattern Command permette di inoltrare richieste ad oggetti senza conoscere l'operazione da eseguire e il destinatario della richiesta. Questo è possibile per il fatto che il pattern tratta la richiesta come un oggetto differente rispetto sia al richiedente che all'oggetto destinatario. Questo oggetto specifica l'azione da svolgere sul destinatario, sfruttandone i comportamenti in modo da tale da poter portare a termine la richiesta.
Il pattern Command permette quindi di incapsulare una richiesta in un oggetto permettendo al client di inoltrare richieste di varia natura, anche in funzione di destinatari diversi. Il pattern in questione può essere applicato:

- per parametrizzare gli oggetti rispetto ad una azione da compiere;
- per specificare, accodare ed eseguire svariate richieste in tempi diversi, anche trasferendo un comando da un contesto di esecuzione ad un altro;
- per consentire l'annullamento delle operazioni eseguite, mantenendo preventivamente lo stato per annullare gli effetti dei comandi stessi.

Il vantaggio più significativo nell'applicazione di questo pattern è il fatto di ottenere un perfetto disaccoppiamento tra l'oggetto che invoca il comando e il destinatario, ovvero quello che conosce il modo per portare a termine l'operazione. Questo aspetto consente di poter aggiungere comandi ulteriori associati a destinatari diversi in modo abbastanza immediato senza che sia necessario modificare le classi già esistenti. Inoltre il fatto che i comandi siano oggetti distinti permette di poterli eventualmente aggregare insieme allo scopo di formare un comando complesso, costituito da una serie arbitraria di azioni.

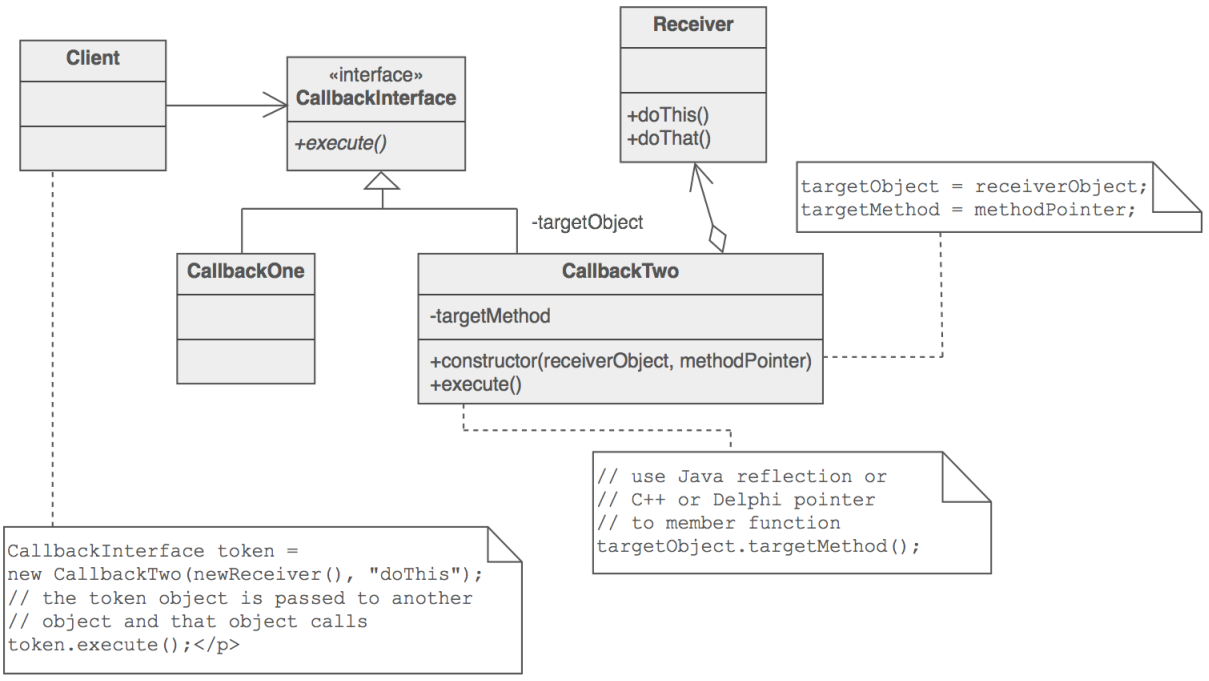

Fig 15: Command