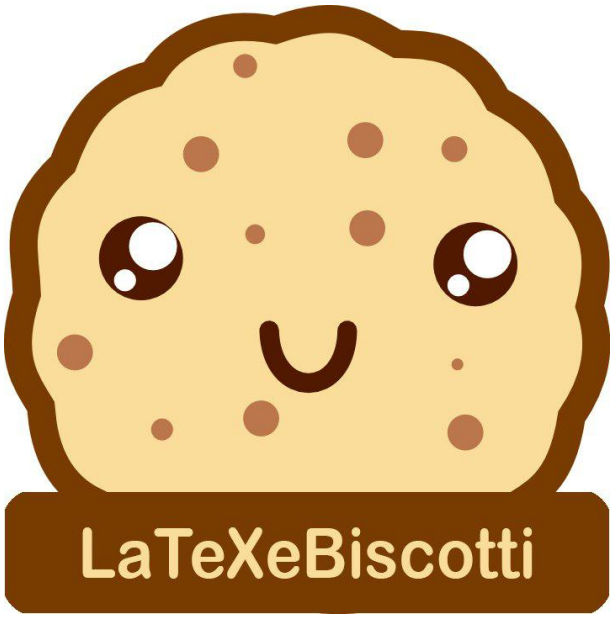

Specifica Tecnica

Gruppo LaTeXeBiscotti — Progetto UMAP

Informazioni sul documento	
Versione	2.0.0
Redazione	Marco Baggio
Verifica	Pietro Marchetto
Approvazione	Marco Baggio
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo LaTeXeBiscotti

Descrizione

Questo documento descrive la specifica tecnica e l'architettura dell'applicazione UMAP.

Diario delle Modifiche

Versione	**Data**	**Persone coinvolte**	**Descrizione**
2.0.0	2016-05-15	Marco Baggio (Responsabile)	Approvazione Documento.
1.2.0	2016-04-26	Pietro Marchetto (Verificatore)	Verifica Documento.
1.0.6	2016-04-24	Marco Baggio (Progettista)	Correzione tracciamento.
1.0.5	2016-04-11	Marco Baggio (Progettista)	Aggiunti design pattern Dependency Injection e DAO in sezione §6 e in appendice.
1.0.4	2016-04-23	Marco Baggio (Progettista)	Correzione nomi dei package come indicato in correzione RP e corretti i diagrammi di attività numeri: 7, 12, 13, 14, 39. Aggiunte guardie mancanti dal 25 in poi.
1.0.3	2016-04-22	Marco Baggio (Progettista)	Introdotti i DAO per accesso al Data Base.
1.0.2	2016-04-21	Marco Baggio (Progettista)	Spostata la logica dell'algoritmo predittivo in un package all'interno dei model.
1.0.1	2016-04-20	Marco Baggio (Progettista)	Introdotti aspetti negativi delle tecnologie in sezione §4.1.1.2.
1.0.0	2016-04-11	Filippo Todescato (Responsabile)	Approvazione Documento.
0.1.0	2016-04-03	Simone Garbin (Verificatore)	Verifica Documento.
0.0.19	2016-03-26	Federica Speggiorin, Andrea Barcaro (Progettisti)	Stesura sezione §8.2.
0.0.18	2016-03-26	Federica Speggiorin, Pietro Marchetto (Progettisti)	Stesura sezione §8.1.
0.0.17	2016-03-26	Pietro Marchetto (Progettista)	Stesura sezione §4.4.
0.0.16	2016-03-24	Marco Baggio (Progettista)	Stesura sezione §4.3.
0.0.15	2016-03-22	Filippo Todescato (Responsabile)	Stesura sezione §7.
0.0.14	2016-03-22	Pietro Marchetto (Progettista)	Stesura sezione §4.2.
0.0.13	2016-03-20	Filippo Todescato (Progettista)	Stesura sezione §4.1.
0.0.12	2016-03-17	Pietro Marchetto (Progettista)	Correzione errori vari sezione §5 e relative sottosezioni.
0.0.11	2016-03-16	Pietro Marchetto (Progettista)	Inserimento sezioni §5.5 e §5.6 con relative sottosezioni.

0.0.10	2016-03-15	Andrea Barcaro (Progettista)	Stesura Appendice A, sezione §A.1.
0.0.9	2016-03-15	Andrea Barcaro (Progettista)	Inizio stesura sezione §6, stesura sezione §6.1.
0.0.8	2016-03-15	Federica Speggiorin (Progettista)	Stesura completa indice.
0.0.7	2016-03-14	Giovanni Rodighiero, Federica Speggiorin (Progettisti)	Stesura sezione §3.2.
0.0.6	2016-03-13	Federica Speggiorin (Progettista)	Stesura sezioni dalla §5.1 alla §5.4.
0.0.5	2016-03-12	Pietro Marchetto (Progettista)	Stesura sezione §3.1.
0.0.4	2016-03-11	Giovanni Rodighiero (Progettista)	Modifica sezioni §2.2, §2.8; Stesura sezioni §2.3, §2.5, §2.6, §2.7, §2.9.
0.0.3	2016-03-11	Andrea Barcaro (Progettista)	Stesura sezioni §2.2, §2.4, §2.8.
0.0.2	2016-03-10	Andrea Barcaro (Progettista)	Stesura indice sezione §2, con scrittura sezione §2.1.
0.0.1	2016-03-10	Andrea Barcaro (Progettista)	Creazione documento e prima stesura sezione §1.

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del Documento

Il documento in esame si pone come obiettivo quello di definire la progettazione ad alto livello del prodotto software UMAP. In seguito, verrà presentata l'architettura generale ideata per il progetto in esame, illustrandone le varie caratteristiche, i componenti software che la compongono e una descrizione approfondita dei *Design Pattern*$_G$ che si andranno ad utilizzare.

1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di un *algoritmo predittivo*$_G$ in ambiente *Internet of Things*$_G$, in grado di analizzare i dati provenienti da "oggetti" inseriti in diversi contesti e fornire delle previsioni su possibili guasti o interazioni con nuovi utenti ed identificare dei pattern di comportamento di questi ultimi, al fine di prevedere le azioni degli stessi rispetto ad altri oggetti o ad altri contesti.

1.3 Glossario

Per evitare tutte le possibili incomprensioni e ambiguità sul linguaggio utilizzato e per massimizzare la comprensione da parte di tutti del documento, della terminologia specifica e di quella di dominio, degli acronimi e di tutte quelle parole che necessitano chiarimento, è stato redatto un Glossario, consultabile nel documento *Glossario v3.0.0*. Tutti i termini, la cui spiegazione è presente in *Glossario v3.0.0*, sono evidenziati, a tale scopo, con una G pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Analisi dei Requisiti**: *Analisi dei Requisiti v3.0.0*;
- **Norme di Progetto**: *Norme di Progetto v3.0.0*;

1.4.2 Informativi

- **Informazioni Scala**
  http://jim-mcbeath.blogspot.it/2010/12/scala-pros-and-cons.html
- **Documentazione Play Framework**
  https://www.playframework.com/documentation/2.5.x/Home
- **Documentazione AWS Machine Learning**
  https://aws.amazon.com/it/machine-learning/
- **Informazioni Node.js**
  http://voidcanvas.com/describing-node-js/
- **Documentazione MongoDB**
  https://docs.mongodb.org/manual/
- **Documentazione Twitter Bootstrap**
  https://bootstrapdocs.com/

- **Design Pattern**
  - http://www.math.unipd.it/~rcardin/sweb/Design%20Pattern%20Architetturali%20-%20Model%20View%20Controller_4x4.pdf
  - http://www.claudiodesio.com/ooa&d/mvc.htm

2 Tecnologie Utilizzate

In tale sezione vengono descritte e discusse tutte le tecnologie che il gruppo di lavoro LaTeXeBiscotti intende utilizzare per lo sviluppo del progetto UMAP. Verrà inoltre fornita una motivazione all'utilizzo di ogni singola tecnologia scelta.

2.1 Scala

Scala è un linguaggio di programmazione studiato ed ideato per integrare le caratteristiche e le funzionalità dei linguaggi orientati agli oggetti e dei linguaggi funzionali. Il codice prodotto da questo linguaggio, una volta compilato, produce del *bytecode*$_G$ che può essere eseguito su una Java *JVM*$_G$.
Aspetti **positivi** considerati:

- **Alta Produttività**: da studi effettuati, si è riscontrato che, dato un programma scritto in Scala, le linee di codice di uno equivalente scritto in Java stanno in un rapporto che può variare da 1:10 a 1:2. Assunte tali considerazioni, si capisce come si possa essere molto più produttivi in un periodo di tempo breve, paragonabile alla fase di Codifica dell'applicativo software;
- **Integrazione con Java**: Scala si integra perfettamente con il linguaggio Java, permettendo di usufruire a pieno della vasta mole di librerie ed *API*$_G$ fornite dal suddetto linguaggio;
- **Interesse del Gruppo**: il gruppo di lavoro, sin da subito, ha espresso la forte volontà di esplorare nuove tecnologie, non strettamente legate ai corsi universitari frequentati; tale scelta è dettata quindi anche per poter arricchire il numero delle competenze di ogni membro del gruppo.

Aspetti **negativi** considerati:

- **Sintassi**: proprio per consentire un minor numero di linee di codice a parità di risultato ottenuto con altri linguaggi, la sintassi può risultare spesso complessa e controintuitiva, soprattutto a chi è alle prime armi.

2.2 Play Framework

Play Framework è un framework *open-source*$_G$, adotta il pattern MVC ed è ottimizzato per la costruzione di applicazioni web sviluppate in Java o Scala.
Aspetti **positivi** considerati:

- **Incremento Produttività**: utilizzando nativamente il paradigma *Convention Over Configuration*$_G$ permette di velocizzare di molto lo sviluppo rendendo automatica l'implementazione di varie componenti a partire da altre definite dal programmatore, così facendo le varie configurazioni che si creano diventano molto più manutenibili, essendo generate dal framework. Rimane comunque possibile sovrascrivere e personalizzare tali configurazioni qualora lo si voglia, senza perdere quindi flessibilità e libertà di sviluppo;
- **Hit Refresh Workflow**: il framework implementa una funzionalità che consente di visualizzare i cambiamenti e le modifiche apportate semplicemente aggiornano la pagina del browser, gestendo quindi automaticamente una eventuale ricompilazione e riavvio dell'applicazione, velocizzando lo sviluppo di quest'ultima;
- **Templating html**: viene offerta la possibilità di scrivere codice html tramite un sistema di templating che utilizza il motore *Groovy*$_G$, così facendo è possibile gestire vari pezzi di codice con elementi della programmazione classica come espressioni condizionali ed ereditarietà tra più file;
- **Assets compiling**: è presente un compilatore per codice come *sass*$_G$ o *less*$_G$ che genera automaticamente il corrispettivo css;

- **Segnalazione degli Errori**: ogni errore rilevato nel codice viene riportato nella finestra del browser, specificando in maniera ottimale il tipo d'errore, la sua posizione nel codice, il percorso del file in cui risiede e varie informazioni utili in fase di sviluppo. L'insieme di questi elementi consentirà al team di mantenere un livello elevato di efficienza, in quanto tutti gli errori verranno individuati e corretti velocemente, senza perdere troppo tempo per identificarli;
- **Licenza Apache 2.0**: tale strumento viene rilasciato sotto la licenza *Apache 2.0*$_G$. Pertanto, a prodotto finito e qualora questo si inserisse nel mercato, il gruppo non sarebbe costretto a rilasciare anche il codice sorgente: è richiesto solamente il rilascio di un richiamo al fatto che l'applicazione sviluppata ha come punto di partenza uno strumento rilasciato sotto licenza *Apache 2.0*$_G$.

Aspetti **negativi** considerati:

- **Diffusione**: l'utilizzo di Play per la realizzazione di applicazioni web è molto inferiore ad altre alternative equivalenti, come ad esempio Ruby on Rails o Sails.js, ciò si riflette sulla quantità di guide, tutorial ed esempi di progetti da utilizzare per fare pratica con il framework e l'integrazione dello stesso con i moduli esterni scelti. Tali moduli esterni, sviluppati spesso da terzi, sono disponibili in un numero piuttosto ridotto e ciò limita la libertà di scelta anche in relazione alla documentazione spesso non chiara o con pochi esempi da seguire.
  Un ulteriore svantaggio è il passaggio dalla versione 2.3 alla 2.4, che influenza pesantemente la struttura del codice introducendo la Dependency Injection e costringendo a un refactoring delle classi. Ciò si riflette nel limitare ancora di più la possibilità di apprendimento attraverso esempi e moduli disponibili in rete, poichè molti di essi utilizzano la versione 2.3, la cui conversione a 2.4, necessita di varie modifiche.
  Il sistema di gestione delle dipendenze verso plugin e moduli esterni è gestito tramite SBT risulta comodo e pratico, tuttavia per quanto è stato possibile riscontrare fin'ora, non raggiunge i livelli di efficienza offerti da package managers equivalenti come NPM o RubyGems.

2.3 Amazon Web Services

Sono una collezione di servizi web che insieme creano una piattaforma di cloud computing. Le ragioni che ci hanno spinto a orientarci su tale servizio piuttosto che un competitor, ad esempio Heroku, sono le seguenti:

- **Flessibilità della piattaforma**: il gran numero di servizi offerti e la possibilità di "combinarli" liberamente fa di questa piattaforma uno strumento molto versatile che potremo sfruttare al massimo, tra cui Amazon Machine Learning che utilizzeremo per elaborare i dati e fornire previsione sugli stessi;
- **Esperienza del proponente**: Zero12 è partner di AWS e ci ha consigliato tale piattaforma vista la loro esperienza positiva nell'utilizzo in altri progetti;

Non sono stati riscontrati aspetti negativi degni di nota verso tale suite di servizi, se non l'inesperienza del gruppo verso gli stessi, ma tale situazione è estesa anche verso altri competitors con un offerta equivalente.

2.4 Node.js

Node.js è un *framework*$_G$ per lo sviluppo di applicazioni web *server-side*$_G$ che utilizza il motore javascript v8 di Google. Nel prodotto che si andrà a definire, Node.js verrà utilizzato per la realizzazione del *broker MQTT*$_G$, ovvero il sistema che si occuperà di gestire la ricezione dei dati da parte degli oggetti e di registrarli nella base di dati. La scelta di utilizzare la seguente tecnologia è dettata dalle seguenti osservazioni:

- **Efficienza**: in una situazione ideale, il sistema che verrà prodotto dovrà essere in grado di gestire un elevato traffico di rete (invio di dati da parte di moltissimi oggetti); pertanto l'efficienza deve essere sin da subito un punto di fondamentale importanza. Tale caratteristica è garantita dall'ambiente Node.js, il quale, basandosi essenzialmente su un paradigma di elaborazione dati in maniera asincrona, fa dell'efficienza il suo punto di forza maggiore;

- **Interesse del Gruppo**: medesime motivazioni che hanno spinto alla scelta di Scala.

Non sono stati riscontrati aspetti negativi degni di nota.

2.5 MongoDB

Un database orientato ai documenti, classificato come NoSQL, infatti al posto delle tradizionali tabelle MongoDB usa una struttura simile al formato JSON con una schemi dinamici, detti BSON. Abbiamo optato per tale database per i seguenti motivi:

- **NoSQL**: una struttura di questo tipo è molto più flessibile e semplice da modellare e progettare e risulta più semplice anche da scalare orizzontalmente, scenario più che plausibile vista la potenziale mole di dati che verranno salvati;
- **Ad hoc queries**: le queries sui dati memorizzati possono ritornare specifici campi dei documenti salvati e possono includere delle funzioni javascript che verranno mandate al database per essere eseguite, permettendo agli sviluppatori la scrittura di queries più "eleganti";
- **Esperienza del proponente**: Zero12 è partner di MongoDB è ha pertanto consigliato al gruppo tale tecnologia che utilizza quotidianamente e con la quale ha avuto ottimi riscontri.

Aspetti **negativi** considerati:

- **NoSQL**: se come detto una struttura noSQL risulta più flessibile, è altrettanto vero che richiede una progettazione più attenta mirata al capire quando utilizzare relazioni verso altre collections, piuttosto che preferire una struttura embedded.

2.6 HTML5

Quinta versione del linguaggio di markup per il web, introduce nuovi elementi tag e semplifica la sintassi rispetto a xhtml, introduce nuove *api*$_G$ come l'interfaccia drag and drop. Nonostante non sia ancora standard W3C abbiamo scelto di utilizzarlo poichè i vincoli rispetto ai browser lo consentono e questa ultima versione del markup permette l'implementazione di varie funzionalità e l'interazione con numerose librerie ed API.

2.7 SASS

Syntactically Awesome Stylesheets è un linguaggio di scripting interpretato nel comune CSS. La versione più recente di tale sintassi ricalca le principale regole sintattiche dei file css classici, ma porta dei vantaggi quali la possibilità di utilizzo di variabili, mixins, ereditarietà, annidamento, che rendono lo sviluppo del codice molto più modulabile e riutilizzabile. Va comunque detto che tale sintassi si fa apprezzare per lo più in progetti corposi, ma può risultare poco intuitiva per chi è alle prime armi.

2.8 Twitter Bootstrap

Twitter Bootstrap è uno dei framework più famosi per lo sviluppo di progetti Web. Esso contiene modelli di progettazione basati su HTML e CSS (con alcune estensioni opzionali che prendono come riferimento JavaScript), sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, bottoni e navigazione. La motivazioni che hanno portato alla scelta dei suddetti strumenti sono le seguenti:

- **Supporto nativo al *responsive web design*$_G$**: tale caratteristica garantisce che il layout delle pagine web si regoli dinamicamente, tenendo conto delle caratteristiche del dispositivo utilizzato, sia esso desktop, tablet o telefono cellulare. Grazie a tale caratteristica si va a sgravare il lavoro del gruppo, evitando di concentrare troppo tempo per la realizzazione di un'interfaccia grafica completamente *responsive*, fornita direttamente dalla tecnologia in esame;

- **Documentazione e supporto della comunità**.

Non sono stati riscontrati aspetti negativi degni di nota per tale framework.

2.9 Javascript

Linguaggio di scripting più diffuso in ambito web, si integra con tutti i framework che verranno utilizzati ed è per tanto indispensabile.

3 Definizione del Prodotto

3.1 Metodo e formalismo di specifica

La piattaforma che verrà realizzata è composta da più moduli indipendenti tra loro che uniti insieme danno vita al progetto vero e proprio. A valle di questa considerazione, è possibile scomporre l'architettura del sistema nei seguenti moduli:

- Business Logic (core della piattaforma);
- Engine Predittivo API;
- Frond End applicativo;
- Authentication API;
- Broker MQTT;

Il progetto verrà sviluppato attraverso il framework Play!, il quale impone l'uso del pattern architetturale MVC, che per tanto influenza fortemente l'organizzazione e la progettazione architetturale dell'applicazione. Play! propone due strati principali, il *Presentation layer*, che si suddivide a sua volta in *Controller layer* e *View layer*, e il *Model layer*; la progettazione seguirà quindi tale schema integrando, dove possibile, moduli già disponibili e consigliati per interagire al meglio con Play! framework. La progettazione è stata sia del tipo top-down, per quanto riguarda l'organizzazione secondo modello MVC, in particolare riguardo a come adattare l'applicazione all'anatomia di Play!, ma anche del tipo bottom-up, poichè da subito abbiamo cercato moduli e API da riutilizzare e integrare all'interno del progetto.

3.2 Architettura generale

Come detto l'architettura segue il modello imposto da Play! rappresentato dal seguente diagramma:

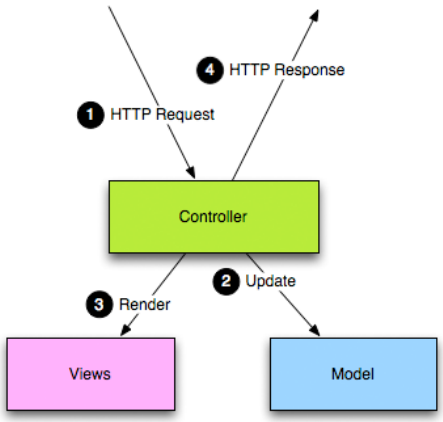

Figura 1: Play! framework MVC model.

Play! è inoltre completamente stateless e orientato unicamente a un approccio del tipo richiesta-risposta HTTP, il ciclo di vita tipico è quindi composto da una serie di richieste HTTP che seguono questo schema:

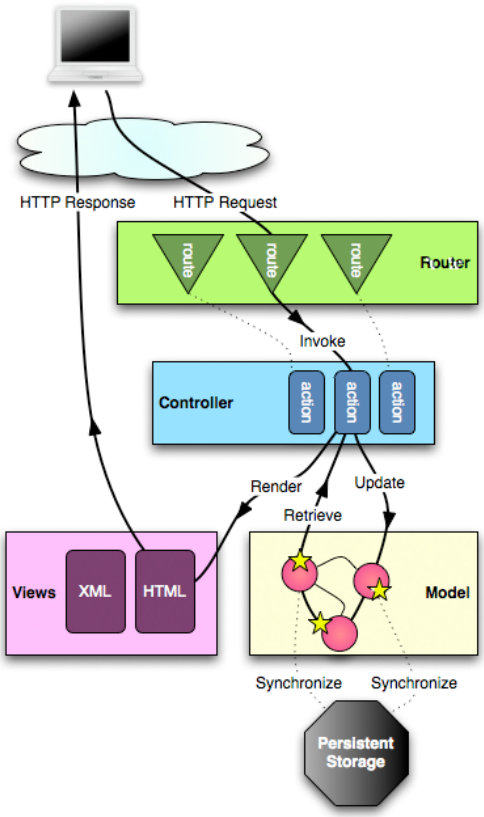

Figura 2: Ciclo di vita Play! framework.

Alla luce di tali premesse, l'architettura che ne deriva è la seguente:

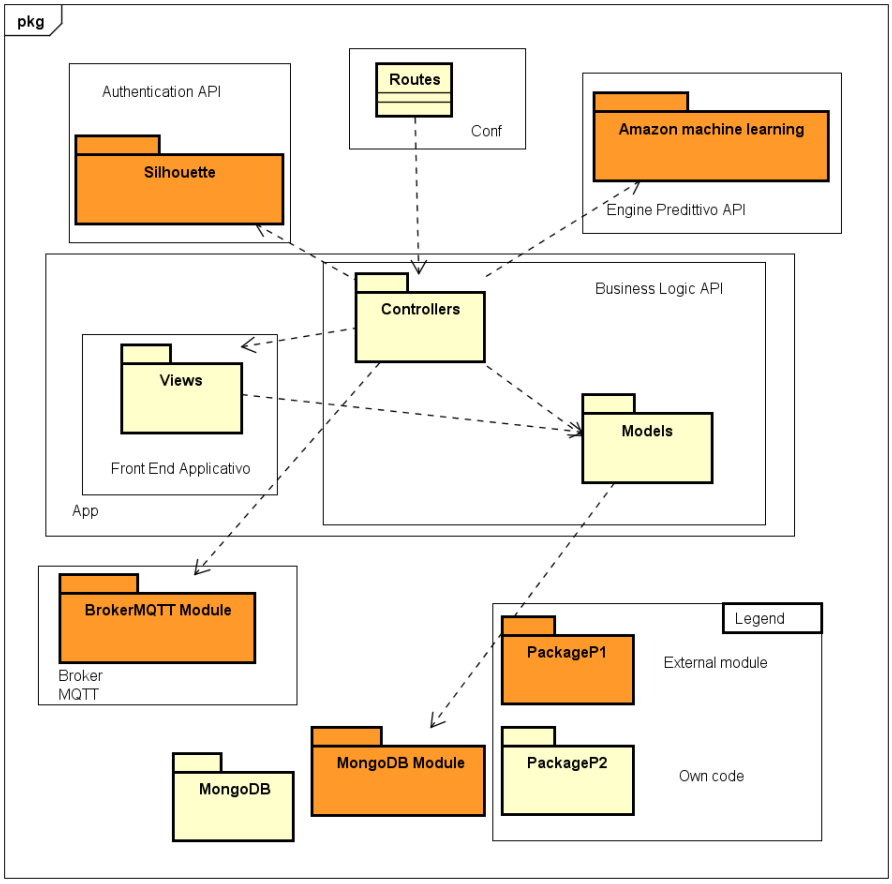

Figura 3: Architettura Generale.

Di seguito vi è una breve spiegazione dei vari package di cui ci compone il diagramma precedente.

- **Business Logic**
  - *Controllers*: un package che contiene tutte le azioni da intraprendere al verificarsi di un determinato evento, o meglio in risposta a una richiesta HTTP, faranno uso dell' **Authentication API** e dell'**Engine predittivo** e ovviamente si occuperanno dell'interazione tra *Views* e *Model*, sfruttando il modulo *Reactive Mongo* per l'interazione con la base di dati;
  - *Models*: contiene le classi che modellano le varie strutture dati, le operazioni su di essi e le funzionalità relative alla base di dati, che si appoggiano anch'esse al modulo *Reactive Mongo*.
- **Engine Predittivo**: è il modulo che si occupa di fare predizioni e elaborazione dei dati raccolti, utilizza il servizio di AWS *Amazon Machine Learning*;
- **Front End applicativo**: rappresentato di fatto dalla componente *Views* del pattern MVC di Play!, tale package contiene infatti tutti i template html utili a creare un'interfaccia grafica che faciliti l'interazione con la piattaforma;
- **Authentication API**: il modulo che gestisce il meccanismo dell'autenticazione, creazione e distruzione delle sessioni, si appoggia al modulo *play2-auth*;
- **Broker MQTT**: rappresenta il mediatore che gestisce lo scambio di informazione tra gli oggetti fisici in uso e la piattaforma stessa, è implementato tramite il servizio di AWS;

- **Routes**: è il file che si occupa di mappare l'esecuzione di un determinato metodo di un dato controller a una corrispondente richiesta HTTP.

4 Descrizione Componenti e Classi

In tale sezione verranno illustrate le principali caratteristiche di tutte le classi e di tutti i moduli esterni presenti nel progetto UMAP. Per convenzione, le classi verranno elencate precedute dalla locazione fisica (package) in cui sono contenute, in modo da fornire un rapido strumento per reperirle.

4.1 Moduli Esterni

Il progetto UMAP prevede di utilizzare i seguenti moduli/librerie esterni:

- **Amazon Machine Learning (AWS SDK)**;
- **Silhouette**;
- **MongoDB Module**;
- **BrokerMQTT Module**.

Tale scelta è stata effettuata per alleggerire il lavoro nelle attività di Progettazione e di Codifica, in modo da poter rilasciare il prodotto finale il prima possibile. Inoltre, si è rilevato che i suddetti moduli svolgano i propri compiti nella maniera ideale per gli obiettivi prefissati nel progetto didattico del gruppo di lavoro LaTeXeBiscotti.

4.1.1 Amazon Machine Learning (AWS SDK)

4.1.1.1 Descrizione

Tale tecnologia fa parte della collezione di servizi di *cloud computing*$_G$ che compongono la piattaforma "on demand" *Amazon Web Services*$_G$ offerta dall'azienda Amazon. Tale modulo permette di creare modelli di *machine learning* senza aver bisogno di conoscere la complessità e le tecnologie che stanno alla base di tale modulo. Una volta creati i suddetti modelli, Amazon Machine Learning permette di ottenere facilmente delle predizioni usando delle semplici *API*$_G$, con il vantaggio notevole di non dover implementare codice e di non dover gestire l'infrastruttura di base per concepirle.
Per una spiegazione più dettagliata del modulo si rimanda a:

https://aws.amazon.com/it/machine-learning/

4.1.1.2 Utilizzo

Tale modulo è utilizzato dal package **models/services**, in particolare dalle classi che hanno il compito della gestione dei dati e delle regole predittive; una volta inviati i dati al modulo, questo ritornerà un pacchetto di previsioni, correlato da grafici significativi che potranno essere utilizzati dagli utenti autorizzati. Il suddetto servizio avrà il compito di importare al loro interno, e quindi utilizzare i seguenti package:

- com.amazonaws.services.machinelearning.AmazonMachineLearningClient;
- com.amazonaws.services.machinelearning.model._;
- com.amazonaws.services.machinelearning.AmazonMachineLearningClient;
- com.amazonaws.services.machinelearning.model.{GetMLModelRequest, PredictRequest}.

4.1.2 Silhouette

4.1.2.1 Descrizione

In tale modulo viene gestita tutta la logica specifica per le operazioni di autenticazione e di riconoscimento del ruolo dell'utente autenticato. In generale tale modulo fornisce all'utente un *token* per tutta la durata

della sessione all'interno della piattaforma, tale *token* sarà utile anche per il riconoscimento del suo ruolo all'interno della stessa.

4.1.2.2 Utilizzo

Tale modulo è utilizzato dal package **controllers**, in particolare dalle classi che hanno il compito di gestire le operazioni di, ad esempio, login o di logout, e dalle classi che devono indirizzare l'utente a views differenti in base al suo ruolo all'interno della piattaforma.

4.1.3 MongoDB Module

4.1.3.1 Descrizione

Tale modulo si occupa di interfacciarsi nella maniera più efficace ed efficiente possibile con la base di dati.

4.1.3.2 Utilizzo

Tale modulo è utilizzato dai packages **controllers** e **models**, in particolare nelle operazioni di lettura e scrittura effettuate dalla varie classi sulla base di dati.

4.1.4 BrokerMQTT Module

4.1.4.1 Descrizione

Il suddetto modulo si fare da mediatore tra i vari messaggi provenienti dai vari *things* collegati con la piattaforma e viceversa.

4.1.4.2 Utilizzo

Il modulo viene utilizzato dal package **contollers**, il quale si occupa poi, nelle classi abilitate al compito, di inserire correttamente nella base di dati le informazioni ricevute dai *things* o di inviare update o comandi agli stessi.

4.2 Models

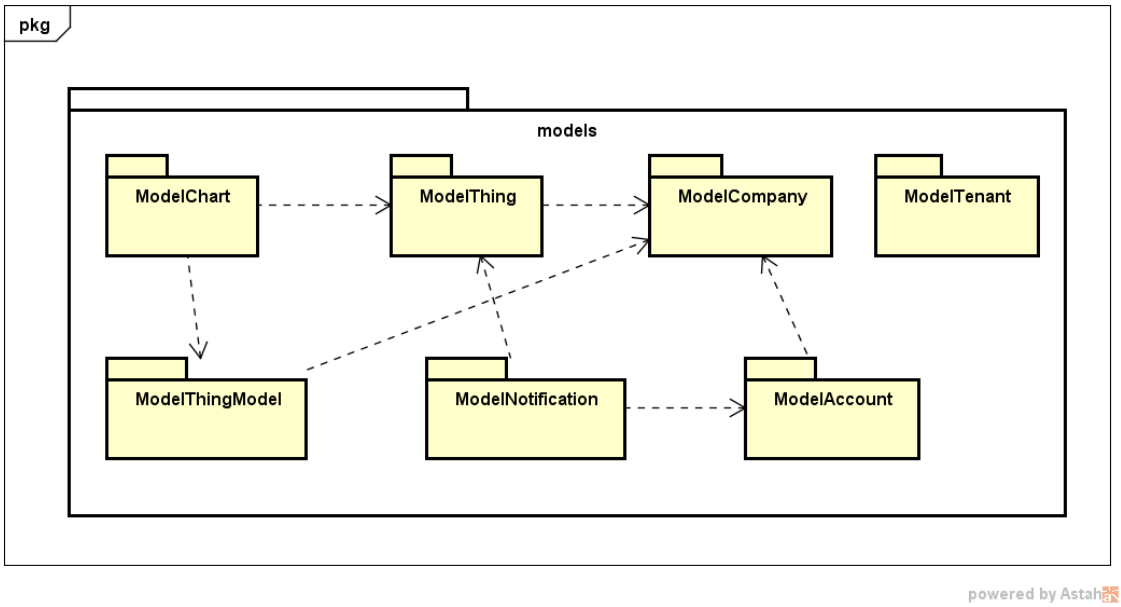

Figura 4: Diagramma package - Visione generale package Models

4.2.1 app :: models :: User

4.2.1.1 Descrizione

Classe contenente tutte le informazioni inerenti ad un qualsiasi utente inserito nella piattaforma. Tali informazioni riguardano essenzialmente, il codice univoco, l'anagrafica dell'utente con la sua email di riferimento e la company alla quale egli appartiene.

4.2.1.2 Utilizzo

Tale classe viene utilizzata per modellare nel database tutti i campi dati riguardanti un account. Essa riceve richieste di modifica, aggiunta, eliminazione e restituzione dei dati da parte del controller.

4.2.1.3 Relazioni con altre classi

- **app :: models :: Company**: relazione entrante, riferimento alla company a cui appartiene un particolare account;
- **app :: controllers :: superAdmin :: SuperAdminController**: relazione uscente, usa la classe User per ricavare le informazioni del proprio account;
- **app :: controllers :: superAdmin :: UserController**: relazione uscente, usa la classe User durante le operazioni CRUD sugli utenti;
- **app :: controllers :: admin :: AdminController**: relazione uscente, usa la classe User per ricavare le informazioni del proprio account;
- **app :: controllers :: admin :: UserController**: relazione uscente, usa la classe User durante le operazioni CRUD sugli utenti;

- **app :: controllers :: user :: UserController**: relazione uscente, usa la classe User per ricavare le informazioni del proprio account;
- **app :: controllers :: shared :: authentication :: AuthenticationController**: relazione uscente, usa le informazioni della classe User durante l'autenticazione;
- **app :: controllers :: shared :: account :: AccountController**: relazione uscente, utilizza la classe User per visualizzare le informazioni del proprio account e per gestirne le modifiche;
- **app :: controllers :: shared :: account :: PasswordRecoverController** relazione uscente, utilizza la classe User per ricavare le informazioni riguardo l'account interessato al recupero password;
- **app :: controllers :: shared :: account :: PasswordResetController** relazione uscente, utilizza la classe User per ricavare le informazioni riguardo l'account interessato al reset password;
- **app :: views :: superAdmin :: home**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: user :: users**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: user :: addUser**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: user :: editUser**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: admin :: home**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: admin :: user :: users**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: admin :: user :: addUser**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: admin :: user :: editUser**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: user :: home**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;
- **app :: views :: shared :: account :: editAccount**: relazione entrante, utilizza la classe User per mostrare informazioni raccolte in tale model;

4.2.2 app :: models :: Company

4.2.2.1 Descrizione

Classe contenente tutte le informazioni inerenti ad una company inserita nella piattaforma. Per ogni company sono di interesse, ad esempio, il nome, la ragione sociale e l'indirizzo di ubicazione.

4.2.2.2 Utilizzo

Tale classe viene utilizzata per modellare nel database tutti i campi dati riguardanti una company. Essa riceve richieste di modifica, aggiunta, eliminazione e restituzione dei dati da parte del controller.

4.2.2.3 Relazioni con altre classi

- **app :: controllers :: superAdmin :: CompanyController** relazione uscente, usa la classe Company durante le operazioni CRUD sulle companies;

- **app :: views :: superAdmin :: company :: companies**: relazione entrante, utilizza la classe Company per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: company :: addCompany**: relazione entrante, utilizza la classe Company per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: company :: editCompany**: relazione entrante, utilizza la classe Company per mostrare informazioni raccolte in tale model;

4.2.3 app :: models :: Thing

4.2.3.1 Descrizione

Classe contenente tutte le informazioni inerenti ad un oggetto (o *thing*) inserito nella piattaforma. Per ogni *thing* importano, ad esempio, il codice univoco, il serial number, la data dell'ultimo aggiornamento, la password utile per inviare dati al database, il suo stato (se in funzione o meno) e la company a cui esso appartiene.

4.2.3.2 Utilizzo

Tale classe viene utilizzata per modellare nel database tutti i campi dati riguardanti un *thing*. Essa riceve richieste di modifica, aggiunta, eliminazione (da un utente *Super Admin*$_G$) e restituzione (da un utente *Super Admin*$_G$ o *Admin*$_G$) dei dati da parte del controller.

4.2.3.3 Relazioni con altre classi

- **app :: models :: Company**: relazione entrante, riferimento alla company a cui appartiene un particolare oggetto;
- **app :: controllers :: superAdmin :: ThingController**: relazione uscente, usa la classe Thing durante le operazioni CRUD sugli oggetti;
- **app :: controllers :: admin :: ThingController**: relazione uscente, usa la classe Thing per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: user :: ThingController**: relazione uscente, usa la classe Thing per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: shared :: administrator :: EngineController**: relazione uscente, usa la classe Thing per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: views :: superAdmin :: thing :: things**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: thing :: addThing**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: thing :: editThing**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: admin :: thing :: thingDetails**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: user :: thing :: thingDetails**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: adminUser :: thing :: thingDetails**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;

4.2.4 app :: models :: ThingType

4.2.4.1 Descrizione
Classe contenente tutte le informazioni inerenti ai modelli ovvero le tipologie di *things* inseriti nella piattaforma. Per ogni modello importano, ad esempio, il nome, la marca, il codice univoco del modello, la company da cui il modello in questione è stato creato e la lista di Company che lo utilizzano.

4.2.4.2 Utilizzo
Tale classe viene utilizzata per modellare nel database tutti i campi dati riguardanti un modello di *things*. Essa riceve richieste di modifica, aggiunta, eliminazione e restituzione dei dati da parte del controller.

4.2.4.3 Relazioni con altre classi

- **app :: models :: Company**: relazione entrante, riferimento alla company a cui appartiene un particolare modello;
- **app :: controllers :: superAdmin :: ThingController**: relazione uscente, usa la classe ThingType durante le operazioni CRUD sugli oggetti;
- **app :: controllers :: admin :: ThingTypeController**: relazione uscente, usa la classe ThingType per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: admin :: ChartController**: relazione uscente, usa la classe ThingType per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: user :: ThingTypeController**: relazione uscente, usa la classe ThingType per le operazioni che ne vanno a leggerelo stato;
- **app :: controllers :: shared :: administrator :: EngineController**:relazione uscente, usa la classe ThingType durante le operazioni CRUD sugli oggetti;
- **app :: views :: superAdmin :: thingType :: thingTypes**: relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: thingType :: addThingType**: relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model;
- **app :: views :: superAdmin :: thingType :: editThingType**: relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model;
- **app :: views :: admin :: thingType :: thingTypeDetails**: relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model;
- **app :: views :: user :: thingType :: thingTypeDetails**: relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model;
- **app :: views :: adminUser :: thingType** : relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model.

4.2.5 app :: models :: Notification

4.2.5.1 Descrizione
Classe contenente tutte le informazioni inerenti ad una notifica inserita da un utente $user_G$ all'interno della piattaforma. Tali informazioni riguardano essenzialmente il suo codice univoco, una descrizione della notifica, l'oggetto o modello scatenante, l'indirizzo email dell'utente a cui inviare la notifica e il tipo di valore su cui inserire valori minimi e massimi, che verranno controllati e utilizzati come soglia per un possibile invio di notifiche.

4.2.5.2 Utilizzo

Tale classe viene utilizzata per modellare nel database tutti i campi dati riguardanti una notifica. Essa riceve richieste di modifica, aggiunta, eliminazione e restituzione dei dati da parte del controller, e notifica alla vista quando gli stessi sono disponibili o aggiornati.

4.2.5.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, recupero dell'indirizzo email a cui inviare la notifica;
- **app :: models :: Thing**: relazione entrante, recupero dei campi dati di un *thing* da cui far scaturire la notifica (se ad esempio tali dati superano un particolare limite);
- **app :: models :: ThingType**: relazione entrante, recupero dei campi dati di un modello da cui far scaturire la notifica (se ad esempio tali dati superano un particolare limite);
- **app :: controllers :: shared :: adminUser :: NotificationController**: relazione uscente, usa la classe Notification durante le operazioni CRUD sugli oggetti;
- **app :: views :: shared :: adminUser :: notifications**: relazione entrante, utilizza la classe Notification per mostrare informazioni raccolte in tale model;
- **app :: views :: shared :: adminUser :: addNotification**: relazione entrante, utilizza la classe Notification per mostrare informazioni raccolte in tale model;
- **app :: views :: shared :: adminUser :: editNotification**: relazione entrante, utilizza la classe Notification per mostrare informazioni raccolte in tale model;

4.2.6 app :: models :: Charts

4.2.6.1 Descrizione

Tale classe viene utilizzata per modellare i grafici utilizzati poi per la rappresentazione delle analisi predittive. Tali informazioni riguardano essenzialmente il codice univoco del grafico, il suo nome e la lista di oggetti o modelli che lo utilizzano.

4.2.6.2 Utilizzo

Verrà utilizzato per fornire le informazioni necessarie alle classi che si occupano di visualizzare i modelli di analisi e relativi grafici attualmente salvati nella base di dati.

4.2.6.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, riferimento alla classe che rappresenta un oggetto;
- **app :: models :: ThingType**: relazione entrante, riferimento alla classe che rappresenta un modello d'oggetto;
- **app :: controllers :: admin :: ThingController**: relazione uscente, usa la classe Charts per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: admin :: ThingTypeController**: relazione uscente, usa la classe Charts per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: admin :: ChartController**: relazione uscente, usa la classe Charts per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: user :: ThingController**: relazione uscente, usa la classe Charts per le operazioni che ne vanno a leggere o modificare lo stato;

- **app :: controllers :: user :: ThingTypeController**: relazione uscente, usa la classe Charts per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: controllers :: shared :: administrator :: EngineController**: relazione uscente, usa la classe Charts per le operazioni che ne vanno a leggere o modificare lo stato;
- **app :: views :: admin :: thing :: thingDetails**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: user :: thing :: thingDetails**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: adminUser :: thing :: thingDetails**: relazione entrante, utilizza la classe Thing per mostrare informazioni raccolte in tale model;
- **app :: views :: admin :: thingType :: thingTypeDetails**: relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model;
- **app :: views :: user :: thingType :: thingTypeDetails**: relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model;
- **app :: views :: adminUser :: thingType** : relazione entrante, utilizza la classe ThingType per mostrare informazioni raccolte in tale model.

4.2.7 app :: models :: Roles

4.2.7.1 Descrizione
Tale classe viene utilizzata per gestire l'autorizzazione dei vari tipi d'utente durante la navigazione nel sito, permettendo loro la visualizzazione solo delle zone di loro competenza.

4.2.7.2 Utilizzo
Verrà utilizzato per verificare il tipo d'utente dopo una richiesta di visualizzazione di una pagina dell'applicativo e in base ad esso sara possibile visualizzarla o altrimenti verrà rendirizzato alla pagina di competenza.

4.2.7.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, riferimento alla classe che rappresenta i vari utenti all'interno della piattaforma;

4.2.8 app :: models :: daos :: chart :: ChartDAO

4.2.8.1 Descrizione
Tale classe viene utilizzata per descrizione delle varie operazioni effettuabili nel database per quanto riguarda la classe Chart, come le ricerche o il salvataggio di un nuovo grafico.

4.2.8.2 Utilizzo
Verrà utilizzato per la gestione del database per quanto riguarda le operazioni sui grafici.

4.2.8.3 Relazioni con altre classi

- **app :: models :: Chart**: relazione entrante, riferimento alla classe che rappresenta il grafico;

- **app :: controllers :: admin :: ThingController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: admin :: ThingTypeController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: admin :: ChartController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: user :: ThingController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: user :: ThingTypeController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: shared :: administrator :: EngineController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database.

4.2.9 app :: models :: daos :: chart :: ChartDAOImpl

4.2.9.1 Descrizione
Tale classe viene utilizzata per realizzare l'implementazione dei metodi descritti nella classe ChartDAO riguardati le varie operazioni effettuabili nel database per quanto riguarda la classe Chart, come le ricerche o il salvataggio di un nuovo grafico.

4.2.9.2 Utilizzo
Verrà utilizzato per l'implementazione dei metodi utilizzati dal controller per la gestione del database per quanto riguarda le operazioni sui grafici.

4.2.9.3 Relazioni con altre classi

- **app :: models :: Chart**: relazione entrante, riferimento alla classe che rappresenta il grafico;
- **app :: models :: daos :: ChartDAO**: relazione entrante, la classe presenta l'implementazione della firma esposta.

4.2.10 app :: models :: daos :: company :: CompanyDAO

4.2.10.1 Descrizione
Tale classe viene utilizzata per descrizione delle varie operazioni effettuabili nel database per quanto riguarda la classe Company, come le ricerche o il salvataggio di una nuova company.

4.2.10.2 Utilizzo
Verrà utilizzato per la gestione del database per quanto riguarda le operazioni sulle company.

4.2.10.3 Relazioni con altre classi

- **app :: models :: Company**: relazione entrante, riferimento alla classe che rappresenta la company;
- **app :: controllers :: superAdmin :: CompanyController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database.

4.2.11 app :: models :: daos :: company :: CompanyDAOImpl

4.2.11.1 Descrizione
Tale classe viene utilizzata per realizzare l'implementazione dei metodi descritti nella classe CompanyDAO riguardati le varie operazioni effettuabili nel database per quanto riguarda la classe Company, come le ricerche o il salvataggio di una nuova company.

4.2.11.2 Utilizzo
Verrà utilizzato per l'implementazione dei metodi utilizzati dal controller per la gestione del database per quanto riguarda le operazioni sulle company.

4.2.11.3 Relazioni con altre classi

- **app :: models :: Company**: relazione entrante, riferimento alla classe che rappresenta la company;
- **app :: models :: daos :: CompanyDAO**: relazione entrante, la classe presenta l'implementazione della firma esposta.

4.2.12 app :: models :: daos :: thing :: ThingDAO

4.2.12.1 Descrizione
Tale classe viene utilizzata per descrizione delle varie operazioni effettuabili nel database per quanto riguarda la classe Thing, come le ricerche o il salvataggio di un nuovo oggetto.

4.2.12.2 Utilizzo
Verrà utilizzato per la gestione del database per quanto riguarda le operazioni sugli oggetti.

4.2.12.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, riferimento alla classe che rappresenta l'oggetto;
- **app :: controllers :: superAdmin :: ThingController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: admin :: ThingController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: admin :: ChartController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: user :: ThingController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: shared :: administrator :: EngineController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database.

4.2.13 app :: models :: daos :: thing :: ThingDAOImpl

4.2.13.1 Descrizione
Tale classe viene utilizzata per realizzare l'implementazione dei metodi descritti nella classe ThingDAO riguardati le varie operazioni effettuabili nel database per quanto riguarda la classe thing, come le ricerche o il salvataggio di un nuovo oggetto.

4.2.13.2 Utilizzo
Verrà utilizzato per l'implementazione dei metodi utilizzati dal controller per la gestione del database per quanto riguarda le operazioni sugli oggetti.

4.2.13.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, riferimento alla classe che rappresenta l'oggetto;
- **app :: models :: daos :: ThingDAO**: relazione entrante, la classe presenta l'implementazione della firma esposta.

4.2.14 app :: models :: daos :: thingType :: ThingTypeDAO

4.2.14.1 Descrizione
Tale classe viene utilizzata per descrizione delle varie operazioni effettuabili nel database per quanto riguarda la classe ThingType, come le ricerche o il salvataggio di un nuovo modello.

4.2.14.2 Utilizzo
Verrà utilizzato per la gestione del database per quanto riguarda le operazioni sui modelli.

4.2.14.3 Relazioni con altre classi

- **app :: models :: ThingType**: relazione entrante, riferimento alla classe che rappresenta il modello d'oggetto;
- **app :: controllers :: superAdmin :: ThingController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: admin :: ThingTypeController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: admin :: ChartController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: user :: ThingTypeController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: shared :: administrator :: EngineController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database.

4.2.15 app :: models :: daos :: thingtype :: ThingTypeDAOImpl

4.2.15.1 Descrizione
Tale classe viene utilizzata per realizzare l'implementazione dei metodi descritti nella classe ThingTypeDAO riguardati le varie operazioni effettuabili nel database per quanto riguarda la classe thingType, come le ricerche o il salvataggio di un nuovo modello.

4.2.15.2 Utilizzo
Verrà utilizzato per l'implementazione dei metodi utilizzati dal controller per la gestione del database per quanto riguarda le operazioni sui modelli.

4.2.15.3 Relazioni con altre classi

- **app :: models :: ThingType**: relazione entrante, riferimento alla classe che rappresenta il modello d'oggetto;
- **app :: models :: daos :: ThingTypeDAO**: relazione entrante, la classe presenta l'implementazione della firma esposta.

4.2.16 app :: models :: daos :: user :: UserDAO

4.2.16.1 Descrizione
Tale classe viene utilizzata per descrizione delle varie operazioni effettuabili nel database per quanto riguarda la classe User, come le ricerche o il salvataggio di un nuovo user.

4.2.16.2 Utilizzo
Verrà utilizzato per la gestione del database per quanto riguarda le operazioni sugli user.

4.2.16.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, riferimento alla classe che rappresenta lo user;
- **app :: controllers :: superAdmin :: UserController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: admin :: UserController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: shared :: authentication :: AuthenticationController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: shared :: account :: AccountController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: shared :: account :: PasswordRecoverdController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database;
- **app :: controllers :: shared :: account :: PasswordResetController**: relazione uscente, utilizza la classe per operazioni di lettura/scrittura con il database.

4.2.17 app :: models :: daos :: user :: UserDAOImpl

4.2.17.1 Descrizione
Tale classe viene utilizzata per realizzare l'implementazione dei metodi descritti nella classe UserDAO riguardati le varie operazioni effettuabili nel database per quanto riguarda la classe user, come le ricerche o il salvataggio di un nuovo user.

4.2.17.2 Utilizzo
Verrà utilizzato per l'implementazione dei metodi utilizzati dal controller per la gestione del database per quanto riguarda le operazioni sugli user.

4.2.17.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, riferimento alla classe che rappresenta lo user;
- **app :: models :: daos :: UserDAO**: relazione entrante, la classe presenta l'implementazione della firma esposta.

4.2.18 app :: models :: daos :: services :: user :: UserService

4.2.18.1 Descrizione
Tale classe viene utilizzata per la definizioni di metodi che servono alla libreria Silhouette per l'identificazione degli oggetti di tipo user e per il suo salvataggio.

4.2.18.2 Utilizzo
Verrà utilizzato per la gestione del database per quanto riguarda le operazioni sugli oggetti.

4.2.18.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, riferimento alla classe che rappresenta lo user;

4.2.19 app :: models :: daos :: services :: user :: UserServiceImpl

4.2.19.1 Descrizione
Tale classe viene utilizzata per l'implementazione dei metodi descritti nella classe UserService utili all'identificazione degli oggetti di tipo user e per il suo salvataggio.

4.2.19.2 Utilizzo
Verrà utilizzato per la gestione del database per quanto riguarda le operazioni sugli oggetti.

4.2.19.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, riferimento alla classe che rappresenta lo user;
- **app :: models :: daos :: services :: user :: UserService**: relazione entrante, la classe presenta l'implementazione della firma esposta.;

4.3 Controllers

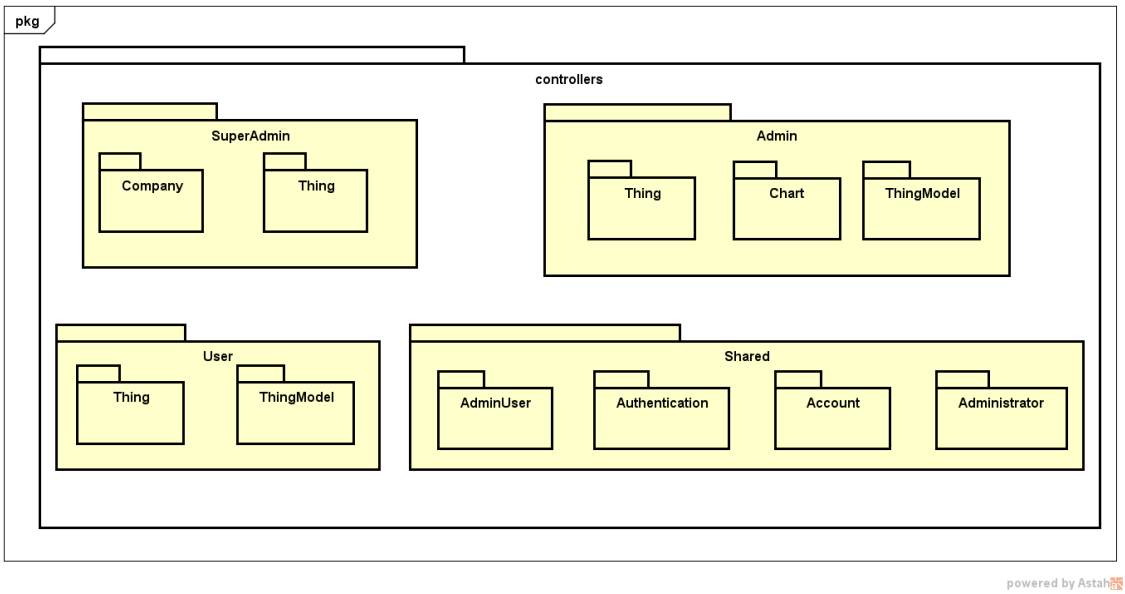

Figura 5: Diagramma package - Visione generale package Controllers

4.3.1 app :: controllers :: superAdmin :: SuperAdminController

4.3.1.1 Descrizione
Classe contenente un metodo che renderizza la pagina di home di un SuperAdmin.

4.3.1.2 Utilizzo
Tale classe viene utilizzata come classe di partenza per gestire le varie funzionalità offerte al SuperAdmin.

4.3.1.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: views :: superAdmin :: home**: relazione entrante, tale classe contiene l'interfaccia grafica relativa alla home di un SuperAdmin.

4.3.2 app :: controllers :: superAdmin :: UserController

4.3.2.1 Descrizione
Classe contenente tutti i comandi atti a cambiare lo stato del model rappresentante gli utenti iscritti alla piattaforma.

4.3.2.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a cambiare lo stato del model rappresentante gli utenti iscritti alla piattaforma. Tali comandi sono, ad esempio, l'aggiunta e la modifica dell'account di un utente.

4.3.2.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: user :: UserDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe User;
- **app :: views :: superAdmin :: user :: editUser**: relazione entrante, tale classe viene usata per mostrare la form di modifica e lanciare la richiesta di edit di una dato User;
- **app :: views :: superAdmin :: user :: users**: relazione entrante, tale classe viene usata per mostrare la lista di tutti gli Users con relativa interfaccia per lanciare le classiche operazioni CRUD;
- **app :: views :: superAdmin :: user :: addUser**: relazione entrante, tale classe viene usata per mostrare la form di aggiunta di un nuovo User.

4.3.3 app :: controllers :: superAdmin :: CompanyController

4.3.3.1 Descrizione
Classe contenente tutti i comandi atti a cambiare lo stato del model rappresentante le company.

4.3.3.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a cambiare lo stato del model rappresentante le company. Tali comandi sono, ad esempio, l'aggiunta e la modifica di una company.

4.3.3.3 Relazioni con altre classi

- **app :: models :: Company**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: company :: CompanyDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Company;
- **app :: views :: superAdmin :: company :: editCompany**: relazione entrante, tale classe viene usata per mostrare la form di modifica e lanciare la richiesta di edit di una data company;
- **app :: views :: superAdmin :: company :: companies**: relazione entrante, tale classe viene usata per mostrare la lista di tutte le company con relativa interfaccia per lanciare le classiche operazioni CRUD;
- **app :: views :: superAdmin :: company :: addCompany**: relazione entrante, tale classe viene usata per mostrare la form di aggiunta di una nuova company.

4.3.4 app :: controllers :: superAdmin :: ThingController

4.3.4.1 Descrizione
Classe contenente tutti i comandi atti a cambiare lo stato del model rappresentante i *things*.

4.3.4.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a cambiare lo stato del model rappresentante i *things*. Tali comandi sono, ad esempio, l'aggiunta e la modifica di un *things*.

4.3.4.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: ThingType**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: thing :: ThingDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Thing;
- **app :: models :: daos :: thingType :: ThingTypeDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe ThingType;
- **app :: views :: superAdmin :: thing :: editThing**: relazione entrante, tale classe viene usata per mostrare la form di modifica e lanciare la richiesta di edit di un thing;
- **app :: views :: superAdmin :: thing :: things**: relazione entrante, tale classe viene usata per mostrare la lista di tutte i things con relativa interfaccia per lanciare le classiche operazioni CRUD;
- **app :: views :: superAdmin :: thing :: addThing**: relazione entrante, tale classe viene usata per mostrare la form di aggiunta di un nuovo thing.
- **app :: views :: superAdmin :: thingType :: editThingType**: relazione entrante, tale classe viene usata per mostrare la form di modifica e lanciare la richiesta di edit di un thingType;
- **app :: views :: superAdmin :: thingType :: thingTypes**: relazione entrante, tale classe viene usata per mostrare la lista di tutte i thingsType con relativa interfaccia per lanciare le classiche operazioni CRUD;
- **app :: views :: superAdmin :: thingType :: addThingType**: relazione entrante, tale classe viene usata per mostrare la form di aggiunta di un nuovo thingType.

4.3.5 app :: controllers :: admin :: AdminController

4.3.5.1 Descrizione
Classe contenente un metodo che renderizza la pagina di home di un Admin.

4.3.5.2 Utilizzo
Tale classe viene utilizzata come classe di partenza per gestire le varie funzionalità offerte all'Admin.

4.3.5.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: views :: admin :: home**: relazione entrante, tale classe contiene l'interfaccia grafica relativa alla home di un Admin.

4.3.6 app :: controllers :: admin :: UserController

4.3.6.1 Descrizione
Classe contenente tutti i comandi atti a cambiare lo stato del model rappresentante gli utenti iscritti alla piattaforma.

4.3.6.2 Utilizzo

Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a cambiare lo stato del model rappresentante gli utenti iscritti alla piattaforma per la propria company. Tali comandi sono, ad esempio, l'aggiunta e la modifica dell'account di un utente.

4.3.6.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: user :: UserDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe User;
- **app :: views :: admin :: user :: editUser**: relazione entrante, tale classe viene usata per mostrare la form di modifica e lanciare la richiesta di edit di una dato User;
- **app :: views :: admin :: user :: users**: relazione entrante, tale classe viene usata per mostrare la lista di tutti gli Users con relativa interfaccia per lanciare le classiche operazioni CRUD;
- **app :: views :: admin :: user :: addUser**: relazione entrante, tale classe viene usata per mostrare la form di aggiunta di un nuovo User.

4.3.7 app :: controllers :: admin :: ThingController

4.3.7.1 Descrizione

Classe contenente tutti i comandi atti a cambiare lo stato del model rappresentante i *things*, in particolare per quanto riguarda l'aggiornamento del $firmware_G$ del *thing* stesso, o l'invio di comandi.

4.3.7.2 Utilizzo

Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a cambiare lo stato del model rappresentante i *things*, in particolare per quanto riguarda l'aggiornamento del $firmware_G$ del *thing* stesso, o l'invio di comandi.

4.3.7.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne o leggere lo stato;
- **app :: models :: daos :: thing :: ThingDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Thing;
- **app :: models :: Chart**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne o leggere lo stato;
- **app :: models :: daos :: chart :: ChartDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Chart;
- **app :: views :: shared :: adminUser :: things**: relazione entrante, tale classe visualizza la lista di Things in uso dalla propria company;
- **app :: views :: admin :: thing :: thingDetails**: relazione entrante, verso tale classe vengono notificate le informazioni relative allo stato del Thing, ai grafici che lo interessano e all'invio di comandi e aggiornamento firmware degli oggetti dello stesso tipo.

4.3.8 app :: controllers :: admin :: ThingTypeController

4.3.8.1 Descrizione Classe contenente tutti i comandi atti a cambiare lo stato del model rappresentante il tipo dei *things*, in particolare per quanto riguarda l'aggiornamento del $firmware_G$ del *thing* stesso.

4.3.8.2 Utilizzo Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a cambiare lo stato del model rappresentante le tipologie dei *things*, in particolare per quanto riguarda l'aggiornamento del $firmware_G$ del tipo stesso.

4.3.8.3 Relazioni con altre classi

- **app :: models :: ThingType**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a leggerne lo stato;
- **app :: models :: daos :: thingType :: ThingTypeDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe ThingType;
- **app :: models :: Chart**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne o leggere lo stato;
- **app :: models :: daos :: chart :: ChartDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Chart;
- **app :: views :: shared :: adminUser :: thingTypes**: relazione entrante, verso tale classe vengono notificate le informazioni sulle tipologie dei things in uso;
- **app :: views :: admin :: thingType :: thingTypeDetails**: relazione entrante, verso tale classe vengono notificate le informazioni relative allo stato del model, ai grafici che lo interessano e all'invio di comandi e aggiornamento firmware della tipologia di oggetto selezionato.

4.3.9 app :: controllers :: admin :: ChartController

4.3.9.1 Descrizione
Classe contenente tutti i comandi atti a creare un nuovo modello di analisi, predittiva o meno, combinando tipi di dati input, funzioni e grafici da visualizzare.

4.3.9.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller per costruire un modello di analisi, predittiva o meno, sulla base della configurazione di parametri scelti dall'utente tramite interfaccia grafica.

4.3.9.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a leggerne lo stato;
- **app :: models :: daos :: thing :: ThingDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Thing;
- **app :: models :: ThingType**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a leggerne lo stato;
- **app :: models :: daos :: thingType :: ThingTypeDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe ThingType;

- **app :: models :: Chart**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a leggerne lo stato;
- **app :: models :: daos :: chart :: ChartDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Chart;
- **app :: views ::admin :: chart :: newChart**: relazione entrante, verso tale classe vengono notificate le informazioni per permettere la scelta dei parametri con cui configurare il modello di analisi.

4.3.10 app :: controllers :: user :: UserController

4.3.10.1 Descrizione
Classe contenente un metodo che renderizza la pagina di home di uno User.

4.3.10.2 Utilizzo
Tale classe viene utilizzata come classe di partenza per gestire le varie funzionalità offerte dello User.

4.3.10.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: views :: user :: home**: relazione entrante, tale classe contiene l'interfaccia grafica relativa alla home di uno User.

4.3.11 app :: controllers :: user :: ThingController

4.3.11.1 Descrizione
Classe contenente tutti i comandi atti a leggere lo stato del model rappresentante i *things* appartenenti ad una particolare $Company_G$.

4.3.11.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a leggere lo stato del model rappresentante i *things* appartenenti ad una particolare $Company_G$.

4.3.11.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a leggerne lo stato;
- **app :: models :: daos :: thing :: ThingDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Thing;
- **app :: models :: Chart**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne o leggere lo stato;
- **app :: models :: daos :: chart :: ChartDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Chart;
- **app :: views :: shared :: adminUser :: things**: relazione entrante, verso tale classe saranno notificate le informazioni sui vari things della company in questione;

- **app :: views :: user :: thing :: thingDetails**: relazione entrante, verso tale classe saranno notificate le informazioni di uno specifico oggetto appartenente della company in questione.

4.3.12 app :: controllers :: user :: ThingTypeController

4.3.12.1 Descrizione
Classe contenente tutti i comandi atti a leggere lo stato del model rappresentante le tipologie dei *things* appartenenti ad una particolare $company_G$.

4.3.12.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a leggere lo stato del model rappresentante le tipologie dei *things* appartenenti ad una particolare $company_G$.

4.3.12.3 Relazioni con altre classi

- **app :: models :: ThingType**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a leggerne lo stato;
- **app :: models :: daos :: thingType :: ThingTypeDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe ThingType;
- **app :: models :: Chart**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne o leggere lo stato;
- **app :: models :: daos :: chart :: ChartDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Chart;
- **app :: views :: shared :: adminUser :: thingTypes**: relazione entrante, verso tale classe saranno notificate le informazioni sulle varie tipologie di oggetti appartenenti alla company in questione;
- **app :: views :: user :: thingType :: thingTypesDetails**: relazione entrante,verso tale classe saranno notificate le informazioni su uno specifico modello di oggetto, così come i grafici che lo interessano, appartenente alla company in questione.

4.3.13 app :: controllers :: shared :: adminUser :: NotificationController

4.3.13.1 Descrizione
Classe contenente tutti i comandi atti a modificare lo stato del model rappresentante le notifiche riguardanti un particolare *thing* appartenente alla company di riferimento.

4.3.13.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a leggere lo stato del model rappresentante le notifiche riguardanti un particolare *thing* appartenente alla company di riferimento.

4.3.13.3 Relazioni con altre classi

- **app :: models :: Notification**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;

- **app :: models :: daos :: notification :: NotificationDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Notification;
- **app :: views :: shared :: adminUser :: notification :: notification**: relazione entrante, verso tale classe vengono notificate le informazioni relative alle impostazioni di notifica salvate e in uso.
- **app :: views :: shared :: adminUser :: notification :: addNotification**: relazione entrante, tale classe renderizza l'interfaccia grafica per gestire l'aggiunta di una nuova notifica;
- **app :: views :: shared :: adminUser :: notification :: editNotification**: relazione entrante, tale classe renderizza l'interfaccia grafica per gestire la modifica di una nuova notifica esistente.

4.3.14 app :: controllers :: shared :: authentication :: AuthenticationController

4.3.14.1 Descrizione
Classe contenente tutti i comandi atti a gestire le operazioni di autenticazione di un particolare utente alla piattaforma.

4.3.14.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a gestire le operazioni di autenticazione di un particolare utente alla piattaforma.

4.3.14.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a verificarne lo stato (utente presente nella piattaforma e con le credenziali corrette per l'accesso o meno);
- **app :: models :: daos :: user :: UserDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe User;
- **app :: views :: shared :: authentication :: signIn**: relazione entrante, verso tale classe sono notificate le informazioni da visualizzare per procedere al login nella piattaforma;
- **app :: views :: shared :: authentication :: signOut**: relazione entrante, verso tale classe si riverbereranno gli esiti di un'operazione di logout eseguita da un qualsiasi punto della piattaforma.

4.3.15 app :: controllers :: shared :: account :: AccountController

4.3.15.1 Descrizione
Classe contenente tutti i comandi atti a modificare lo stato del model rappresentante gli account che hanno già accesso alla piattaforma.

4.3.15.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a modificare lo stato del model rappresentante gli account che hanno già acceduto alla piattaforma (operazioni di modifica profilo).

4.3.15.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;

- **app :: models :: daos :: user :: UserDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe User;
- **app :: views :: shared :: account :: editAccount**: relazione entrante, verso tale classe sono notificate le informazioni relative all'account in uso e per procedere alla modifica di questo.

4.3.16 app :: controllers :: shared :: account :: PasswordRecoverController

4.3.16.1 Descrizione
Classe contenente tutti i comandi atti a interagire con il model rappresentante gli account per un particolare utente che ha dimenticato o smarrito la password di accesso alla piattaforma.

4.3.16.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a leggere lo stato del model rappresentante gli account, in modo tale che l'utente abbia la possibilità di recuperare le credenziali di accesso alla piattaforma.

4.3.16.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a leggerne lo stato;
- **app :: models :: daos :: user :: UserDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe User;
- **app :: views :: shared :: authentication :: passwordRecover**: relazione entrante,verso tale classe sono notificate le informazioni utili per generare l'interfaccia grafica relativa al recupero password.

4.3.17 app :: controllers :: shared :: account :: PasswordResetController

4.3.17.1 Descrizione
Classe contenente tutti i comandi atti a modificare lo stato del model rappresentante gli account per un particolare utente che ha dimenticato o smarrito la password di accesso alla piattaforma.

4.3.17.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a modificare lo stato del model rappresentante gli account, in modo tale che l'utente abbia la possibilità di resettare le credenziali di accesso alla piattaforma.

4.3.17.3 Relazioni con altre classi

- **app :: models :: User**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: user :: UserDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe User;
- **app :: views :: Shared :: Authentication :: ResetPassword**: relazione entrante, verso tale classe sono notificate le informazioni per generare l'interfaccia grafica che fornisce il mezzo per effettuare il reset password.

4.3.18 app :: controllers :: shared :: administrator :: EngineController

4.3.18.1 Descrizione
Classe contenente tutti i comandi atti a cambiare lo stato del model rappresentante la configurazione del motore di analisi predittiva: il *Super Admin*$_G$ o Admin, potrà da qui filtrare, o meno, i dati provenienti dal *thing* di riferimento e le regole predittive che si potranno applicare a tali dati. La selezione di questi componenti sarà visibile all'utente *Admin*$_G$, che andrà quindi a combinarli secondo le sue necessità.

4.3.18.2 Utilizzo
Tale classe viene utilizzata per reagire alle richieste HTTP generate dalla view corrispondente a tale controller e per la gestione dei comandi atti a cambiare lo stato del model rappresentante la configurazione delle funzioni disponibili durante la creazione di grafici e del model relativo alla tipologia di oggetti (modello).

4.3.18.3 Relazioni con altre classi

- **app :: models :: Thing**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: thing :: ThingDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Thing;
- **app :: models :: ThingType**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: thingType :: ThingTypeDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe ThingType;
- **app :: models :: Charts**: relazione entrante, verso tale classe sono rivolte le operazioni che ne vanno a modificarne lo stato;
- **app :: models :: daos :: charts :: ChartsDAO**: relazione entrante, viene utilizzata tale classe per le operazioni che vanno a interessare la base di dati e la classe Chart;
- **app :: views :: superAdmin :: engine :: engine**: relazione entrante, verso tale classe vengono notificate le informazioni relative ai tipi di dati degli oggetti e funzioni disponibili durante la creazione di grafici per ogni Company;
- **app :: views :: admin :: engine :: editFunctions**: relazione entrante, verso tale classe vengono notificate le informazioni relative alle funzioni disponibili durante la creazione di grafici per la Company dell'Admin loggato;
- **app :: views :: admin :: engine :: editData**: relazione entrante, verso tale classe vengono notificate le informazioni relative ai tipi di dato disponibili durante la creazione di grafici per la Company dell'Admin loggato.

4.4 Views

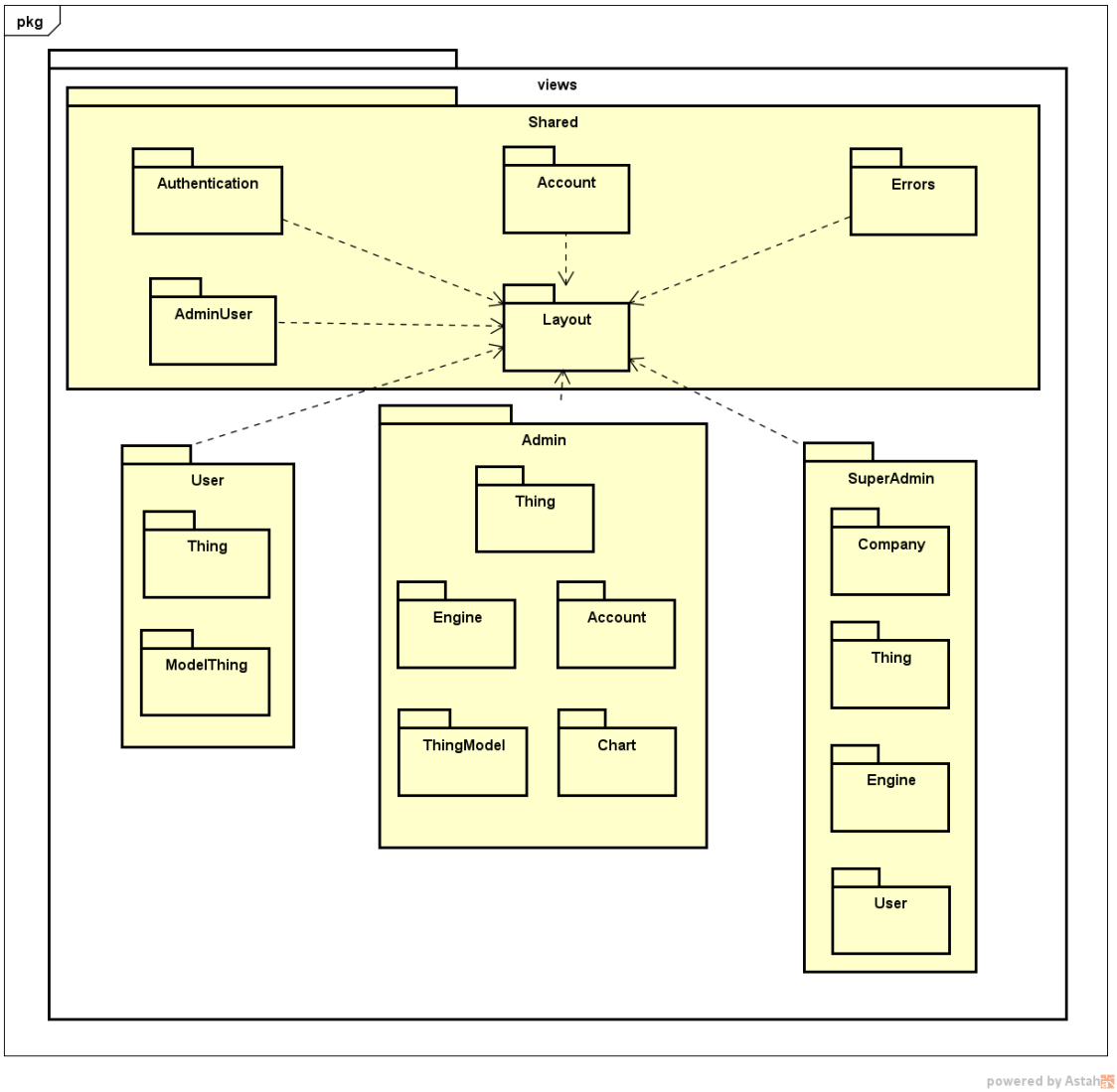

Figura 6: Diagramma package - Visione generale package Views

4.4.1 app :: views :: superAdmin :: home

4.4.1.1 Descrizione

Classe che visualizza la homepage per un utente di tipo SuperAdmin.

4.4.1.2 Utilizzo

Viene utilizzata dalla classe app :: controllers :: superAdmin :: SuperAdminController per visualizzare la pagina di home della proprio tipologia di account.

4.4.2 app :: views :: superAdmin :: user :: users

4.4.2.1 Descrizione
Classe che visualizza la lista con tutti gli utenti iscritti alla piattaforma e la relativa interfaccia per lanciare le classiche operazioni CRUD.

4.4.2.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: UserController per gestire le richieste di operazione relative agli utenti iscritti alla piattaforma.

4.4.3 app :: views :: superAdmin :: user :: addUser

4.4.3.1 Descrizione
Classe che visualizza la form necessaria ad aggiungere un nuovo utente alla piattaforma.

4.4.3.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: UserController per gestire le richieste di operazione relative agli utenti iscritti alla piattaforma.

4.4.4 app :: views :: superAdmin :: user :: editUser

4.4.4.1 Descrizione
Classe che visualizza la form necessaria a modificare un utente presente nella piattaforma.

4.4.4.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: UserController per gestire le richieste di operazione relative agli utenti iscritti alla piattaforma.

4.4.5 app :: views :: superAdmin :: company :: companies

4.4.5.1 Descrizione
Classe che visualizza la lista con tutte le company presenti nella piattaforma e le relative operazioni di CRUD disponibili.

4.4.5.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: CompanyController per visualizzare la pagina di gestione delle company.

4.4.6 app :: views :: superAdmin :: company :: addCompany

4.4.6.1 Descrizione
Classe che visualizza la form per aggiungere una nuova company alla piattaforma.

4.4.6.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: CompanyController per visualizzare la pagina di aggiunta di una nuova company.

4.4.7 app :: views :: superAdmin :: company :: editCompany

4.4.7.1 Descrizione
Classe che visualizza la form per modificare i dati di una company già presente nella piattaforma.

4.4.7.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: CompanyController per visualizzare la pagina di modifica di una company presente nella piattaforma.

4.4.8 app :: views :: superAdmin :: thing :: things

4.4.8.1 Descrizione
Classe che visualizza la lista con tutte i thing presenti nella piattaforma e le relative operazioni di CRUD disponibili.

4.4.8.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: ThingController per visualizzare la pagina di gestione dei thing.

4.4.9 app :: views :: superAdmin :: thing :: addThing

4.4.9.1 Descrizione
Classe che visualizza la form per aggiungere una nuova company alla piattaforma.

4.4.9.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: ThingController per visualizzare la pagina di aggiunta di un nuovo thing per una data company.

4.4.10 app :: views :: superAdmin :: thing :: editThing

4.4.10.1 Descrizione
Classe che visualizza la form per modificare i dati di un thing già presente nella piattaforma.

4.4.10.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: ThingController per visualizzare la pagina di modifica di un thing presente nella piattaforma.

4.4.11 app :: views :: superAdmin :: thingType :: thinTypes

4.4.11.1 Descrizione
Classe che visualizza la lista con tutte i tipi di thing presenti nella piattaforma e le relative operazioni di CRUD disponibili.

4.4.11.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: ThingTypeController per visualizzare la pagina di gestione dei tipi dei thing.

4.4.12 app :: views :: superAdmin :: thingType :: addThingType

4.4.12.1 Descrizione
Classe che visualizza la form per aggiungere un nuovo tipo di thing alla piattaforma.

4.4.12.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: ThingTypeController per visualizzare la pagina di aggiunta di un nuovo tipo di thing per una data company.

4.4.13 app :: views :: superAdmin :: thingType :: editThingType

4.4.13.1 Descrizione
Classe che visualizza la form per modificare i dati di un tipo di thing già presente nella piattaforma.

4.4.13.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: superAdmin :: ThingController per visualizzare la pagina di modifica di un tipo di thing presente nella piattaforma.

4.4.14 app :: views :: superAdmin :: engine :: engine

4.4.14.1 Descrizione
Classe che visualizza delle tabelle relative alle funzioni utilizzabili in fase di definizione di un modello di analisi e un interfaccia relativa a quali dati per ogni modello di oggetto sono effettivamente inoltrati alla company.

4.4.14.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: administrator :: EngineController per filtrare le funzioni che rendere disponibili a una data company, così come per i dati relativi ad ogni modello di oggetto ch vengono effettivamente inoltrati alla company.

4.4.15 app :: views :: admin :: thing :: thingDetails

4.4.15.1 Descrizione
Classe che visualizza una tabella contenente informazioni relative allo stato del Thing, ai grafici che lo interessano e all'invio di comandi e aggiornamento firmware degli oggetti dello stesso tipo.

4.4.15.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: ThingController per visualizzare la dei dettagli del things, dei grafici che lo interessano e della form per l'invio di comandi e update firmware.

4.4.16 app :: views :: admin :: thingType :: thingTypeDetails

4.4.16.1 Descrizione Classe che visualizza una tabella contenente informazioni relative allo stato del Thing, ai grafici che lo interessano e aggiornamento firmware per quella tipologia di oggetto.

4.4.16.2 Utilizzo Viene utilizzata dalla classe app :: controllers :: sdmin :: thingType :: ThingTypeController per visualizzare la pagina dei dettagli di un modello di thing, i grafici che lo interessano e la form per l'invio di update del firmware.

4.4.17 app :: views :: admin :: chart :: newChart

4.4.17.1 Descrizione
Classe che visualizza una pagina con la procedura a passaggi che permette di creare dei grafici personalizzati per un particolare modello di oggetti selezionando le funzioni, il tipo di grafico, i dati e come relazionarli tra loro.

4.4.17.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: chart :: ChartController per visualizzare la pagina con la procedura di creazione grafici.

4.4.18 app :: views :: admin :: engine :: editFunctions

4.4.18.1 Descrizione Classe che visualizza delle tabelle relative alle funzioni utilizzabili in fase di definizione di un modello di analisi.

4.4.18.2 Utilizzo Viene utilizzata dalla classe app :: controllers :: shared :: administrator :: EngineController per filtrare le funzioni che rende disponibili alla Company dell'Admin autenticato.

4.4.19 app :: views :: admin :: engine :: editData

4.4.19.1 Descrizione Classe che visualizza delle tabelle relative ai dati visualizzabili e utilizzabili poi in fase di definizione di un modello di analisi.

4.4.19.2 Utilizzo Viene utilizzata dalla classe app :: controllers :: shared :: administrator :: EngineController per filtrare i dati da rendere disponibili alla Company dell'Admin autenticato.

4.4.20 app :: views :: admin :: user :: users

4.4.20.1 Descrizione
Classe che visualizza la lista con tutti gli utenti iscritti alla piattaforma e la relativa interfaccia per lanciare le classiche operazioni CRUD.

4.4.20.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: UserController per gestire le richieste di operazione relative agli utenti iscritti alla piattaforma.

4.4.21 app :: views :: admin :: user :: addUser

4.4.21.1 Descrizione
Classe che visualizza la form necessaria ad aggiungere un nuovo utente alla piattaforma.

4.4.21.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: UserController per gestire le richieste di operazione relative agli utenti iscritti alla piattaforma.

4.4.22 app :: views :: admin :: user :: editUser

4.4.22.1 Descrizione
Classe che visualizza la form necessaria a modificare un utente presente nella piattaforma.

4.4.22.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: UserController per gestire le richieste di operazione relative agli utenti iscritti alla piattaforma.

4.4.23 app :: views :: admin :: home

4.4.23.1 Descrizione
Classe che visualizza la homepage per un utente di tipo Admin.

4.4.23.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: AdminController per visualizzare la pagina di home della proprio tipologia di account.

4.4.24 app :: views :: user :: home

4.4.24.1 Descrizione
Classe che visualizza la homepage per un utente di tipo User.

4.4.24.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: user :: UserController per visualizzare la pagina di home della proprio tipologia di account.

4.4.25 app :: views :: user :: thing :: thingDetails

4.4.25.1 Descrizione
Classe che visualizza una tabella contenente i dati di un particolare $thing_G$ e i grafici relativi allo stesso.

4.4.25.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: user :: ThingController per visualizzare le informazioni dei singoli oggetti.

4.4.26 app :: views :: user :: thingType :: thingTypeDetails

4.4.26.1 Descrizione
Classe che visualizza una tabella contenente il riepilogo dei dati degli oggetti di un particolare modello e i grafici generici relativi al modello stesso.

4.4.26.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: user :: ThingTypeController per visualizzare le informazioni dei singoli modelli.

4.4.27 app :: views :: shared :: adminUser :: things

4.4.27.1 Descrizione
Classe che visualizza la pagina con la lista dei things in uso dalla propria company.

4.4.27.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: ThingController e dalla classe app :: controllers :: user :: ThingController per la visualizzazione della pagina con la lista dei things.

4.4.28 app :: views :: shared :: adminUser :: thingTypes

4.4.28.1 Descrizione
Classe che visualizza la pagina con la lista delle tipologie di things in uso dalla propria company.

4.4.28.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: admin :: ThingTypeController e dalla classe app :: controllers :: user :: ThingTypeController per la visualizzazione della pagina con la lista dei modelli dei things in uso dalla propria company.

4.4.29 app :: views :: shared :: adminUser :: notifications

4.4.29.1 Descrizione
Classe che visualizza la lista con tutte le notifiche impostate per i thing o un loro tipo e la relativa interfaccia per lanciare le classiche operazioni CRUD.

4.4.29.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: adminUser :: NotificationController per gestire le richieste di operazione relative alle notitiche su thing e loro tipi.

4.4.30 app :: views :: shared :: adminUser :: addNotification

4.4.30.1 Descrizione
Classe che visualizza la form necessaria ad aggiungere una nuova notifica per un thing o una loro tipologia.

4.4.30.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: adminUser :: NotificationController per gestire le richieste di operazione relative alle notitiche su thing e loro tipi.

4.4.31 app :: views :: shared :: adminUser :: editNotification

4.4.31.1 Descrizione
Classe che visualizza la form necessaria a modificare una notifica impostata per un dato thing o una tipologia di esso.

4.4.31.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: adminUser :: NotificationController per gestire le richieste di operazione relative alle notitiche su thing e loro tipi.

4.4.32 app :: views :: shared :: authentication :: singIn

4.4.32.1 Descrizione
Classe che visualizza la pagina di $login_G$ con il $form_G$ necessario ad autenticarsi.

4.4.32.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: authentication :: AuthenticationController per la visualizzazione della pagina di $login_G$.

4.4.33 app :: views :: shared :: authentication :: singOut

4.4.33.1 Descrizione
Classe che visualizza gli esiti di un'operazione di logout.

4.4.33.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: authentication :: AuthenticationController per la visualizzazione della pagina necessaria dopo un'operazione di logout.

4.4.34 app :: views :: shared :: authentication :: passwordRecover

4.4.34.1 Descrizione
Classe che visualizza la pagina di recupero password e il form associato atto ad assolvere tale compito.

4.4.34.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: account :: PasswordRecoverController per la visualizzazione della pagina di recupero password.

4.4.35 app :: views :: shared :: authentication :: resetPassword

4.4.35.1 Descrizione
Classe che visualizza la pagina di reset password e il form associato atto ad assolvere tale compito.

4.4.35.2 Utilizzo
Viene utilizzata dalla classeapp :: controllers :: shared :: account :: PasswordResetController per la visualizzazione della pagina di reset password.

4.4.36 app :: views :: shared :: account :: editAccount

4.4.36.1 Descrizione
Classe che visualizza una pagina per la modifica del profilo contenente i $form_G$ per la modifica delle varie informazioni, sia anagrafiche che di autenticazione.

4.4.36.2 Utilizzo
Viene utilizzata dalla classe app :: controllers :: shared :: account :: AccountController per la visualizzazione della pagina di modifica del profilo.

4.4.37 app :: views :: shared :: layout :: header

4.4.37.1 Descrizione
Classe che permette la creazione dell'$header_G$ delle varie $view_G$.

4.4.37.2 Utilizzo
Viene utilizzata da tutte le $view_G$ per creare la parte di $header_G$.

4.4.38 app :: views :: shared :: layout :: footer

4.4.38.1 Descrizione
Classe che permette la creazione dell'$footer_G$ delle varie $view_G$.

4.4.38.2 Utilizzo
Viene utilizzata da tutte le $view_G$ per creare la parte di $footer_G$.

4.4.39 app :: views :: shared :: layout :: layout

4.4.39.1 Descrizione
Classe che permette la creazione del $layout_G$ delle varie $view_G$.

4.4.39.2 Utilizzo
Viene utilizzata da tutte le $view_G$ per creare la parte di $layout_G$.

4.4.40 app :: views :: shared :: layout :: nav

4.4.40.1 Descrizione
Classe che permette la creazione del menu di navigazione delle varie $view_G$. Ogni view riempirà il proprio menu a seconda dell'esigenza.

4.4.40.2 Utilizzo
Viene utilizzata da tutte le $view_G$ per creare il menu di navigazione.

4.4.41 app :: views :: shared :: errors :: 404

4.4.41.1 Descrizione
Classe che visualizza la pagina di errore 404_G.

4.4.41.2 Utilizzo
Tale classe entra in gioco ogni qual volta un utente inserisca una URL_G non corretta all'interno della barra degli indirizzi del browser per raggiungere una parte della piattaforma.

4.4.42 app :: views :: shared :: errors :: 502

4.4.42.1 Descrizione
Classe che visualizza la pagina di errore 502_G.

4.4.42.2 Utilizzo

Tale classe entra in gioco quando un qualsiasi utente, inserendo una URL_G all'interno della barra degli indirizzi del browser, cerchi di entrare in una pagina per la quale non ha i permessi di accesso.

5 Diagrammi di attività

In questa sezione vengono riportati i diagrammi di attività prodotti durante la progettazione architetturale. Essi hanno l'obiettivo di descrivere le interazioni degli utenti con il sistema.
Verrà inizialmente fornito uno schema generale, di alto livello, e successivamente saranno presenti i diagrammi di dettaglio di tali attività.

5.1 Attività Principali

Il seguente schema generale di alto livello indica le attività principali eseguibili dagli utenti all'interno della piattaforma.

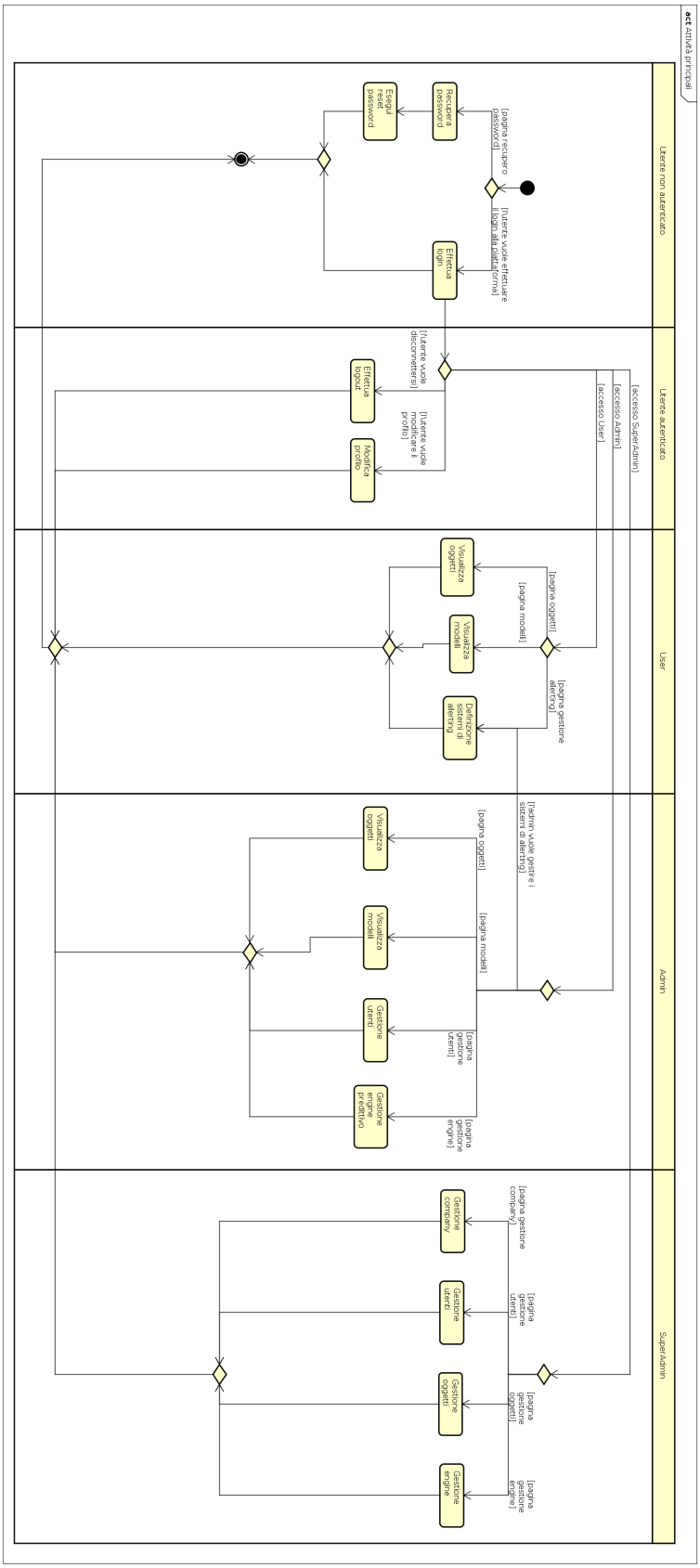

Figura 7: Diagramma di attività - Attività Principali.

La piattaforma $UMAP_G$ è costituita da una serie di pagine web che permettono di utilizzare il sistema. Un utente, accedendo inizialmente all'applicazione, ha la possibilità di effettuare il login o di recuperare la password dimenticata. Una volta entrato, a seconda della categoria alla quale appartiene, viene indirizzato alla pagina ad esso dedicata. Seguono le funzioni predisposte per ogni categoria di utente.

Uno User può:

- Visualizzare l'elenco degli oggetti della company a cui è associato;
- Visualizzare l'elenco dei modelli degli oggetti della company;
- Definire sistemi di allerting personalizzati.

Un Admin può:

- Visualizzare gli oggetti della propria $company_G$ e inviargli comandi da terminale e aggiornamenti;
- visualizzare i modelli della propria $company_G$ e inviargli aggiornamenti;
- gestire gli utenti della propria $company_G$;
- gestire l'algoritmo predittivo per la propria $company_G$;
- creare grafici per gli utenti della propria $company_G$.

Un SuperAdmin può:

- Gestire le $company_G$;
- gestire gli utenti della piattaforma;
- gestire gli oggetti della piattaforma (Thing e modelli);
- gestire l'algoritmo predittivo per tutte le $company_G$ della piattaforma.

Sono previste inoltre, per ogni categoria di utente, funzionalità per:

- Modificare il proprio profilo;
- Effettuare il logout.

5.2 Utente non Autenticato

5.2.1 Recupera password

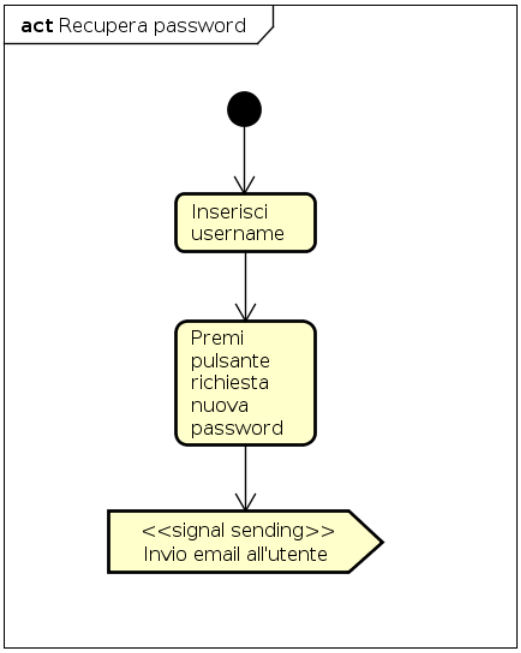

Figura 8: Diagramma di attività - Recupera password.

Nella pagina predisposta per il recupero della password, l'utente dovrà inserire il proprio username e premere il pulsante di invio. Il sistema procederà ad inviare una email contenente un link che rimanda alla pagina di inserimento di una nuova password.

5.2.2 Esegui reset password

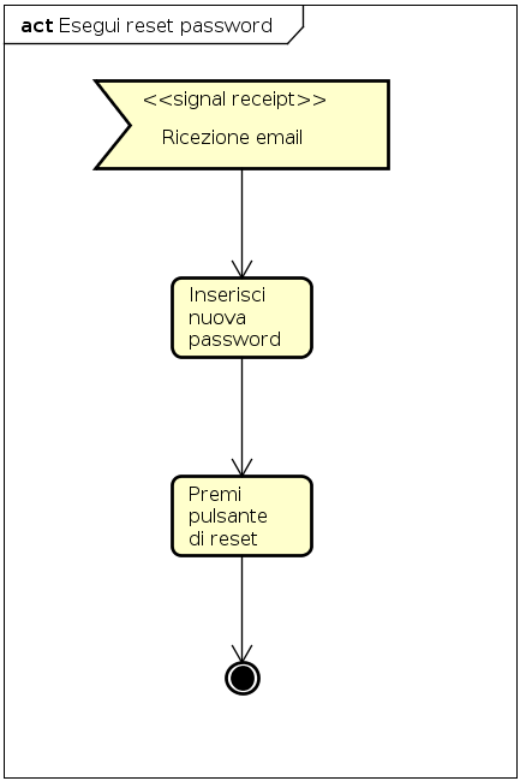

Figura 9: Diagramma di attività - Esegui reset password.

Una volta ricevuta l'email, basterà per l'utente cliccare sul link in essa contenuto per essere indirizzato alla pagina di inserimento di una nuova password. Tale pagina contiene un campo dati per in suo inserimento. Il pulsante di invio causerà il salvataggio della nuova password associata all'utente permettendogli, quindi, di accedere nuovamente al sistema.

5.2.3 Effettua login

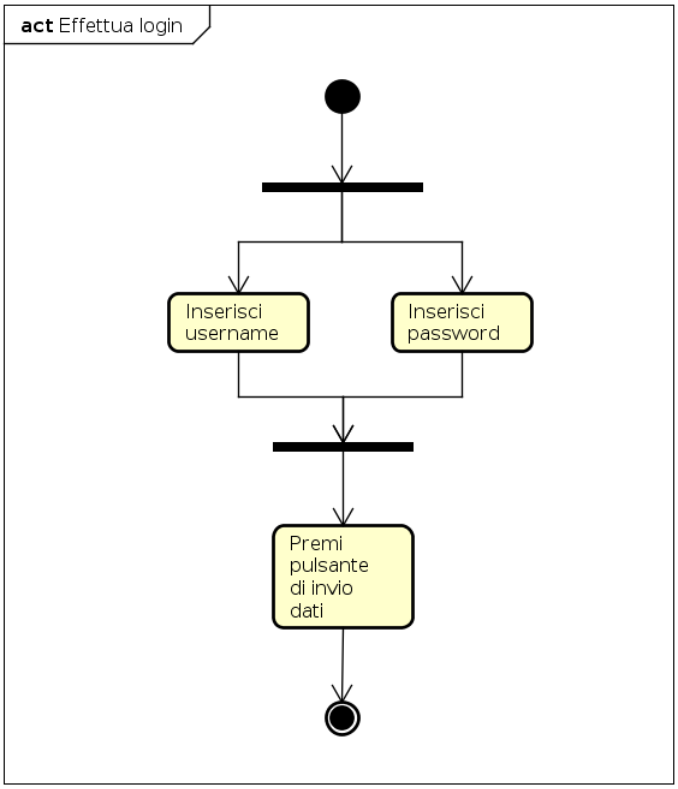

Figura 10: Diagramma di attività - Effettua login.

Per effettuare il login l'utente deve inserire i propri username e password e cliccare sul pulsante di invio. Il sistema, dopo aver verificato la correttezza delle credenziali, indirizzerà l'utente alla pagina ad esso dedicata, a seconda della categoria a cui appartiene.

5.3 Utente Autenticato

5.3.1 Modifica profilo

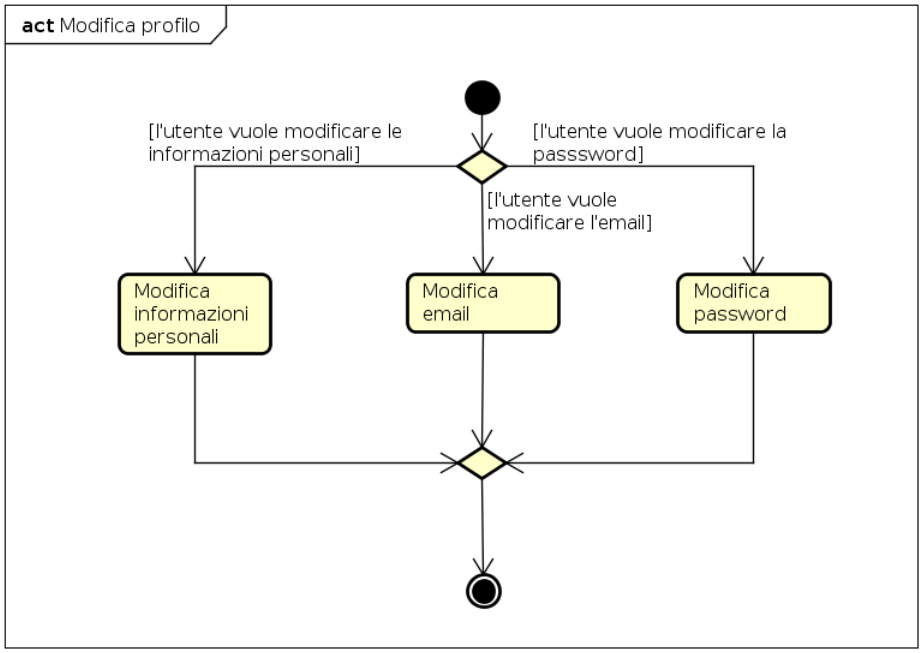

Figura 11: Diagramma di attività - Modifica profilo.

Nella pagina dedicata alla modifica del profilo, l'utente può scegliere se:

- Modificare le informazioni personali (nome e cognome);
- Modificare l'email;
- Modificare la password.

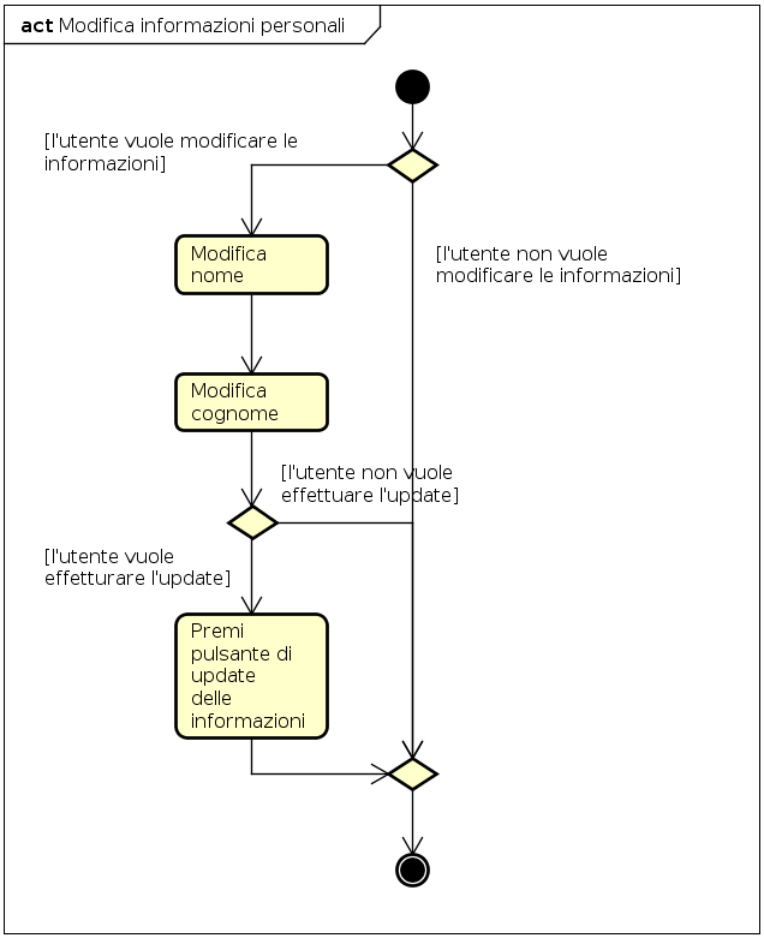

Figura 12: Diagramma di attività - Modifica informazioni personali.

5.3.1.1 Modifica informazioni personali Per quanto riguarda la modifica della informazioni personali, l'utente avrà a disposizione due campi dati in cui inserire i nuovi nome e cognome. Esso può anche annullare in ogni momento la procedura attraverso un apposito pulsante. Il pulsante di invio, invece, causa il salvataggio delle nuove informazioni da parte del sistema.

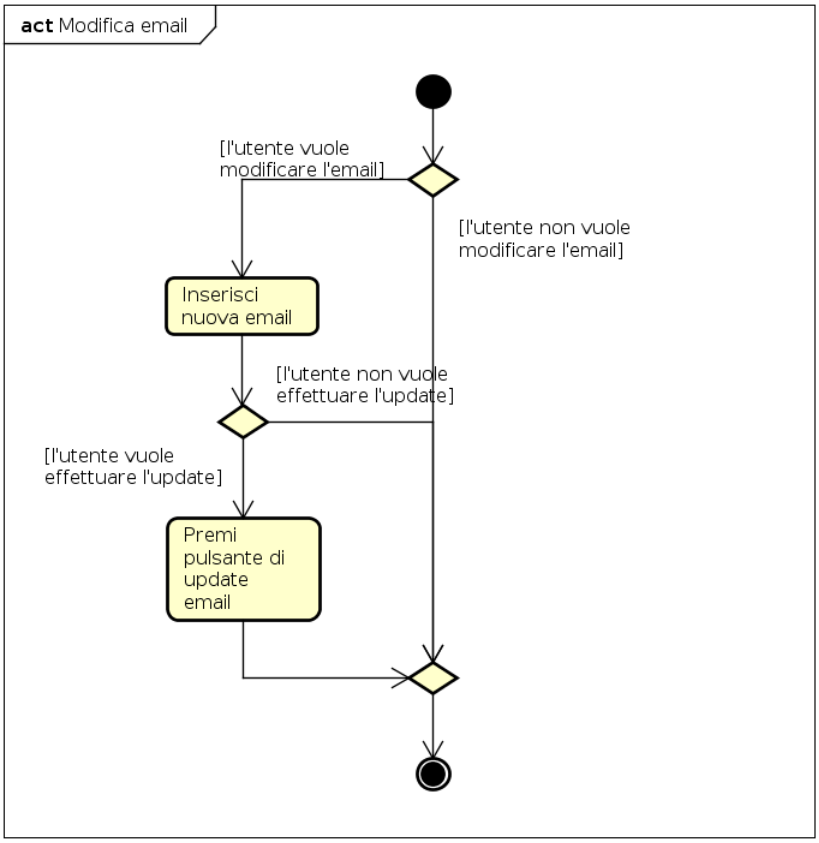

Figura 13: Diagramma di attività - Modifica email.

5.3.1.2 Modifica email Nella pagina dedicata alla modifica dell'email un campo dati permette all'utente di inserire il nuovo indirizzo. Il pulsante di invio causa il salvataggio delle modifiche nel sistema. Anche qui l'operazione può essere annullata in qualunque momento attraverso il pulsante apposito.

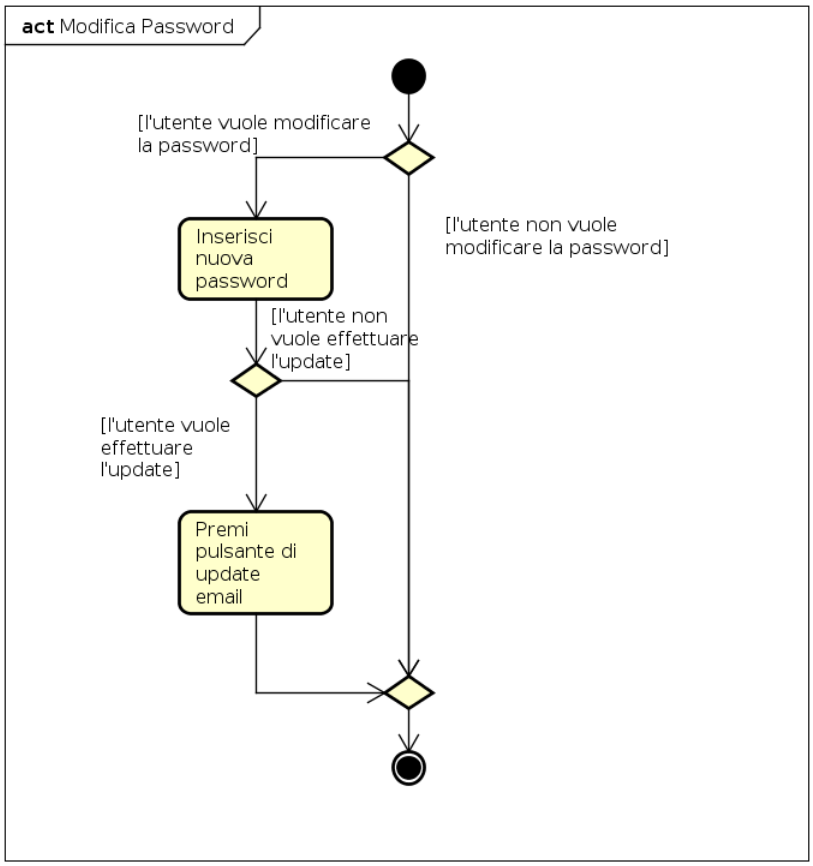

Figura 14: Diagramma di attività - Modifica password.

5.3.1.3 Modifica password Nella pagina dedicata alla modifica della password, un campo dati permette all'utente di inserire quella nuova. Il pulsante di invio causa il salvataggio delle modifiche nel sistema. Anche qui l'operazione può essere annullata in qualunque momento attraverso il pulsante apposito.

5.4 User

5.4.1 Visualizza Oggetti

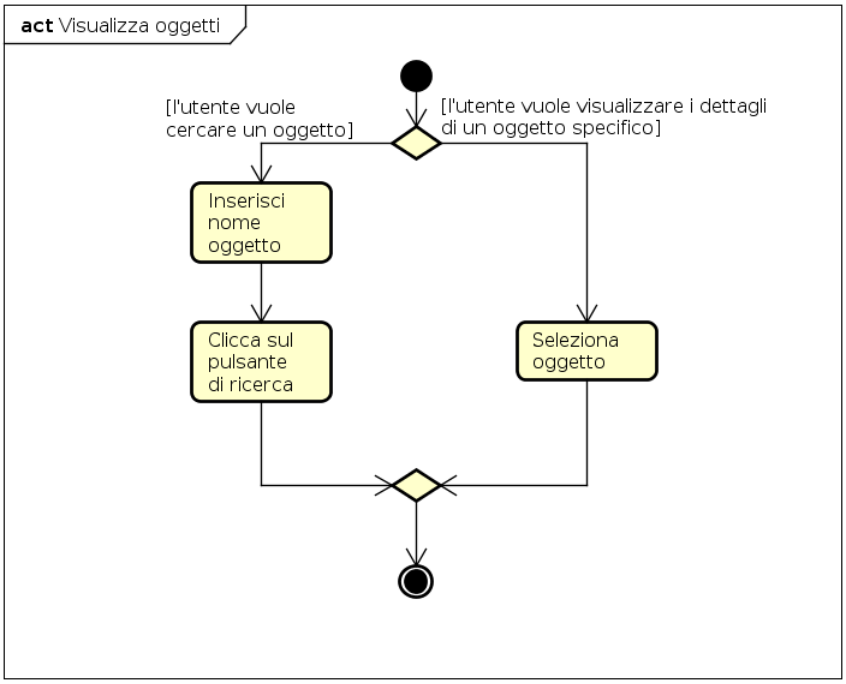

Figura 15: Diagramma di attività - Visualizza oggetti.

Una volta che l'utente entra nell'area dedicata alla visualizzazione degli oggetti, gli viene presentata una tabella contenente la lista degli oggetti appartenente alla company a cui è associato. Da qui può decidere se:

- Cercare un oggetto per nome: per eseguire questa attività l'utente deve inserire il nome dell'oggetto e successivamente, cliccando il pulsante di ricerca, la tabella si aggiorna con i risultati;
- Selezionare un oggetto per vedere i dettagli ad esso associati.

5.4.2 Visualizza modelli

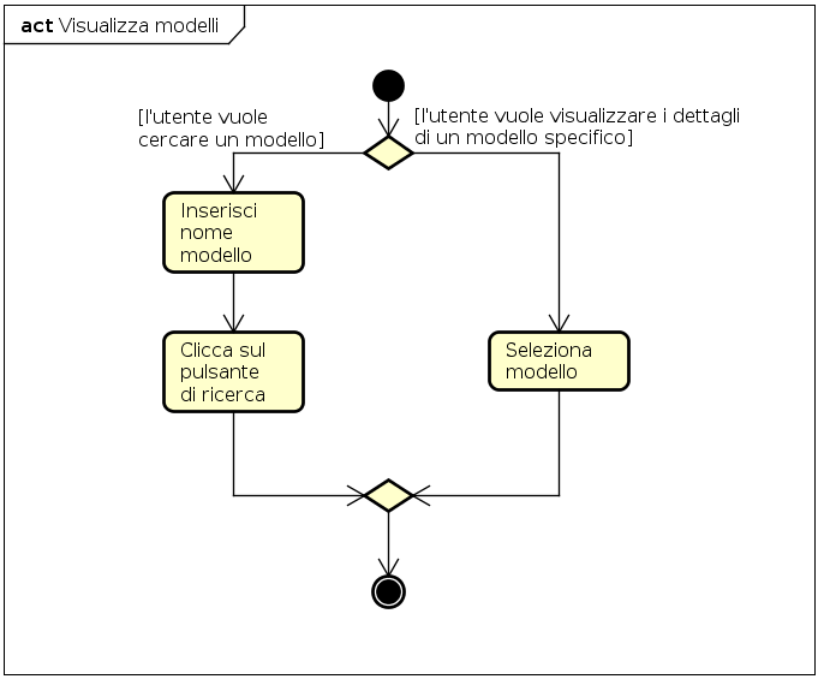

Figura 16: Diagramma di attività - Visualizza modelli.

Nella pagina dedicata alla visualizzazione dei modelli, l'utente visualizza una tabella contenente l'elenco dei modelli degli oggetti appartenenti alla company a cui è associato. Qui ha la possibilità di:

- Cercare un modello per nome: inserendo il nome di un modello e cliccano sul pulsante di ricerca, la tabella si aggiorna con i risultati trovati.
- Selezionare un modello per visualizzarne i dettagli.

5.4.3 Definizione sistemi di allerting

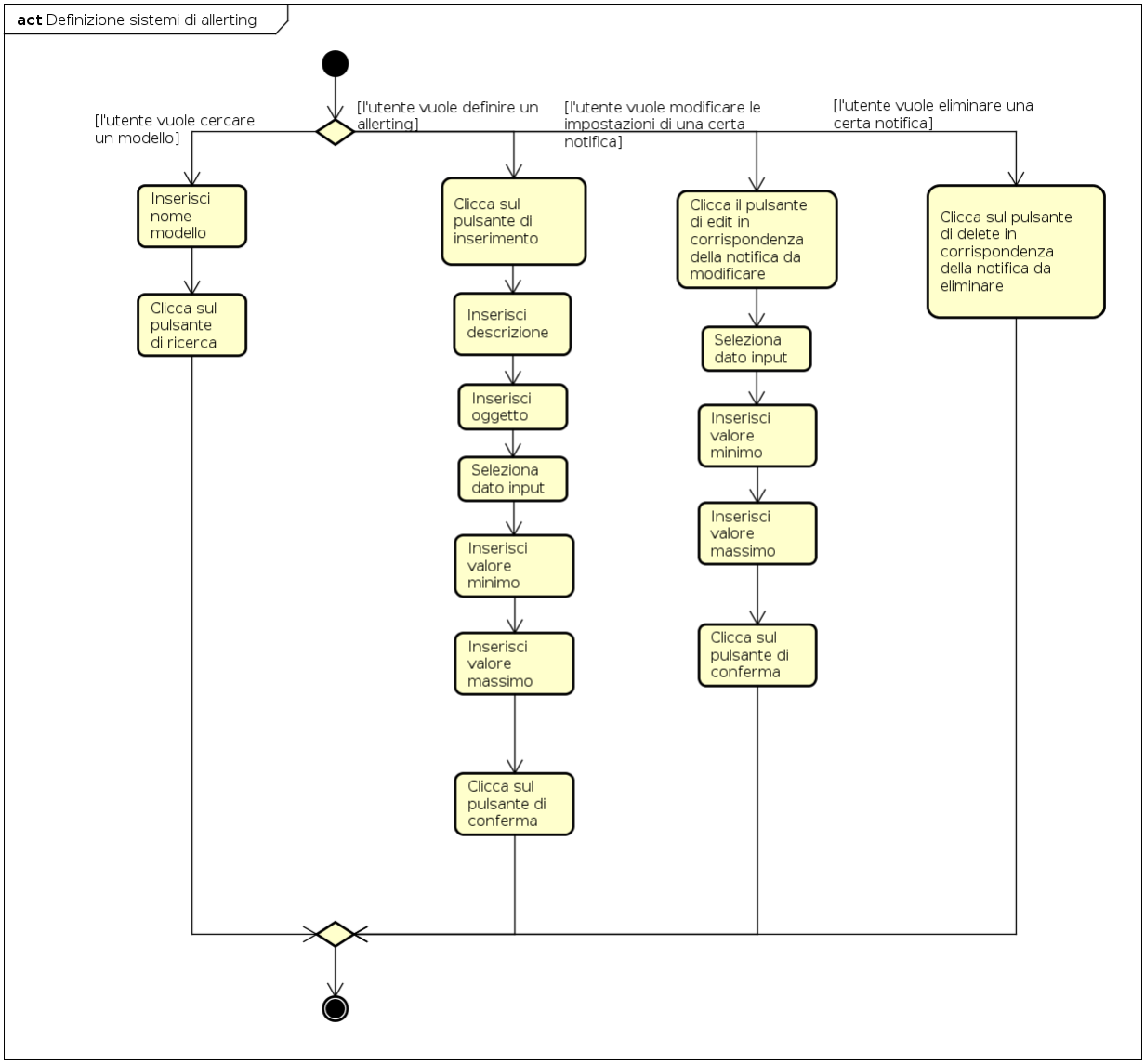

Figura 17: Diagramma di attività - Definizione sistemi di allerting.

Nella piattaforma è prevista la possibilità, per l'utente, di definire sistemi di allerting personalizzati. La pagina presenta, innanzitutto, una tabella contenente le notifiche già inserite e i relativi dettagli. Le operazioni possibili sono:

- Ricerca delle notifiche relative a oggetti di un certo modello;
- Inserimento di nuove notifiche;
- Modifica di notifiche già esistenti;
- Eliminazione di una notifica.

  Per cercare un modello è sufficiente, per l'utente, inserire il nome nella barra di ricerca e cliccare sul pulsante apposito. La tabella si aggiornerà con tutte le notifiche predisposte per oggetti di quel modello.

Cliccando sul pulsante di creazione di un nuovo allerting, il sistema presenta all'utente un pop-up in cui inserire tutte le informazioni necessarie alla creazione di una nuova notifica, associata ad uno specifico oggetto.
Cliccando sul pulsante di modifica in corrispondenza di una specifica notifica è possibile, nel pop-up che si apre, modificare le informazioni precedentemente inserite per definire la notifica.
Infine viene data la possibilità all'utente di eliminare un allerting precedentemente definito, cliccando sul pulsante di eliminazione in corrispondenza della notifica desiderata.
Le operazioni di creazione e modifica di una notifica possono essere annullate in qualunque momento attraverso un pulsante apposito nella finestra del pop-up.

5.5 Admin

5.5.1 Visualizza Oggetti

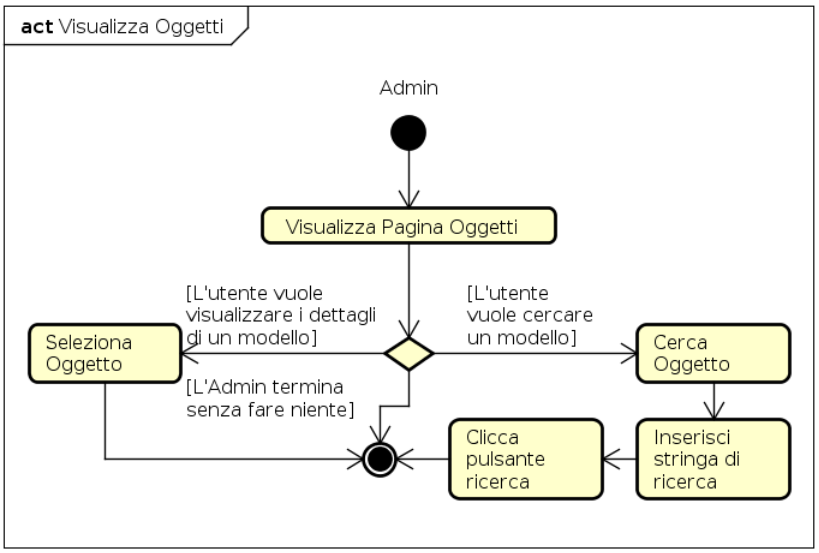

Figura 18: Diagramma di attività - Visualizza Oggetti.

Nella pagina dedicata alla visualizzazione degli oggetti, l'$admin_G$ visualizza una tabella contenente l'elenco degli oggetti appartenenti alla propria $company_G$. Qui ha la possibilità di:

- Cercare un oggetto per nome: inserendo una stringa e cliccando sul pulsante di ricerca, la tabella si aggiorna con i risultati trovati.
- Selezionare un oggetto per visualizzarne i dettagli.

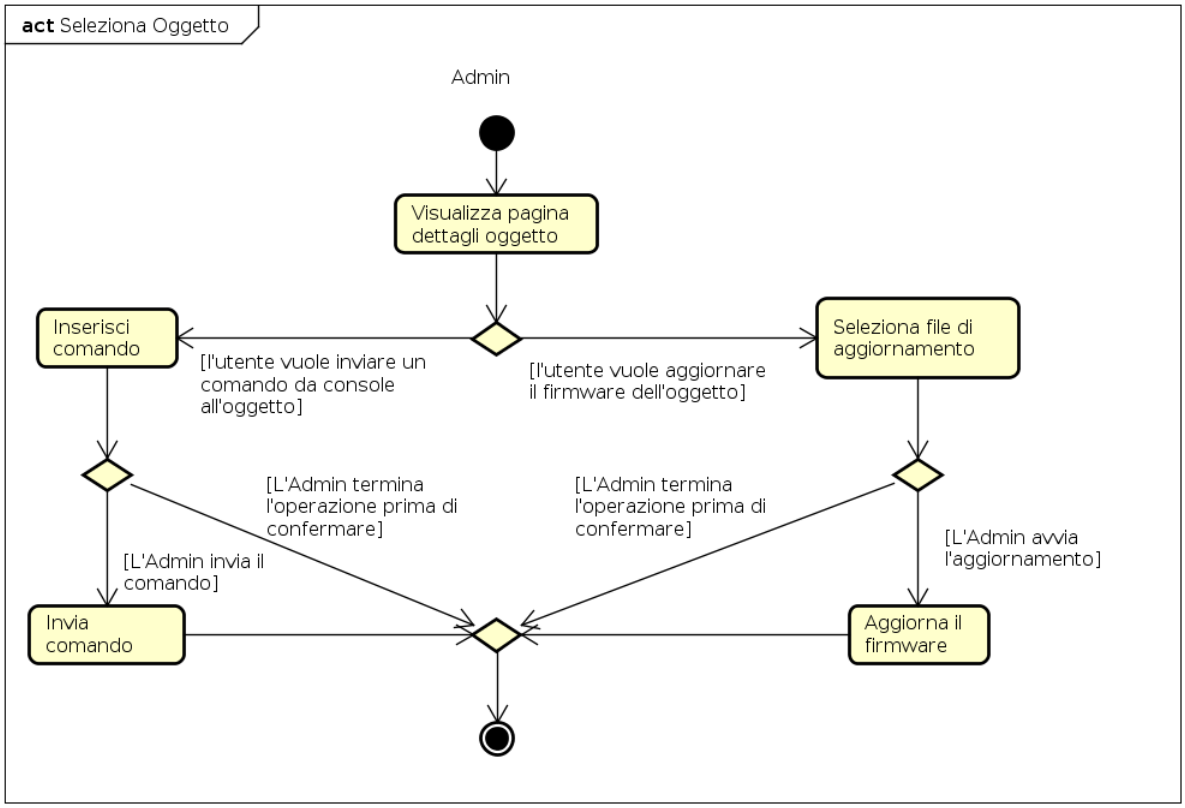

Figura 19: Diagramma di attività - Seleziona Oggetto.

5.5.1.1 Seleziona Oggetto Selezionando un oggetto l'$admin_G$ aprirà la pagina con i dettagli relativi a quel particolare oggetto. Oltre a visualizzare una tabella e dei grafici relativi ad esso, sarà possibile:

- Inviare dei comandi da riga di comando all'oggetto
- Caricare un file per l'aggiornamento e di conseguenza aggiornare il firmware dell'oggetto.

5.5.2 Visualizza Modelli

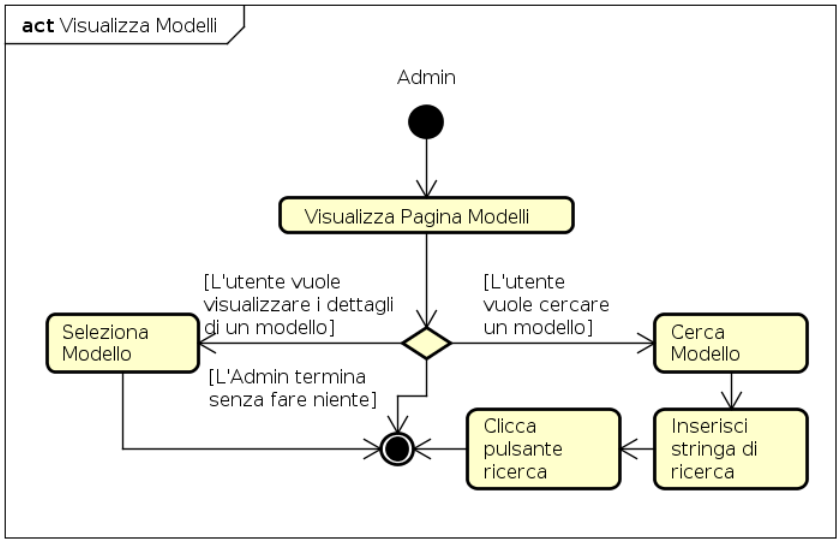

Figura 20: Diagramma di attività - Visualizza Modelli.

Nella pagina dedicata alla visualizzazione dei modelli, l'$admin_G$ visualizza una tabella contenente l'elenco dei modelli di oggetti appartenenti alla propria $company_G$. Qui ha la possibilità di:

- Cercare un modello per nome: inserendo una stringa e cliccando sul pulsante di ricerca, la tabella si aggiorna con i risultati trovati.
- Selezionare un modello per visualizzarne i dettagli.

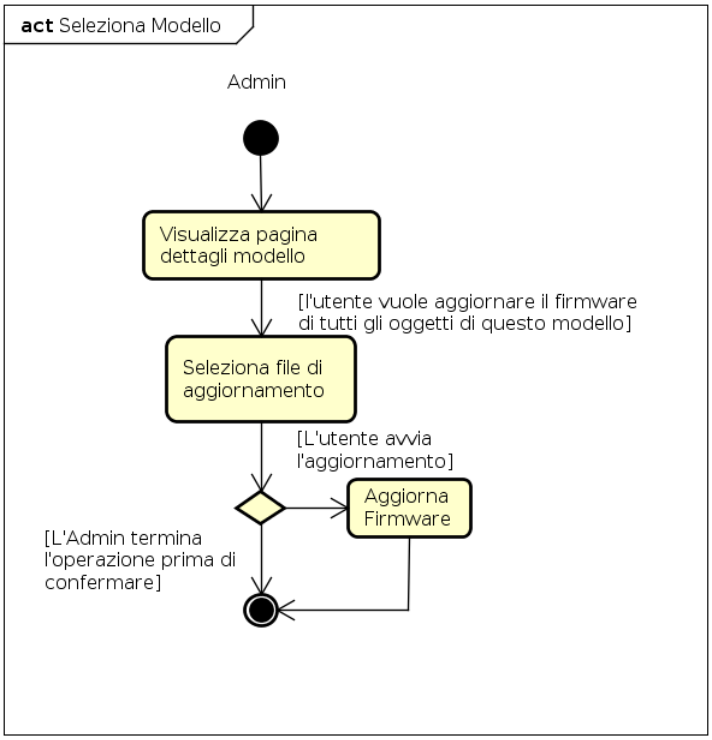

Figura 21: Diagramma di attività - Seleziona Modello.

5.5.2.1 Seleziona Modello Selezionando un modello l'$admin_G$ aprirà la pagina con i dettagli relativi a quel particolare modello. Oltre a visualizzare una tabella e dei grafici relativi ad esso, sarà possibile caricare un file per l'aggiornamento e di conseguenza aggiornare il firmware di tutti gli oggetti di quel particolare modello facenti parte della propria $company_G$.

5.5.3 Gestione Utenti

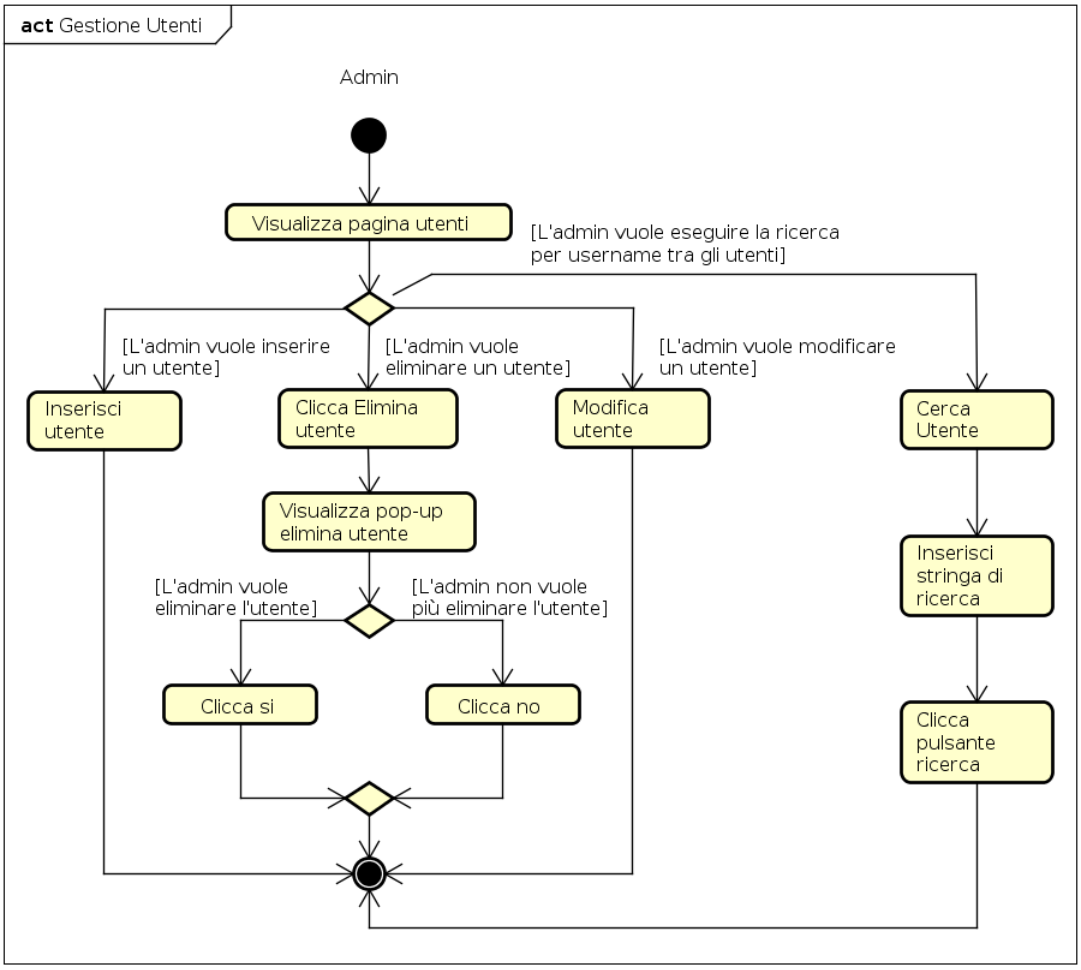

Figura 22: Diagramma di attività - Gestione Utenti.

Nella pagina dedicata alla visualizzazione degli utenti, l'$admin_G$ visualizza una tabella contenente l'elenco degli utenti appartenenti alla propria $company_G$. Qui ha la possibilità di:

- Inserire un nuovo utente;
- Eliminare un utente esistente premendo l'apposito pulsante;
- Modificare un utente già esistente;
- Cercare un utente per username: inserendo una stringa e cliccando sul pulsante di ricerca, la tabella si aggiorna con i risultati trovati.

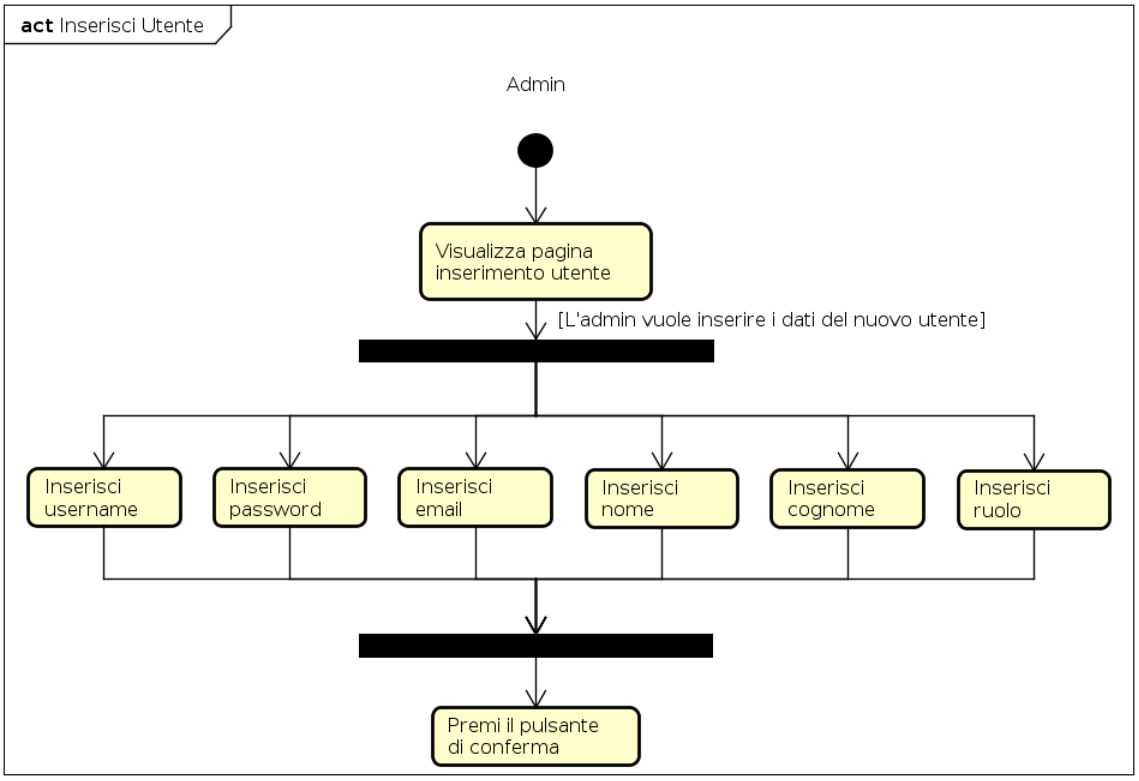

Figura 23: Diagramma di attività - Inserisci Utente.

5.5.3.1 Inserisci Utente Visualizzando la pagina per l'inserimento dell'utente l'$admin_G$ potrà inserire tutti i campi obbligatori per aggiungere un nuovo utente alla propria $company_G$ e inserirlo. I campi sono:

- $Username_G$;
- Password;
- E-mail;
- Nome;
- Cognome;
- Ruolo.

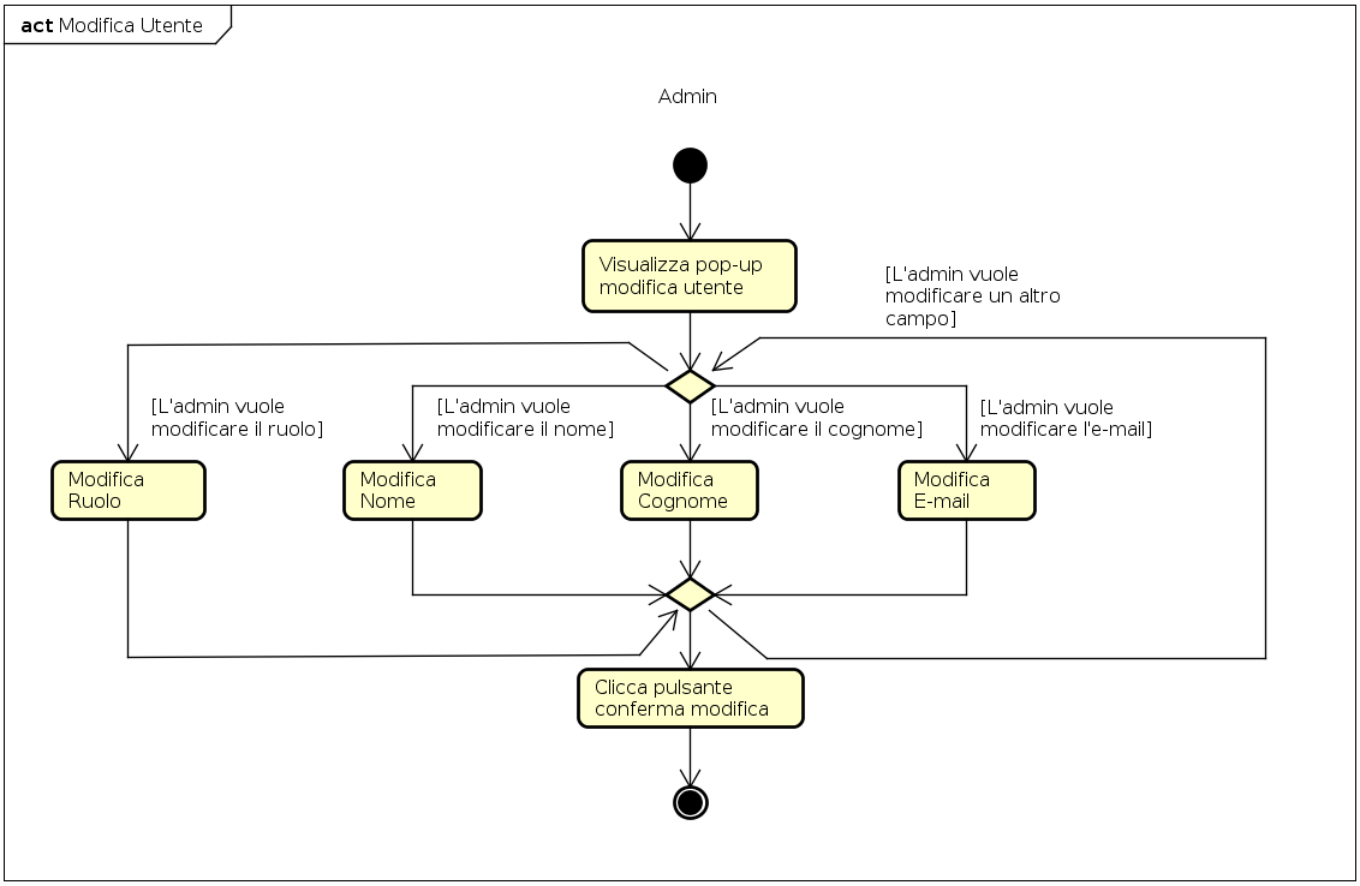

Figura 24: Diagramma di attività - Modifica Utente.

5.5.3.2 Modifica Utente Visualizzando la pagina per modificare un utente l'$admin_G$ potrà cambiare tutti i campi tranne lo $username_G$, che identifica univocamente l'utente, e la password. I campi sono:

- E-mail;
- Nome;
- Cognome;
- Ruolo.

5.5.4 Gestione Engine

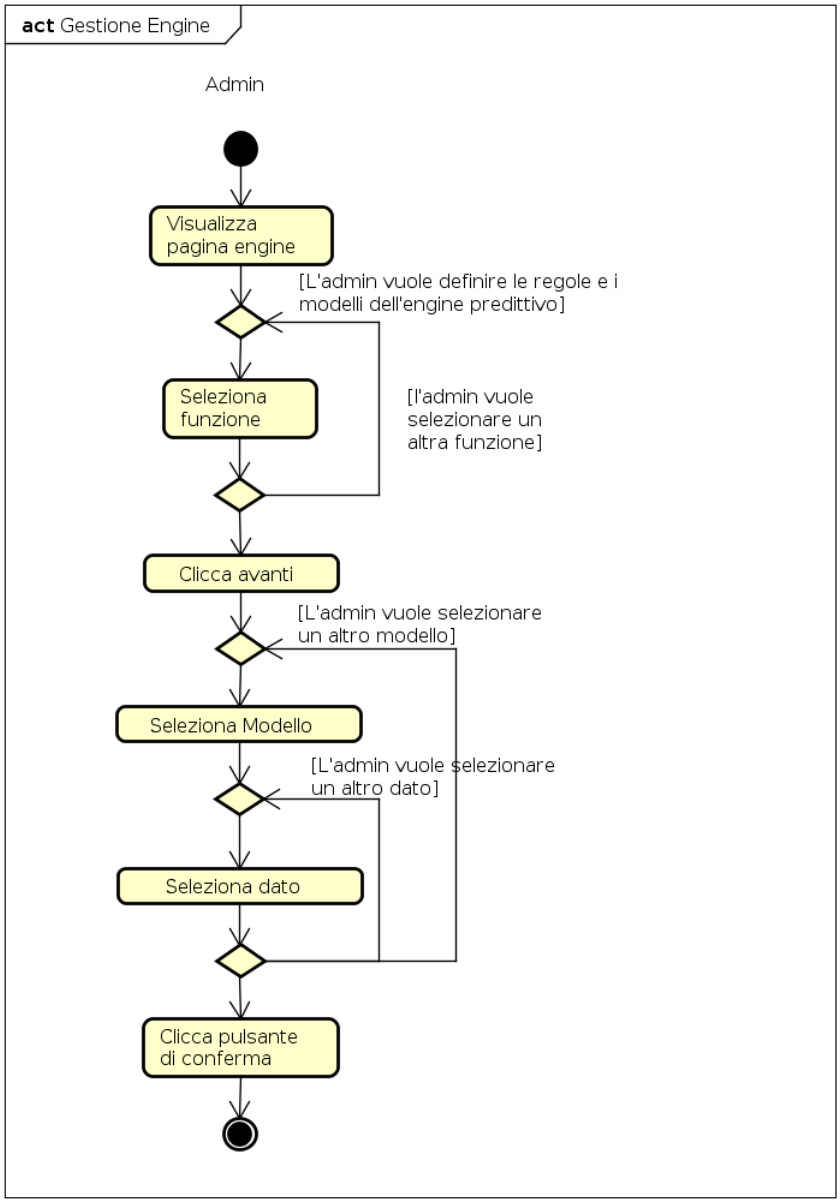

Figura 25: Diagramma di attività - Gestione Engine.

Nella pagina dedicata alla gestione dell'engine è presente un procedimento a due fasi che permetterà all'$admin_G$ di decidere prima le funzioni e poi i dati che potranno essere utilizzati dall'algoritmo e dagli $admin_G$ per la creazione di grafici e tabelle relative agli oggetti e ai modelli relativi ad essi.

5.5.5 Creazione Grafici

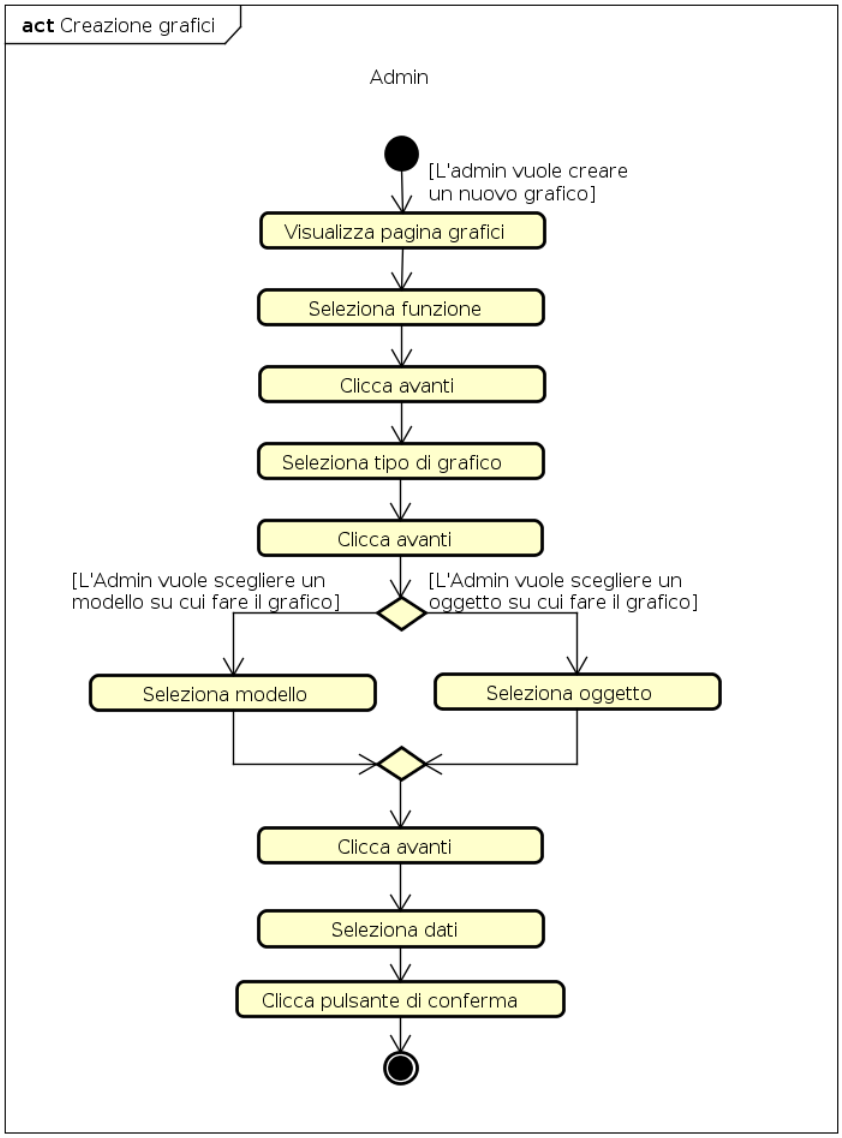

Figura 26: Diagramma di attività - Creazione Grafici.

Nella pagina dedicata alla creazione di grafici sarà possibile, attraverso un procedimento a quattro fasi, decidere in ordine la funzione da utilizzare, il tipo di grafico da creare, l'oggetto o il modello e quali dei suoi campi dati analizzare.

La pagina presenterà quindi 2 sezioni distinte:

- Una prima sezione più in alto che cambierà dinamicamente a seconda della fase in cui si trova il procedimento. Conterrà gli elementi da poter scegliere, tramite un meccanismo di *drag & drop*$_G$;
- La seconda sezione fa da contenitore per gli elementi già selezionati, in ogni fase del procedimento sarà possibile vedere tutte le scelte prese durante le fasi precedenti.

5.6 Super Admin

5.6.1 Gestione Company

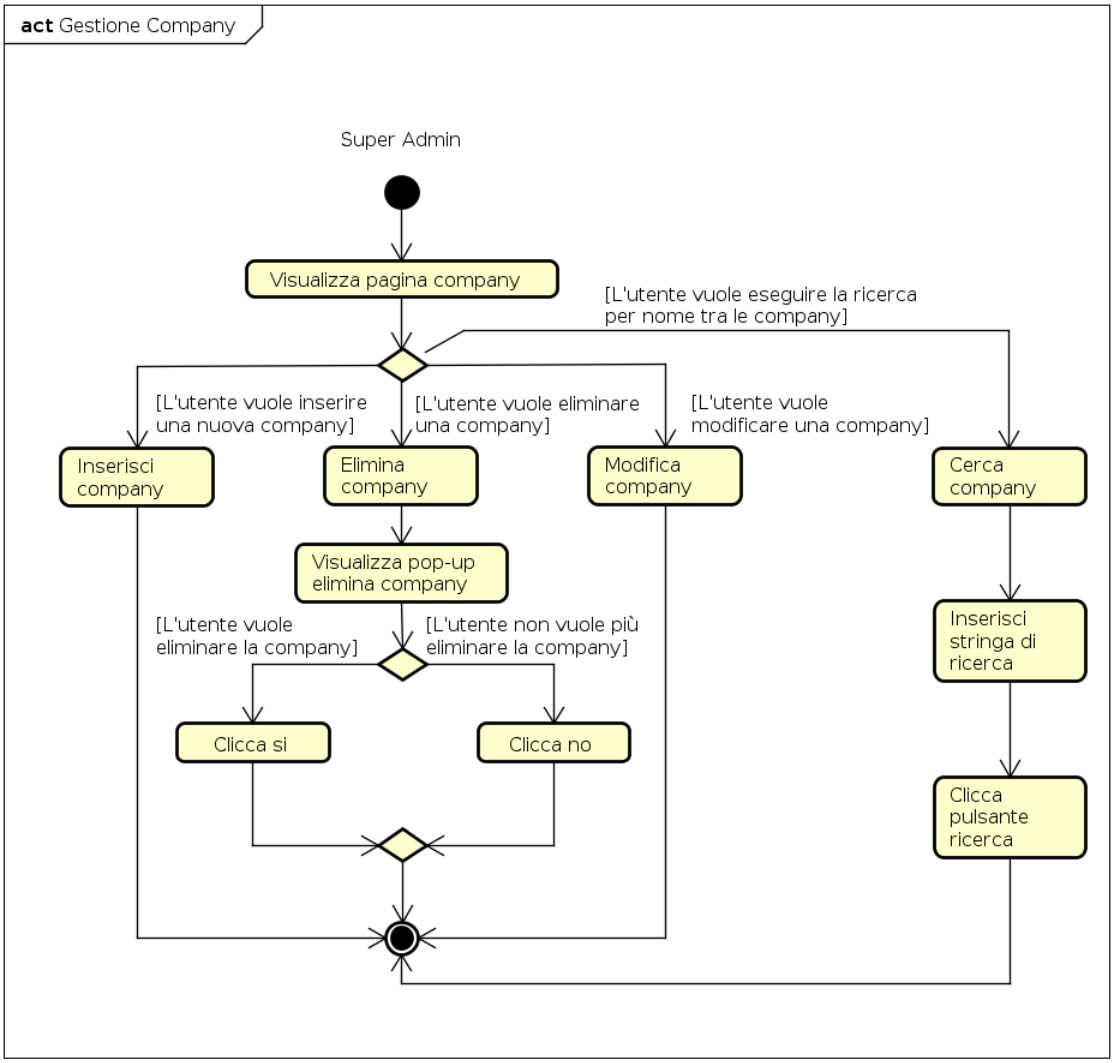

Figura 27: Diagramma di attività - Gestione Company.

Nella pagina dedicata alla gestione delle $company_G$ sarà presente una tabella con la lista di tutte le $company_G$ presenti nella piattaforma. Sarà inoltre possibile eseguire le seguenti operazioni:

- Inserire una nuova $company_G$;
- Eliminare una $company_G$ già esistente tramite l'apposito pulsante;
- Modificare una $company_G$ già esistente tramite l'apposito pulsante;
- Cercare una $company_G$ per nome: inserendo una stringa e premendo il pulsante di ricerca la tabella si aggiornerà con i risultati trovati.

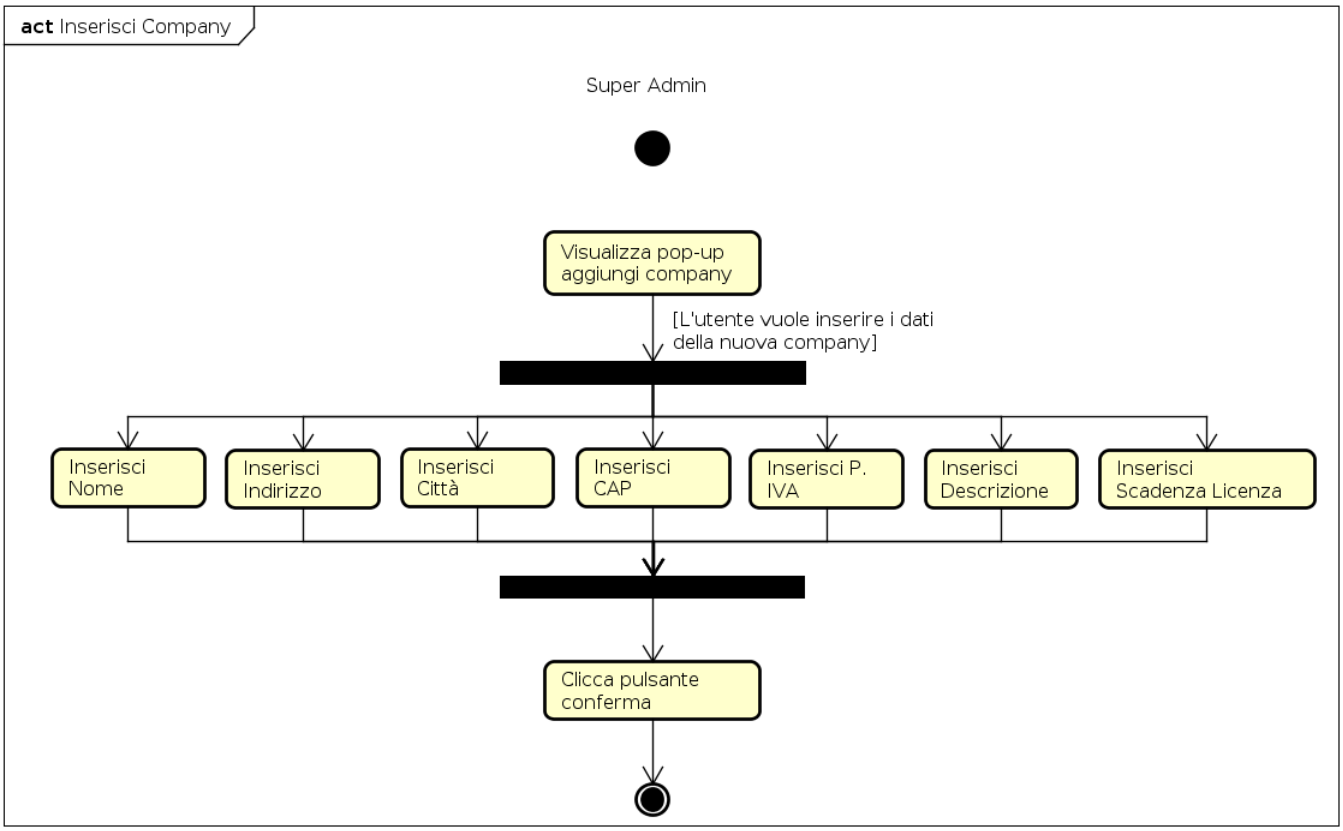

Figura 28: Diagramma di attività - Inserisci Company.

5.6.1.1 Inserisci Company Cliccando sul pulsante per l'inserimento di una $company_G$ verrà visualizzato il pop-up relativo che permetterà di inserire i campi obbligatori per l'aggiunta di una nuova $company_G$. I campi sono:

- Nome;
- Indirizzo;
- Città;
- CAP;
- P.IVA;
- Descrizione;
- Scadenza Licenza.

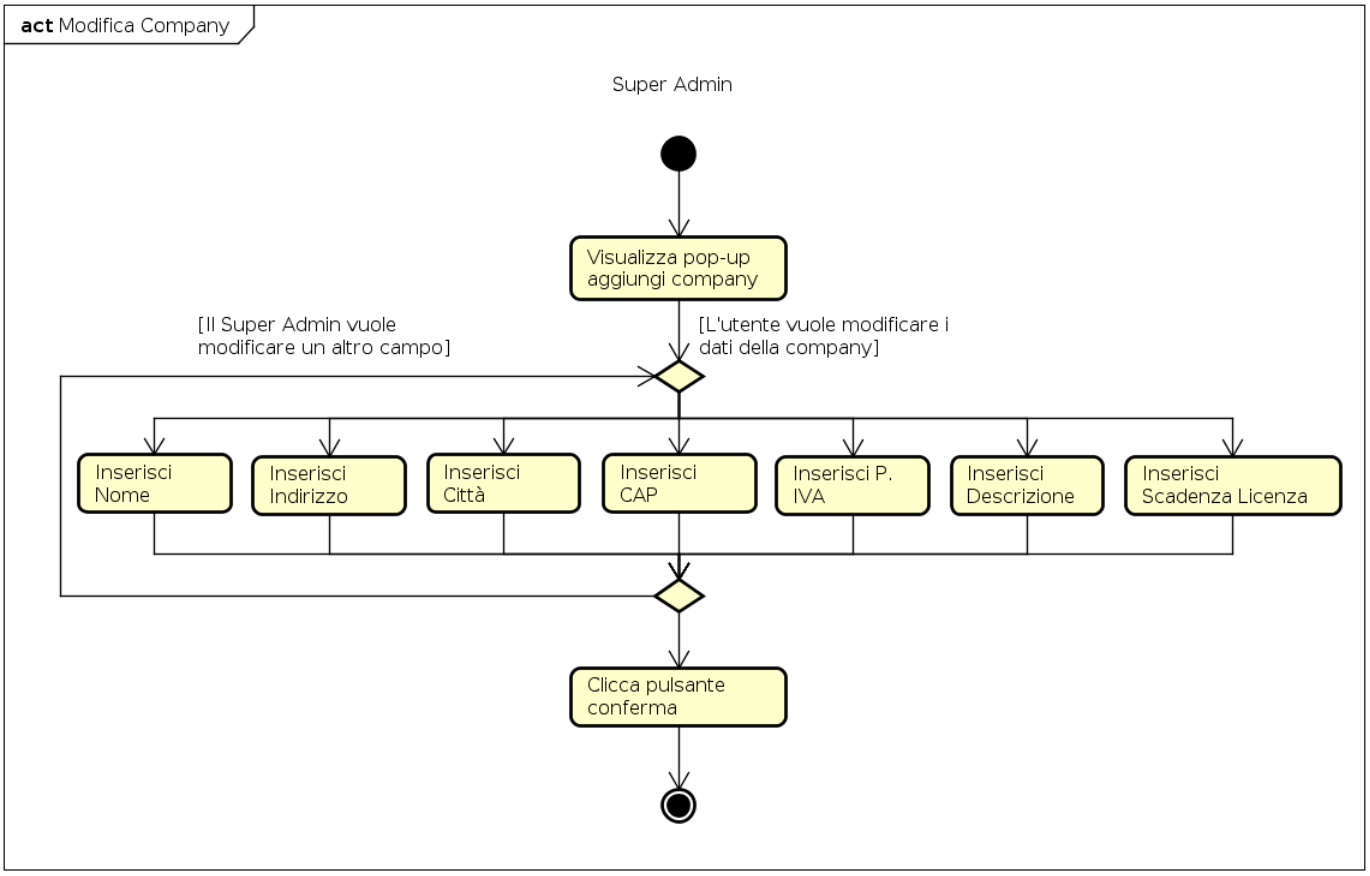

Figura 29: Diagramma di attività - Modifica Company.

5.6.1.2 Modifica Company Cliccando sul pulsante per la modifica della $company_G$ verrà visualizzato il pop-up relativo che permetterà di modificare i campi relativi alla $company_G$ selezionata. I campi sono:

- Nome;
- Indirizzo;
- Città;
- CAP;
- P.IVA;
- Descrizione;
- Scadenza Licenza.

5.6.2 Gestione Utenti

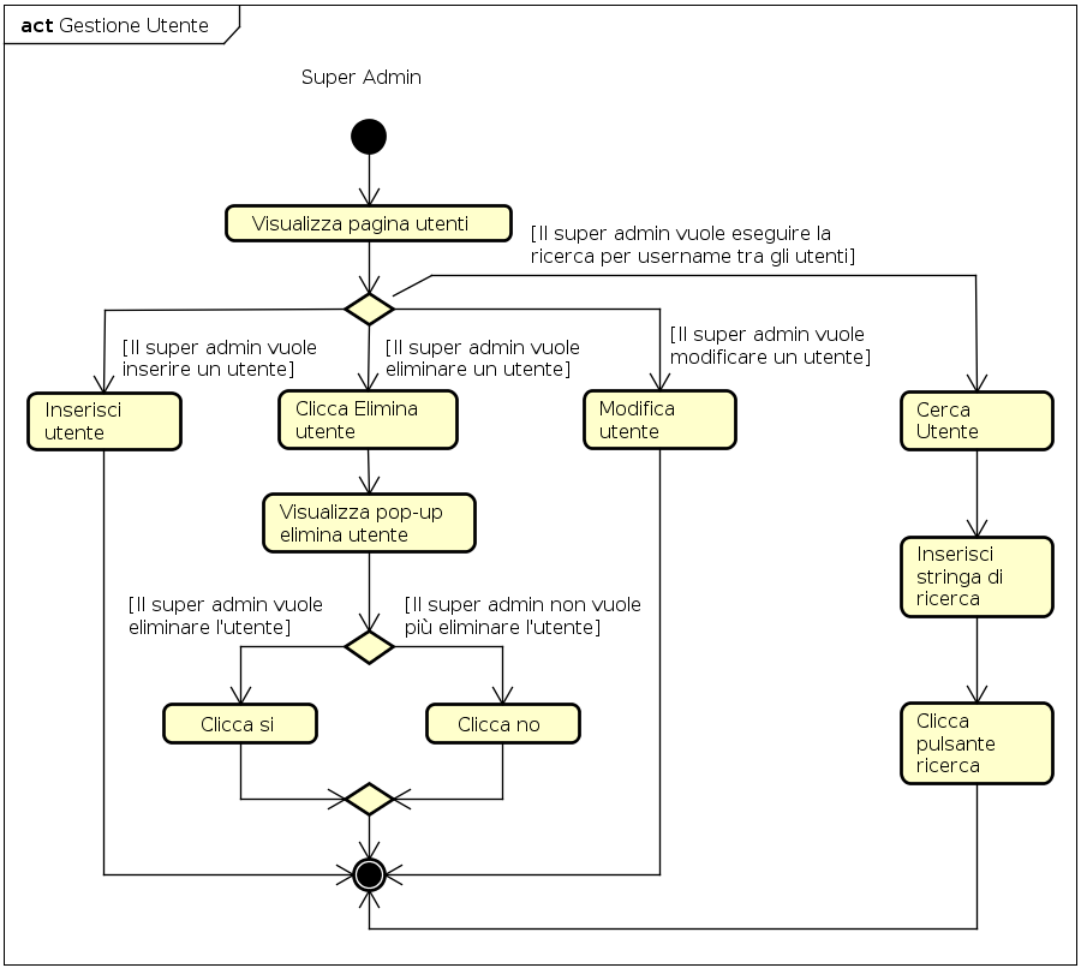

Figura 30: Diagramma di attività - Gestione Utenti.

Nella pagina dedicata alla visualizzazione degli utenti, il *super admin*$_G$ visualizza una tabella contenente l'elenco degli utenti presenti nella piattaforma. Qui ha la possibilità di:

- Inserire un nuovo utente;
- Eliminare un utente esistente premendo l'apposito pulsante;
- Modificare un utente già esistente;
- Cercare un utente per username: inserendo una stringa e cliccando sul pulsante di ricerca, la tabella si aggiorna con i risultati trovati.

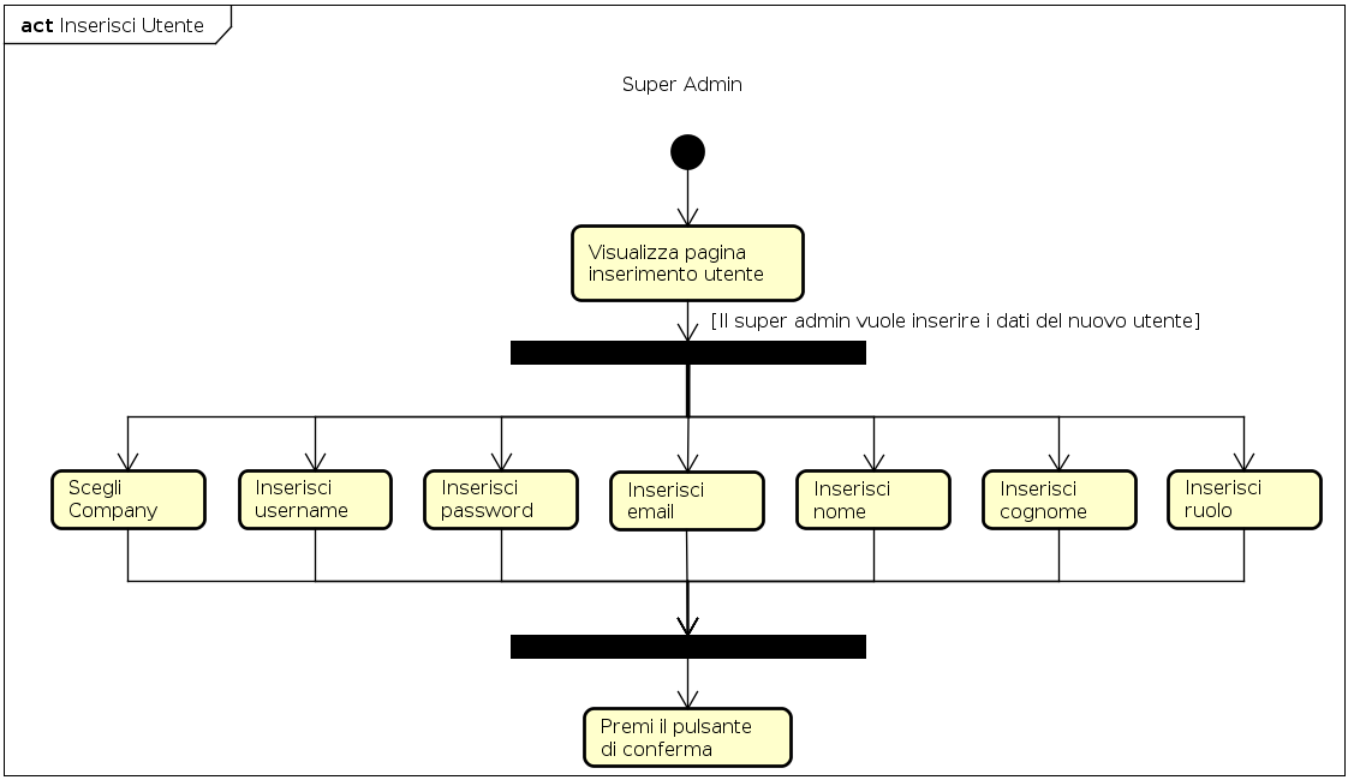

Figura 31: Diagramma di attività - Inserisci Utente.

5.6.2.1 Inserisci Utente Visualizzando la pagina per l'inserimento dell'utente il *super admin*$_G$ potrà inserire tutti i campi obbligatori per aggiungere un nuovo utente alla piattaforma e inserirlo. I campi sono:

- *Company*$_G$;
- *Username*$_G$;
- Password;
- E-mail;
- Nome;
- Cognome;
- Ruolo.

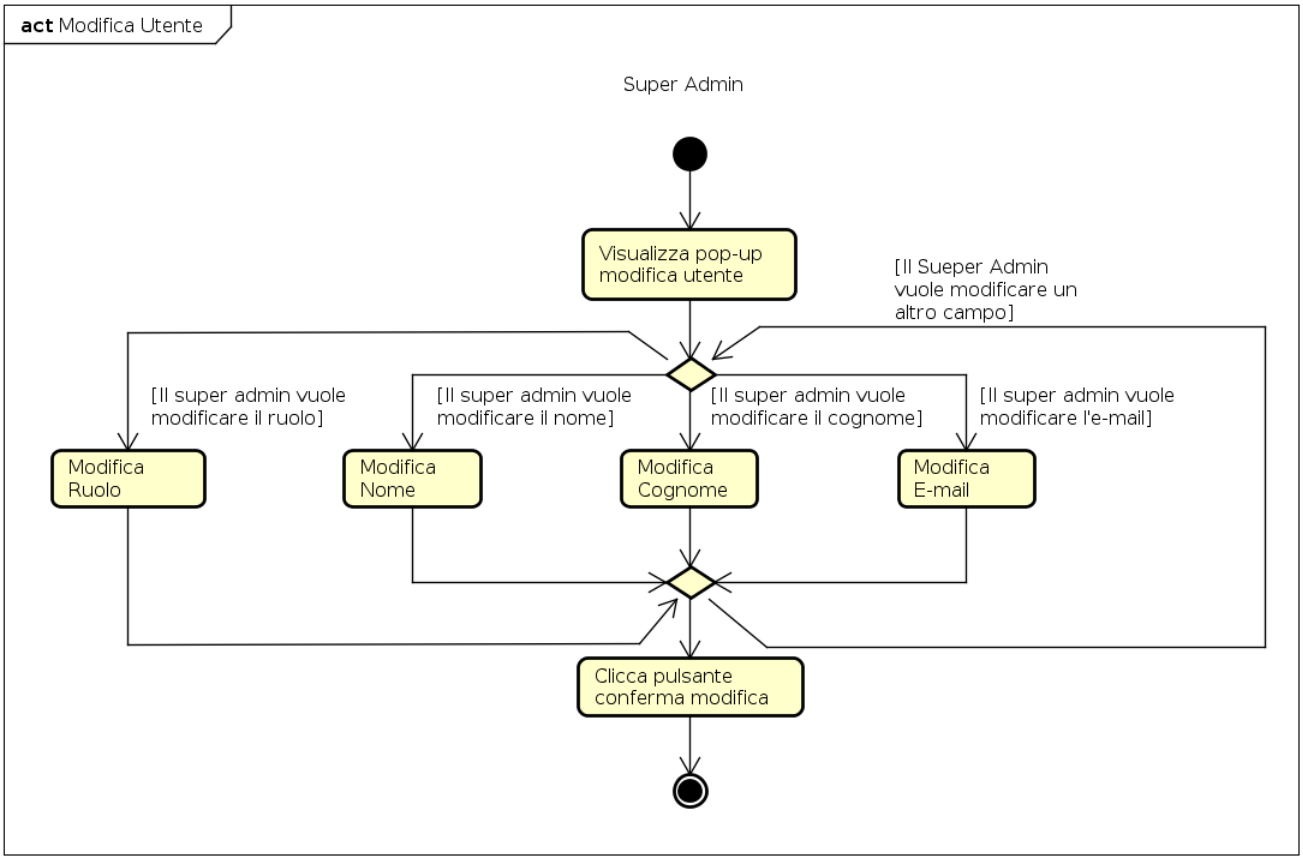

Figura 32: Diagramma di attività - Modifica Utente.

5.6.2.2 Modifica Utente Premendo il pulsante per la modifica apparirà l'apposito pop-up per modificare un utente. Il *super admin*$_G$ potrà cambiare tutti i campi tranne lo *username*$_G$, che identifica univocamente l'utente, e la password. I campi sono:

- E-mail;
- Nome;
- Cognome;
- Ruolo.

5.6.3 Gestione Oggetti

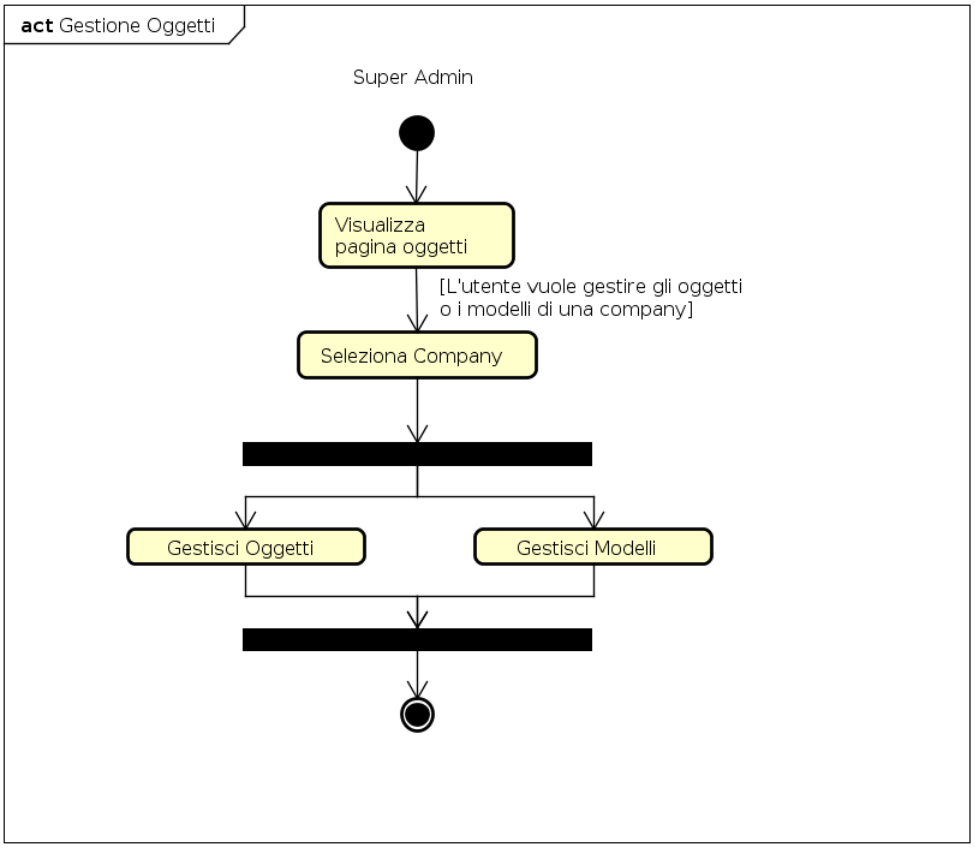

Figura 33: Diagramma di attività - Gestione Oggetti.

Nella pagina dedicata alla gestione degli oggetti saranno presenti due tabelle contenenti rispettivamente la lista di tutti gli oggetti presenti nella piattaforma e la lista di tutti i modelli di oggetti presenti nella piattaforma. Sarà possibile, attraverso l'apposita tabella, gestire gli oggetti o i modelli.

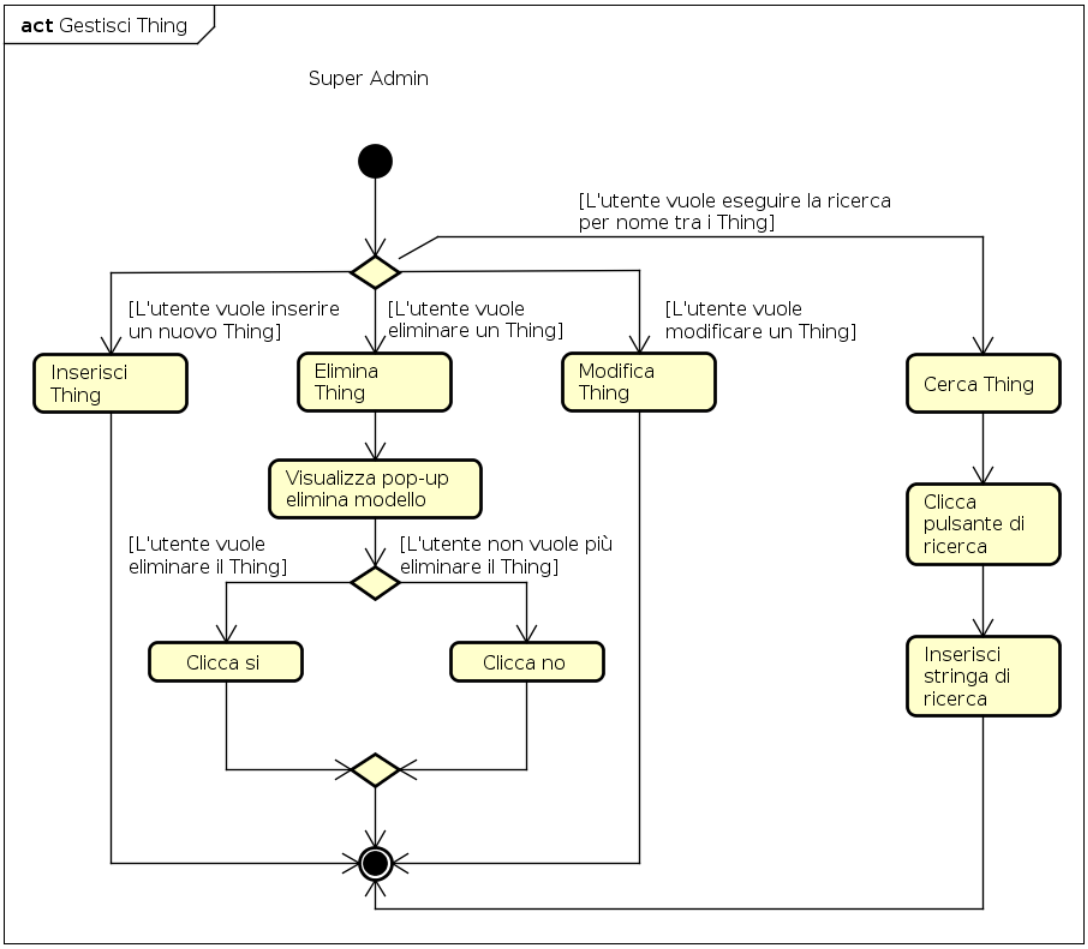

Figura 34: Diagramma di attività - Gestisci Thing.

5.6.3.1 Gestisci Thing Dalla tabella degli oggetti sarà possibile eseguire le seguenti operazioni:

- Inserire un nuovo $Thing_G$ all'interno della piattaforma;
- Eliminare un $Thing_G$ già presente nella piattaforma;
- Modificare un $Thing_G$ già presente nella piattaforma;
- Cercare un $Thing_G$ per nome: inserendo una stringa e cliccando sul pulsante di ricerca, la tabella si aggiorna con i risultati trovati.

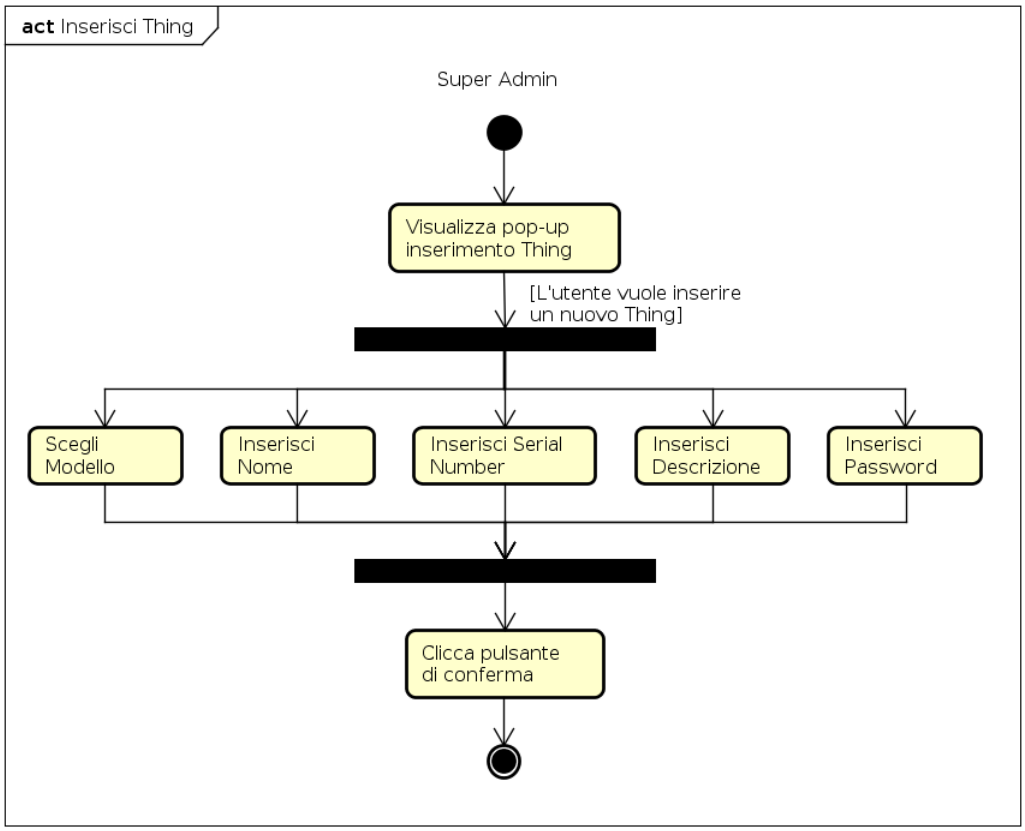

Figura 35: Diagramma di attività - Inserisci Thing.

5.6.3.1.1 Inserisci Thing Premendo l'apposito pulsante si aprirà un pop-up per l'inserimento di un nuovo $Thing_G$ dove sarà possibile inserire i campi dati obbligatori per la sua creazione. I campi sono i seguenti:

- Modello;
- Nome;
- Password;
- Serial Number;
- Descrizione;

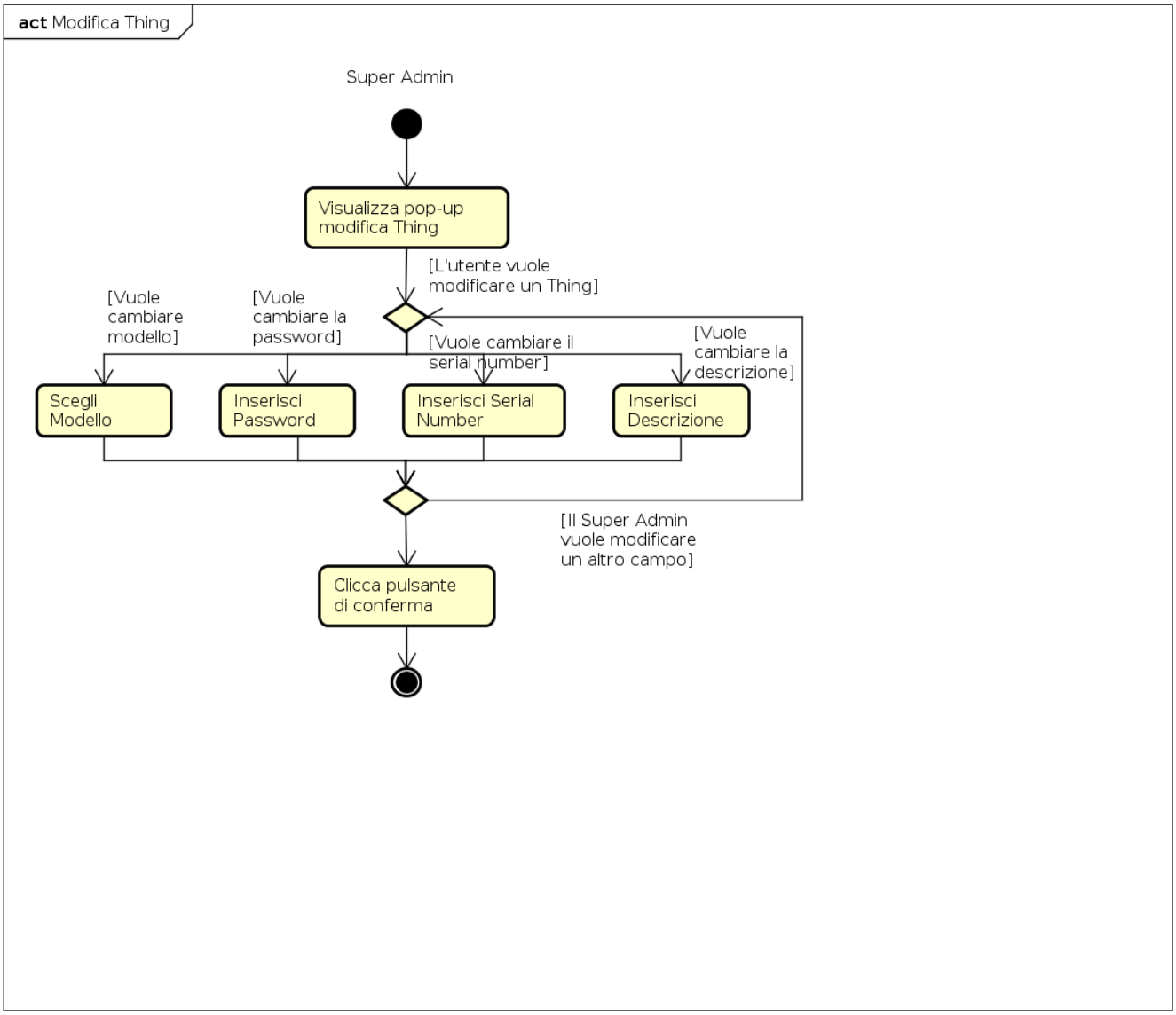

Figura 36: Diagramma di attività - Modifica Thing.

5.6.3.1.2 Modifica Thing Premendo l'apposito pulsante si aprirà un pop-up per la modifica del $Thing_G$ selezionato dove sarà possibile modificare tutti i campi dati tranne il nome, che lo identifica univocamente. I campi sono i seguenti:

- Modello;
- Password;
- Serial Number;
- Descrizione;

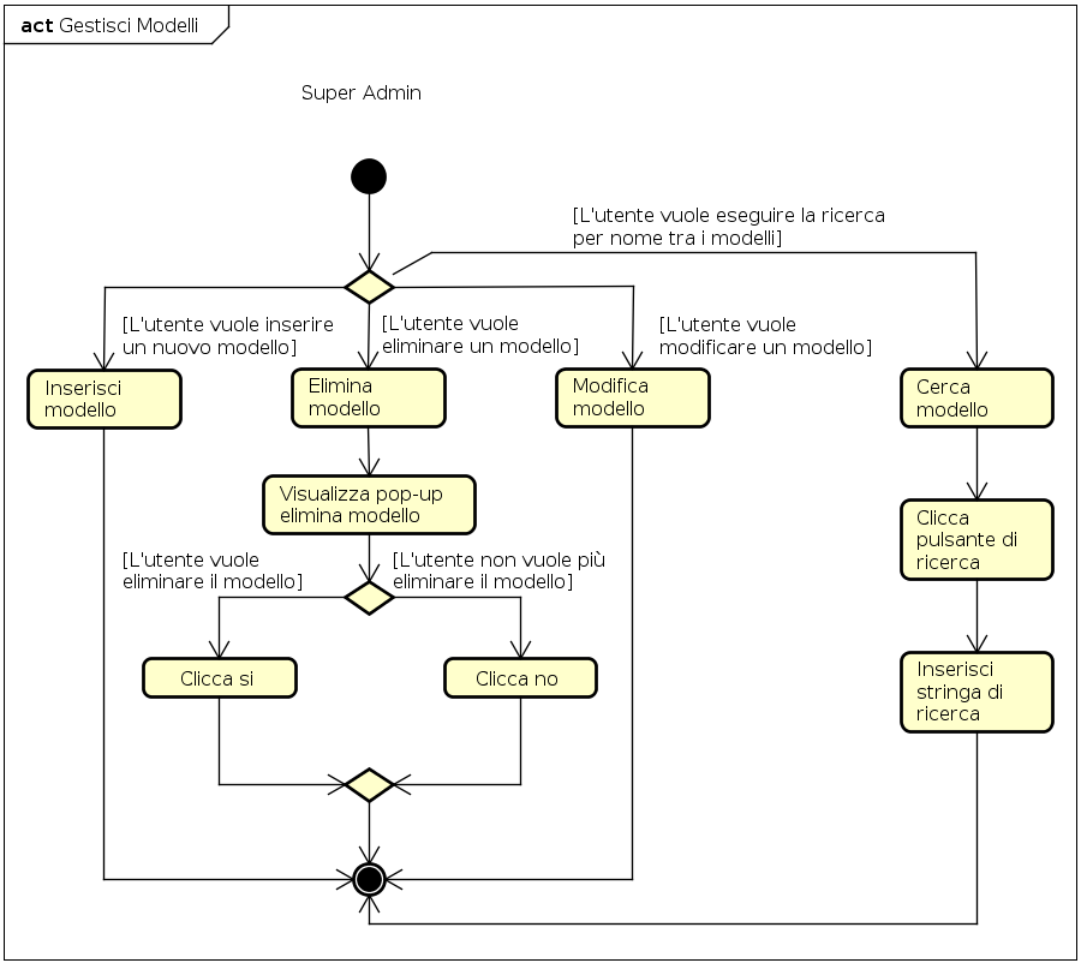

Figura 37: Diagramma di attività - Gestisci Modelli.

5.6.3.2 Gestisci Modelli

Dalla tabella dei modelli sarà possibile eseguire le seguenti operazioni:

- Inserire un nuovo modello all'interno della piattaforma;
- Eliminare un modello già presente nella piattaforma;
- Modificare un modello già presente nella piattaforma;
- Cercare un modello per nome: inserendo una stringa e cliccando sul pulsante di ricerca, la tabella si aggiorna con i risultati trovati.

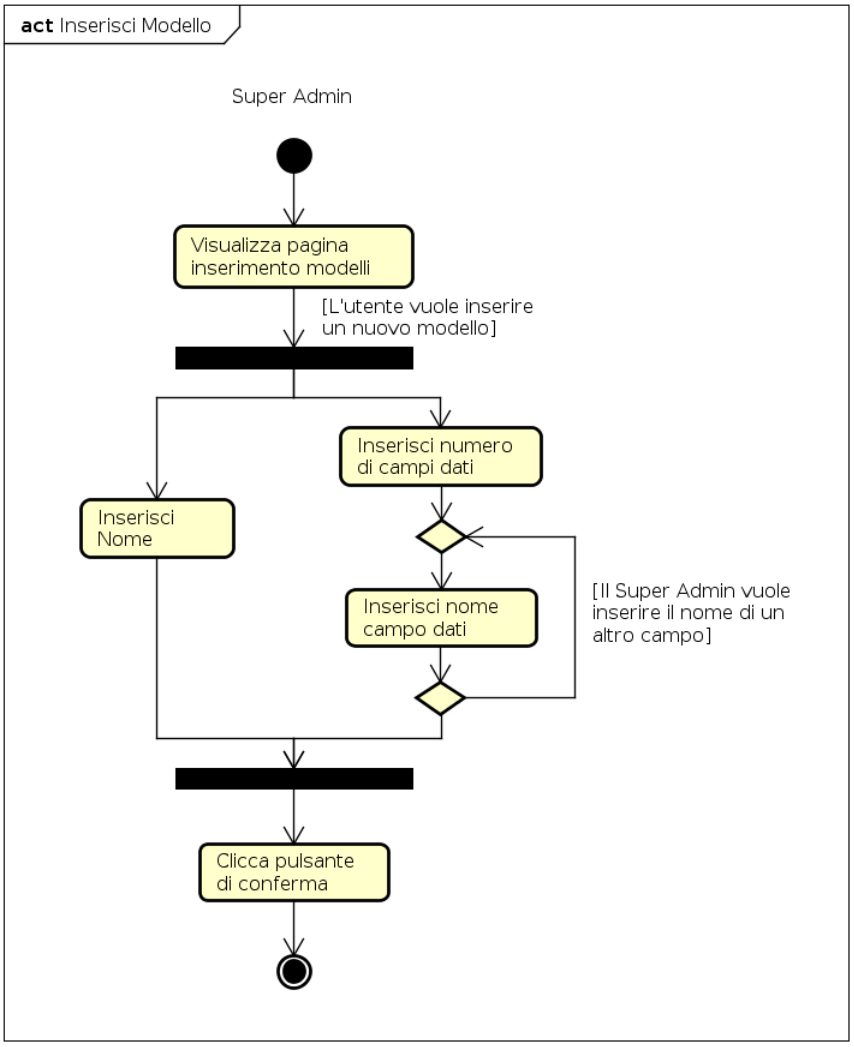

Figura 38: Diagramma di attività - Inserisci Modello.

5.6.3.2.1 Inserisci Modello Premendo l'apposito pulsante si aprirà la pagina per l'inserimento di un nuovo modello dove sarà possibile decidere un nome identificativo per il modello, il numero di dati che gli oggetti di quel determinato modello invieranno alla piattaforma e un nome per ognuno di questi campi dati.

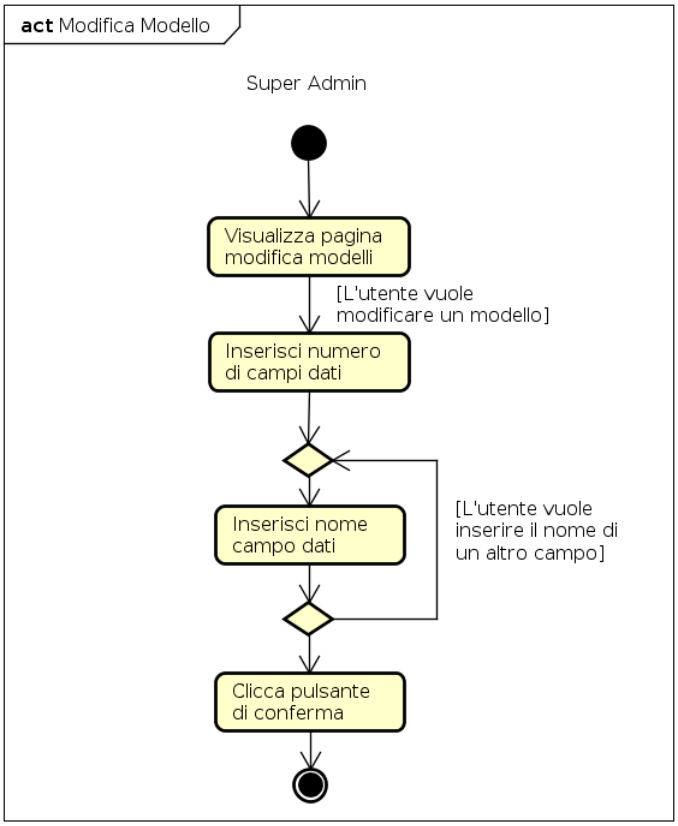

Figura 39: Diagramma di attività - Modifica Modello.

5.6.3.2.2 Modifica Modello Premendo l'apposito pulsante si aprirà la pagina per la modifica del modello selezionato dove sarà possibile modificare numero e nome dei dati che gli oggetti di quel determinato modello invieranno alla piattaforma. Non sarà invece possibile modificare il nome, che lo identifica univocamente.

5.6.4 Gestione Engine

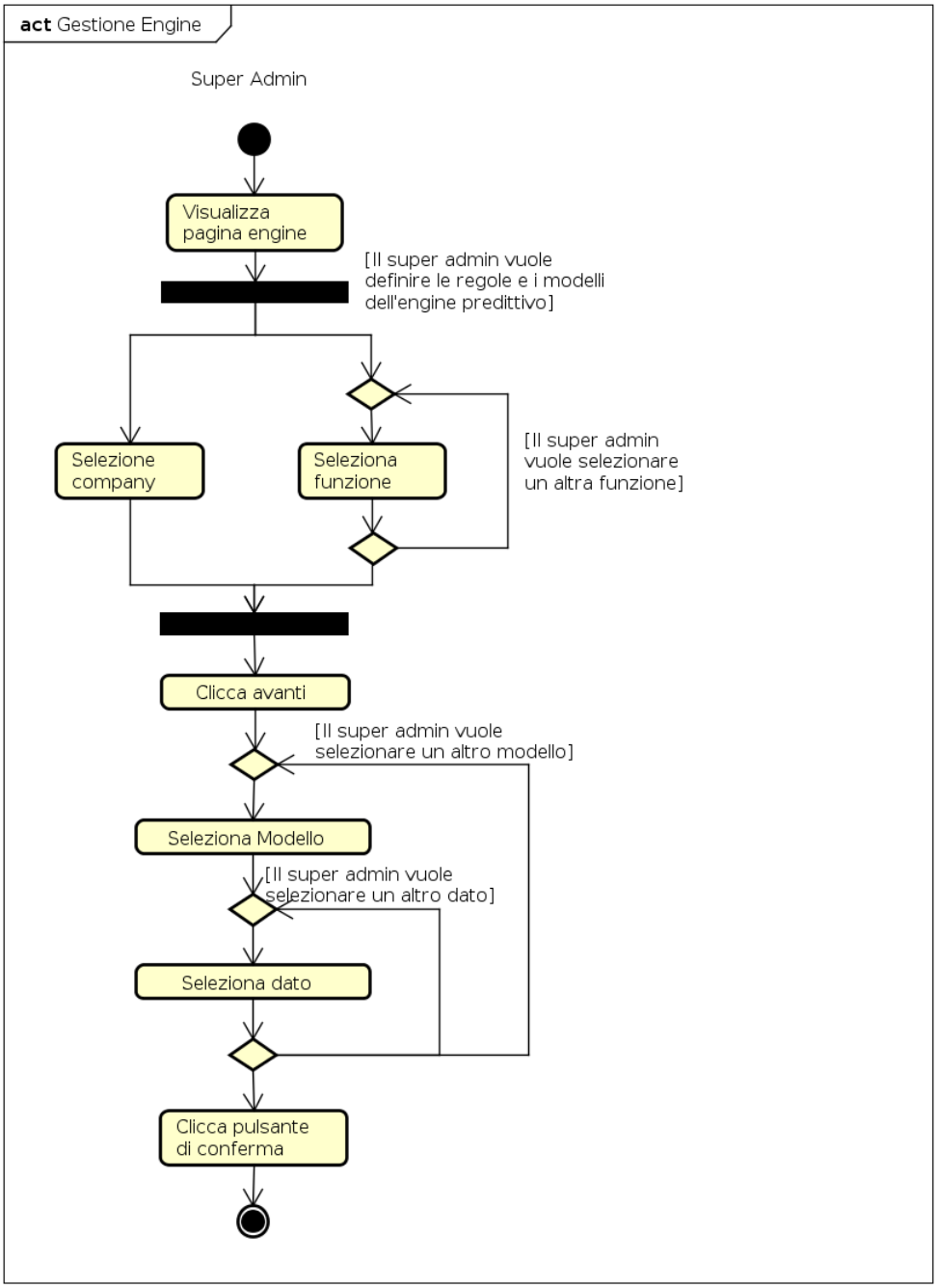

Figura 40: Diagramma di attività - Gestione Engine.

Nella pagina dedicata alla gestione dell'engine è presente un procedimento a due fasi che permetterà al *Super Admin*$_G$ di decidere prima le funzioni e poi i dati che potranno essere utilizzati dall'algoritmo e dagli *Admin*$_G$ per la creazione di grafici e tabelle relative agli oggetti e ai modelli relativi ad essi.

6 Design Pattern

Quando si parla di *Design Pattern*$_G$, si intende un insieme di soluzioni a problemi ricorrenti. Tali *best practice*$_G$ possono essere applicata nell'attività di progettazione, in modo tale da renderne più agevole lo svolgimento. Per convenzione, i *Design Pattern*$_G$ si dividono in:

- ***Design Pattern*$_G$ architetturali**: pensati per definire e strutturare l'architettura del software ad un livello più elevato di astrazione;
- ***Design Pattern*$_G$ creazionali**: definiti per costruire un oggetto senza conoscere la loro implementazione (vanno quindi a nascondere i costruttori delle classi in cui si applicano);
- ***Design Pattern*$_G$ strutturali**: ideati per il riutilizzo di classi già esistenti, fornendone un interfaccia specifica allo scopo;
- ***Design Pattern*$_G$ comportamentali**: sviluppati per consentire legami tra oggetti.

Si andrà a dettagliare in profondità ogni *Design Pattern*$_G$ citato in seguito in Appendice A.

6.1 Design Pattern Architetturali

6.1.1 MVC

- **Scopo dell'utilizzo**: si utilizza il pattern MVC per separare la *business logic*$_G$ dalla rappresentazione grafica;
- **Contesto di utilizzo**: il pattern *MVC*$_G$ viene utilizzato per poter sviluppare correttamente con Play Framework. Tale strumento, infatti, consente di costruire un'applicazione software solamente seguendo il pattern in esame.

6.1.2 Dependency Injection

- **Scopo dell'utilizzo**: si utilizza il pattern *Dependency Injection*$_G$ per separare il comportamento dei componenti dalla risoluzione delle loro dipendenze. Le componenti dichiarano le loro dipendenze e attraverso la *DI*$_G$ il *framework*$_G$ aiuta a cablarle insieme in modo da non dover farlo manualmente.
- **Motivazioni**: utilizziamo la *DI*$_G$ perché presenta vantaggi come consentire di associare facilmente diverse implementazioni per lo stesso componente. Ciò è utile soprattutto per i test, in cui è possibile istanziare manualmente i componenti che utilizzano delle finte dipendenze o iniettare un'implementazione alternativa.
- **Contesto di utilizzo**: il pattern *Dependency Injection*$_G$ viene utilizzato attraverso un impostazione predefinita di *Play Framework*$_G$, che prevede la generazione di una classe routes dove dichiara i suoi controllers come dipendenze nel costruttore. Inoltre per ogni classe all'interno del package controllers viene inserito @Inject() dopo il nome della classe, ma prima della lista dei parametri, attivando cosi il particolare tipo di Injection chiamata Constructor Injection.

6.1.3 DAO

- **Scopo dell'utilizzo**: si utilizza per sollevare i controllers dalle responsabilità delle operazioni che si interfacciano con la base di dati, introducendo un livello intermedio tra questi e i models veri e propri;
- **Contesto di utilizzo**: l'implementazione di tali classi viene separata in un'interfaccia e un'implementazione di quest'ultima in file separati seguendo quindi i principi di encapsulation.

7 Stime di fattibilità e bisogno di risorse

L'architettura sopra definita ha raggiunto un livello di dettaglio tale da poter fornire una stima sulla fattibilità e di bisogno di risorse.
Per quanto riguarda la scelta delle tecnologie da adottare per lo sviluppo del prodotto, si è visto che risultano adeguate a ricoprire le esigenze progettuali.
Gli strumenti scelti sono conosciuti dalla maggioranza dei membri del gruppo, che si impegneranno comunque ad approfondire le loro conoscenze a riguardo. Per riassumere, gli strumenti adottati sono i seguenti:

- **Play Framework**, per la stesura del codice;
- **Amazon Web Services**, che fornisce un pacchetto di *machine learning*$_G$ molto ben strutturato e ideale per il prodotto che si andrà a sviluppare.

Tali strumenti potranno garantire, in unione con le tecnologie scelte, una realizzazione efficace di tutti gli aspetti architetturali.

8 Tracciamento

8.1 Tracciamento requisiti-componenti

Requisiti	Componenti
R0F1	app :: models, app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R0F1.1	app :: models, app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R0F1.1.1	app :: models, app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R0F1.1.2	app :: models, app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R0F1.1.3	app :: models, app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R0F2	app :: models, app :: controllers :: shared :: authentication, app :: controllers :: shared :: account, app :: view :: shared :: authentication
R1F2.1	app :: models, app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R0F2.2	app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R0F2.2.1	app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R1F3	app :: models, app :: controllers :: shared :: account, app :: view :: shared :: authentication
R1F3.1	app :: models, app :: controllers :: shared :: account, app :: controllers :: shared :: authentication, app :: view :: shared :: authentication
R1F3.1.1	app :: models, app :: controllers :: shared :: account, app :: view :: shared :: authentication
R1F3.2	app :: controllers :: shared :: account, app :: view :: shared :: authentication

R1F3.3	app :: controllers :: shared :: account, app :: view :: shared :: authentication,
R1F3.3.1	app :: controllers :: account, app :: views :: shared :: authentication
R1F4	app :: models, app :: controllers :: shared :: authentication, app :: views :: shared :: authentication
R1F4.1	app :: controllers :: shared :: authentication
R0F5	app :: models, app :: models :: daos :: user, app :: controllers :: shared :: account, app :: views :: shared :: account
R2F5.1	app :: models, app :: models :: daos :: user, app :: controllers :: shared :: account, app :: views :: shared :: account
R2F5.2	app :: models, app :: models :: daos :: user, app :: controllers :: shared :: account, app :: views :: shared :: account
R0F5.3	app :: models, app :: models :: daos :: user, app :: controllers :: shared :: account , app :: views :: shared :: account
R0F5.4	app :: models, app :: models :: daos :: user, app :: controllers :: shared :: account, app :: views :: shared :: account
R1F6	app :: models, app :: models :: daos :: user, app :: controllers :: shared :: account, app :: views :: shared :: account
R0F7	app :: models, app :: model :: daos :: company, app :: controllers :: user app :: views :: user :: thingType
R0F8	app :: models, app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1	app :: models, app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1.1	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1.2	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin

R0F8.1.3	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1.4	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1.5	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1.6	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1.7	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.1.8	app :: models app :: model :: daos :: company, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.1	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.1.1	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.1.2	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.1.3	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.1.4	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.1.5	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin

R0F8.2.2	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.3	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.3.1	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.3.2	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.3.3	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.4	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.4.1	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.5	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R0F8.2.6	app :: models, app :: models :: daos :: thing, app :: controllers :: superAdmin, app :: views :: superAdmin
R1F8.3	app :: models, app :: models :: daos :: thing, app :: controllers :: admin, app :: views :: admin
R1F8.3.1	app :: models, app :: models :: daos :: thing, app :: controllers :: admin, app :: views :: admin
R1F8.3.1.1	app :: models, app :: models :: daos :: thing, app :: controllers :: admin, app :: views :: admin
R1F8.3.2	app :: models, app :: models :: daos :: thing, app :: controllers :: admin, app :: views :: admin
R0F8.4	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin

R0F8.4.1	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.1.1	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.1.2	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.1.3	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.1.4	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.1.5	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.1.6	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.2	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.2.1	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.2.2	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.2.3	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin

R0F8.4.2.4	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.3	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.3.1	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.3.2	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F8.4.4	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R1F8.4.5	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R1F8.4.6	app :: models, app :: models :: daos :: company, app :: models :: daos :: user, app :: controllers :: superAdmin, app :: controllers :: admin, app :: views :: superAdmin, app :: views :: admin
R0F9	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.1	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.1.1	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.1.2	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.1.2.1	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType

R0F9.1.2.2	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.2	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.2.1	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.2.2	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.2.2.1	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.2.2.2	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.3	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.4	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.5	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.6	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.7	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F9.7.1	app :: models, app :: models :: daos :: thingType app :: controllers :: superAdmin, app :: views :: SuperAdmin :: thingType
R0F10	app :: controllers :: shared :: administrator, app :: views :: superAdmin :: engine, app :: views :: admin :: engine

R0F10.1	app :: controllers :: shared :: administrator, app :: views :: superAdmin :: engine, app :: views :: admin :: engine
R0F10.2	app :: controllers :: shared :: administrator, app :: views :: superAdmin :: engine, app :: views :: admin :: engine
R0F10.3	app :: controllers :: shared :: administrator, app :: views :: superAdmin :: engine, app :: views :: admin :: engine
R0F11	app :: models, app :: controllers :: shared :: administrator, app :: controllers :: admin, app :: views :: admin :: chart
R0F11.1	app :: models, app :: controllers :: shared :: administrator, app :: controllers :: admin, app :: views :: admin :: chart
R0F11.2	app :: models, app :: controllers :: shared :: administrator, app :: controllers :: admin, app :: views :: admin :: chart
R0F11.3	app :: models, app :: controllers :: shared :: administrator, app :: controllers :: admin, app :: views :: admin :: chart
R0F11.3.1	app :: models, app :: controllers :: shared :: administrator, app :: controllers :: admin, app :: views :: admin :: chart
R0F11.3.2	app :: models, app :: controllers :: shared :: administrator, app :: controllers :: admin, app :: views :: admin :: chart
R0F11.3.3	app :: models, app :: controllers :: shared :: administrator, app :: controllers :: admin, app :: views :: admin :: chart
R0F12	app :: models, app :: models :: daos :.thing, app :: controllers :: admin, app :: controllers :: user, app :: views :: user :: thing.
R0F13	app :: models, app :: models :: daos :.thing, app :: controllers :: admin, app :: controllers :: user, app :: views :: user :: thing.
R0F13.1	app :: models, app :: models :: daos :.thing, app :: controllers :: admin, app :: controllers :: user, app :: views :: user :: thing.

R0F14	app :: models, app :: models :: daos :.thing, app :: controllers :: admin, app :: controllers :: user, app :: views :: admin :: thing.
R0F14.1	app :: models, app :: models :: daos :.thing, app :: controllers :: admin, app :: controllers :: user, app :: views :: admin :: thing,
R0F15	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F15.1	app :: models, app :: models :: daos :: notification, app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications.
R0F15.1.1	app :: models, app :: models :: daos :: notification, app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications.
R0F15.1.2	app :: models, app :: models :: daos :: notification, app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications.
R0F15.1.3	app :: models, app :: models :: daos :: notification, app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications.
R0F15.1.4	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F15.1.5	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F15.1.6	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F15.2	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F15.2.1	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F15.2.2	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications

R0F15.2.3	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F15.2.4	app :: models, app :: models :: daos :: notification app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: notifications
R0F16	app :: models, app :: models :: daos :: thing, app :: controllers :: shared ::authentication
R0F16.1	app :: models, app :: models :: daos :: thing, app :: controllers :: shared ::authentication
R0F16.2	app :: models, app :: models :: daos :: thing, app :: controllers :: shared ::authentication
R0F17	app :: models, app :: models :: daos :: thing, app :: controllers :: shared ::authentication
R0F18	app :: models, app :: models :: daos :: thing, app :: controllers :: shared ::authentication
R1F19	app :: models, app :: models :: daos :: thing, app :: controllers :: admin :: thing, app :: controllers :: admin :: thingType, app :: views :: admin :: thing :: thingthingTypeDetails, app :: views :: admin :: thingType
R0F20	app :: views e tutti i package interni.
R0F21	app :: views :: admin :: chart
R0F22	app :: controllers :: shared :.administrator
R0F23	app :: models :: daos
R0F23.1	app :: models :: daos
R0F23.2	app :: models :: daos
R0F24	app :: models, app :: controllers :: shared ::authentication, app :: controllers :: shared :.administrator, app :: controllers :: shared :: account
R0F25	app :: models, app :: models :: daos :: thingType, app :: controllers :: admin, app :: controllers :: user, app :: views :: admin :: thingType
R0F26	app :: models, app :: models :: daos :: thingType, app :: models :: daos :: chart, app :: controllers :: admin, app :: controllers :: user, app :: views :: admin :: thingType

R0F27	app :: models, app :: models :: daos :: thing app :: controllers :: admin app :: views :: admin
R0F27.1	app :: models, app :: models :: daos :: thing app :: controllers :: admin app :: views :: admin
R0F28	app :: models, app :: controllers :: dao, app :: controllers :: admin, app :: views :: admin :: thingType,
R0F28.1	app :: models, app :: controllers :: dao, app :: controllers :: admin, app :: views :: admin :: thingType,
R0F28.1.1	app :: models, app :: controllers :: dao, app :: controllers :: admin, app :: views :: admin :: thingType,
R0F29	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F29.1	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F29.2	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F29.3	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F29.4	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F29.5	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F29.6	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F29.7	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F30	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company

R0F31	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F31.1	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F32	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F33	app :: models, app :: controllers :: dao :: company, app :: controllers :: superAdmin, app :: views :: superAdmin :: company
R0F34	app :: models, app: models :: dao, app :: controllers :: admin, app :: controllers :: user, app :: views :: shared :: adminUser :: things
R0F34.1	app :: models, app: models :: dao, app :: controllers :: admin, app :: controllers :: user, app :: views :: shared :: adminUser :: things
R0F35	app :: models, app: models :: dao, app :: controllers :: admin, app :: controllers :: user, app :: views :: shared :: adminUser :: thingsType
R0F35.1	app :: models, app: models :: dao, app :: controllers :: admin, app :: controllers :: user, app :: views :: shared :: adminUser :: thingsType
R0F36	app :: models, app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: Notifications
R0F36.1	app :: models, app :: controllers :: shared :: adminUser, app :: views :: shared :: adminUser :: Notifications
R0F37	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: superAdmin :: company
R0F38	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: user :: thing
R0F38.1	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: user :: chart

R0F39	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: admin :: engine
R0F39.1	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: admin :: engine
R0F39.2	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: admin :: engine
R0F39.3	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: admin :: engine
R0F40	app :: models, app :: models :: daos, app :: controllers :: shared :: administrator, app :: views :: admin :: engine

Tabella 2: Tracciamento requisiti-componenti.

8.2 Tracciamento componenti-requisiti

Componenti	**Requisiti**
Requisiti	
app :: models	R0F1, R0F1.1, R0F1.1.1, R0F1.1.2, R0F1.1.3, R0F2, R0F2.1, R1F3, R1F3.1, R1F3.1.1, R1F4, R0F5, R2F5.1, R2F5.2, R0F5.3, R0F5.4, R1F6, R0F7, R0F8, R0F8.1, R0F8.1.1, R0F8.1.2, R0F8.1.3, R0F8.1.4, R0F8.1.5, R0F8.1.6, R0F8.1.7, R0F8.1.8, R0F8.2, R0F8.2.1, R0F8.2.1.1, R0F8.2.1.2, R0F8.2.1.3, R0F8.2.1.4, R0F8.2.1.5, R0F8.2.2, R0F8.2.3, R0F8.2.3.1, R0F8.2.3.2, R0F8.2.3.3, R0F8.2.3.4, R0F8.2.4.1, R0F8.2.5, R0F8.2.6, R1F8.3, R1F8.3.1, R1F8.3.1.1, R1F8.3.2, R0F8.4, R0F8.4.1, R0F8.4.1.1, R0F8.4.1.2, R0F8.4.1.3, R0F8.4.1.4, R0F8.4.1.5, R0F8.4.1.6, R0F8.4.2, R0F8.4.2.1, R0F8.4.2.2, R0F8.4.2.3,R0F8.4.2.4, R0F8.4.3, R0F8.4.3.1, R0F8.4.3.2, R0F8.4.4, R1F8.4.5, R1F8.4.6, R0F9, R0F9.1.1, R0F9.1.2, R0F9.1.2.1, R0F9.1.2.2, R0F9.2, R0F9.2.1, R0F9.2.2, R0F9.2.2.1, R0F9.2.2.2, R0F9.3, R0F9.4, R0F9.5, R0F9.6, R0F9.7, R0F9.7.1, R0F11, R0F11.1, R0F11.2, R0F11.3, R0F11.3.1, R0F11.3.2, R0F11.3.3, R0F12, R0F13, R0F13.1, R0F14, R0F14.1, R0F15, R0F15.1, R0F15.1.1, R0F15.1.2, R0F15.1.3, R0F15.1.1.4, R0F15.1.1.5, R0F15.1.1.6, R0F15.2, R0F15.2.1, R0F15.2.2, R0F15.2.3, R0F15.2.4, R0F16, R0F16.1, R0F16.2, R0F17, R0F18, R1F19, R0F24, R0F25, R0F26, R0F27, R0F27.1, R0F28, R0F28.1, R0F28.1.1, R0F29, R0F29.1, R0F29.2, R0F29.3, R0F29.4, R0F29.5, R0F29.6, R0F29.7, R0F30, R0F31, R0F31.1, R0F32, R0F33
app :: models :: daos :: user	R0F5, R2F5.1, R2F5.2, R0F5.3, R0F5.4, R2F6, R0F8.4, R0F8.4.1, R0F8.4.1.1, R0F8.4.1.2, R0F8.4.1.3, R0F8.4.1.4, R0F8.4.1.5, R0F8.4.1.6, R0F8.4.2, R0F8.4.2.1, R0F8.4.2.2, R0F8.4.2.3,R0F8.4.2.4, R0F8.4.3, R0F8.4.3.1, R0F8.4.3.2, R0F8.4.4, R1F8.4.5, R1F8.4.6
app :: model :: daos :: company	R0F7, R0F8, R0F8.1, R0F8.1.1, R0F8.1.2, R0F8.1.3, R0F8.1.4, R0F8.1.5, R0F8.1.6, R0F8.1.7, R0F8.1.8, R0F8.4, R0F8.4.1, R0F8.4.1.1, R0F8.4.1.2, R0F8.4.1.3, R0F8.4.1.4, R0F8.4.1.5, R0F8.4.1.6, R0F8.4.2, R0F8.4.2.1, R0F8.4.2.2, R0F8.4.2.3,R0F8.4.2.4, R0F8.4.3, R0F8.4.3.1, R0F8.4.3.2, R0F8.4.4, R1F8.4.5, R1F8.4.6, R0F29, R0F29.1, R0F29.2, R0F29.3, R0F29.4, R0F29.5, R0F29.6, R0F29.7, R0F30, R0F31, R0F31.1, R0F32, R0F33

app :: models :: daos :: thing	R0F8.2, R0F8.2.1, R0F8.2.1.1, R0F8.2.1.2, R0F8.2.1.3, R0F8.2.1.4, R0F8.2.1.5, R0F8.2.2, R0F8.2.3, R0F8.2.3.1, R0F8.2.3.2, R0F8.2.3.3, R0F8.2.3.4, R0F8.2.4.1, R0F8.2.5, R0F8.2.6, R1F8.3, R1F8.3.1, R1F8.3.1.1, R1F8.3.2, R0F9, R0F9.1.1, R0F9.1.2, R0F9.1.2.1, R0F9.1.2.2, R0F9.2, R0F9.2.1, R0F9.2.2, R0F9.2.2.1, R0F9.2.2.2, R0F9.3, R0F9.4, R0F9.5, R0F9.6, R0F9.7, R0F9.7.1, R0F12, R0F13, R0F13.1, R0F14, R0F14.1, R0F16, R0F16.1, R0F16.2, R0F17, R0F18, R1F19, R0F27, R0F27.1
app :: models :: daos	R0F23, R0F23.1, R0F23.2, R0F28, R0F28.1, R0F28.1.1
app :: models :: daos :: chart	R0F26
app :: models :: daos :: thingType	R0F25, R0F26
app :: models :: daos :: notification	R0F15, R0F15.1, R0F15.1.1, R0F15.1.2, R0F15.1.3, R0F15.1.1.4, R0F15.1.1.5, R0F15.1.1.6, R0F15.2, R0F15.2.1, R0F15.2.2, R0F15.2.3, R0F15.2.4
app :: controllers :: shared :: account	R0F1, R0F1.1, R0F1.1.1, R0F1.1.2, R0F1.1.3, R0F2, R0F2.1, R0F2.2, R0F2.2.1, R1F3, R1F3.1, R1F3.1.1, R1F3.2, R1F3.3, R0F5, R2F5.1, R2F5.2, R0F5.3, R0F5.4, R1F6, R0F24
app :: controllers :: shared :: adminUser	R0F15, R0F15.1, R0F15.1.1, R0F15.1.2, R0F15.1.3, R0F15.1.1.4, R0F15.1.1.5, R0F15.1.1.6, R0F15.2, R0F15.2.1, R0F15.2.2, R0F15.2.3, R0F15.2.4
app :: controllers :: shared :: authentication	R0F1, R0F1.1, R0F1.1.1, R0F1.1.2, R0F1.1.3, R0F2, R0F2.1, R0F2.2, R0F2.2.1, R1F3, R1F3.1, R1F4, R1F4.1,R0F16, R0F16.1, R0F16.2, R0F17, R0F18, R0F24
app :: controllers :: shared :: administrator	R0F10, R0F10.1, R0F10.2, R0F10.3, R0F11, R0F11.1, R0F11.2, R0F11.3, R0F11.3.1, R0F11.3.2, R0F11.3.3, R0F22, R0F24

app :: controllers :: superAdmin	R0F8, R0F8.1, R0F8.1.1, R0F8.1.2, R0F8.1.3, R0F8.1.4, R0F8.1.5, R0F8.1.6, R0F8.1.7, R0F8.1.8, R0F8.2, R0F8.2.1, R0F8.2.1.1, R0F8.2.1.2, R0F8.2.1.3, R0F8.2.1.4, R0F8.2.1.5, R0F8.2.2, R0F8.2.3, R0F8.2.3.1, R0F8.2.3.2, R0F8.2.3.3, R0F8.2.3.4, R0F8.2.4.1, R0F8.2.5, R0F8.2.6, R0F8.4, R0F8.4.1, R0F8.4.1.1, R0F8.4.1.2, R0F8.4.1.3, R0F8.4.1.4, R0F8.4.1.5, R0F8.4.1.6, R0F8.4.2, R0F8.4.2.1, R0F8.4.2.2, R0F8.4.2.3,R0F8.4.2.4, R0F8.4.3, R0F8.4.3.1, R0F8.4.3.2, R0F8.4.4, R1F8.4.5, R1F8.4.6, R0F9, R0F9.1.1, R0F9.1.2, R0F9.1.2.1, R0F9.1.2.2, R0F9.2, R0F9.2.1, R0F9.2.2, R0F9.2.2.1, R0F9.2.2.2, R0F9.3, R0F9.4, R0F9.5, R0F9.6, R0F9.7, R0F9.7.1, R0F29, R0F29.1, R0F29.2, R0F29.3, R0F29.4, R0F29.5, R0F29.6, R0F29.7, R0F30, R0F31, R0F31.1, R0F32, R0F33
app :: controllers :: user	R0F12, R0F13, R0F13.1, R0F14, R0F14.1, R0F25, R0F26, R0F7
app :: controllers :: admin	R1F8.3, R1F8.3.1, R1F8.3.1.1, R1F8.3.2, R0F8.4, R0F8.4.1, R0F8.4.1.1, R0F8.4.1.2, R0F8.4.1.3, R0F8.4.1.4, R0F8.4.1.5, R0F8.4.1.6, R0F8.4.2, R0F8.4.2.1, R0F8.4.2.2, R0F8.4.2.3,R0F8.4.2.4, R0F8.4.3, R0F8.4.3.1, R0F8.4.3.2, R0F8.4.4, R1F8.4.5, R1F8.4.6, R0F11, R0F11.1, R0F11.2, R0F11.3, R0F11.3.1, R0F11.3.2, R0F11.3.3, R0F12, R0F13, R0F13.1, R0F14, R0F14.1, R0F25, R0F26, R0F27, R0F27.1, R0F28, R0F28.1, R0F28.1.1
app :: controllers :: admin :: thing	R1F19
app :: controllers :: admin :: thingType	R1F19, R0F25, R0F26
app :: view :: shared :: authentication	R0F1, R0F1.1, R0F1.1.1, R0F1.1.2, R0F1.1.3, R0F2, R0F2.1, R0F2.2, R0F2.2.1, R1F3, R1F3.1, R1F3.1.1, R1F3.2, R1F3.3, R1F4
app :: view :: shared :: notification	R0F15, R0F15.1, R0F15.1.1, R0F15.1.2, R0F15.1.3, R0F15.1.1.4, R0F15.1.1.5, R0F15.1.1.6, R0F15.2, R0F15.2.1, R0F15.2.2, R0F15.2.3, R0F15.2.4
app :: view :: shared :: account	R0F5, R2F5.1, R2F5.2, R0F5.3, R0F5.4, R1F6

app :: view :: superAdmin	R0F8, R0F8.1, R0F8.1.1, R0F8.1.2, R0F8.1.3, R0F8.1.4, R0F8.1.5, R0F8.1.6, R0F8.1.7, R0F8.1.8, R0F8.2, R0F8.2.1, R0F8.2.1.1, R0F8.2.1.2, R0F8.2.1.3, R0F8.2.1.4, R0F8.2.1.5, R0F8.2.2, R0F8.2.3, R0F8.2.3.1, R0F8.2.3.2, R0F8.2.3.3, R0F8.2.3.4, R0F8.2.4.1, R0F8.2.5, R0F8.2.6, R0F8.4, R0F8.4.1, R0F8.4.1.1, R0F8.4.1.2, R0F8.4.1.3, R0F8.4.1.4, R0F8.4.1.5, R0F8.4.1.6, R0F8.4.2, R0F8.4.2.1, R0F8.4.2.2, R0F8.4.2.3,R0F8.4.2.4, R0F8.4.3, R0F8.4.3.1, R0F8.4.3.2, R0F8.4.4, R1F8.4.5, R1F8.4.6
app :: views :: superAdmin :: thingType	R0F9, R0F9.1.1, R0F9.1.2, R0F9.1.2.1, R0F9.1.2.2, R0F9.2, R0F9.2.1, R0F9.2.2, R0F9.2.2.1, R0F9.2.2.2, R0F9.3, R0F9.4. R0F9.5, R0F9.6, R0F9.7, R0F9.7.1
app :: views :: superAdmin :: company	R0F29, R0F29.1, R0F29.2, R0F29.3, R0F29.4, R0F29.5, R0F29.6, R0F29.7, R0F30, R0F31, R0F31.1, R0F32, R0F33
app :: view :: admin	R1F8.3, R1F8.3.1, R1F8.3.1.1, R1F8.3.2, R0F8.4, R0F8.4.1, R0F8.4.1.1, R0F8.4.1.2, R0F8.4.1.3, R0F8.4.1.4, R0F8.4.1.5, R0F8.4.1.6, R0F8.4.2, R0F8.4.2.1, R0F8.4.2.2, R0F8.4.2.3,R0F8.4.2.4, R0F8.4.3, R0F8.4.3.1, R0F8.4.3.2, R0F8.4.4, R1F8.4.5, R1F8.4.6, R0F27, R0F27.1
app :: view :: superAdmin :: engine	R0F10, R0F10.1, R0F10.2, R0F10.3
app :: view :: admin :: engine	R0F10, R0F10.1, R0F10.2, R0F10.3
app :: view :: admin :: chart	R0F11, R0F11.1, R0F11.2, R0F11.3, R0F11.3.1, R0F11.3.2, R0F11.3.3, R0F21
app :: view :: admin :: thing :: thingTypeDetails	R1F19
app :: view :: admin :: thingType	R1F19, R0F28, R0F28.1, R0F28.1.1
app :: view :: user :: thing	R0F12, R0F13, R0F13.1, R0F14, R0F14.1, R0F7

Tabella 3: Tracciamento componenti-requisiti.

A Descrizione Design Pattern

A.1 Design Pattern Architetturali

A.1.1 MVC

Model-View-Controller (MVC) è un pattern architetturale atto all'implementazione di interfacce utente. L'applicazione, per poter essere coerente con la struttura proposta da MVC, deve separare i componenti software che implementano il modello delle funzionalità di business dai componenti che implementano la logica di presentazione e di controllo che utilizzano tali funzionalità. Vengono quindi definite tre tipologie di componenti che soddisfano tali requisiti:

- **Model**: implementa le funzionalità di business;
- **View**: implementa la logica di presentazione;
- **Controller**: implementa la logica di controllo.

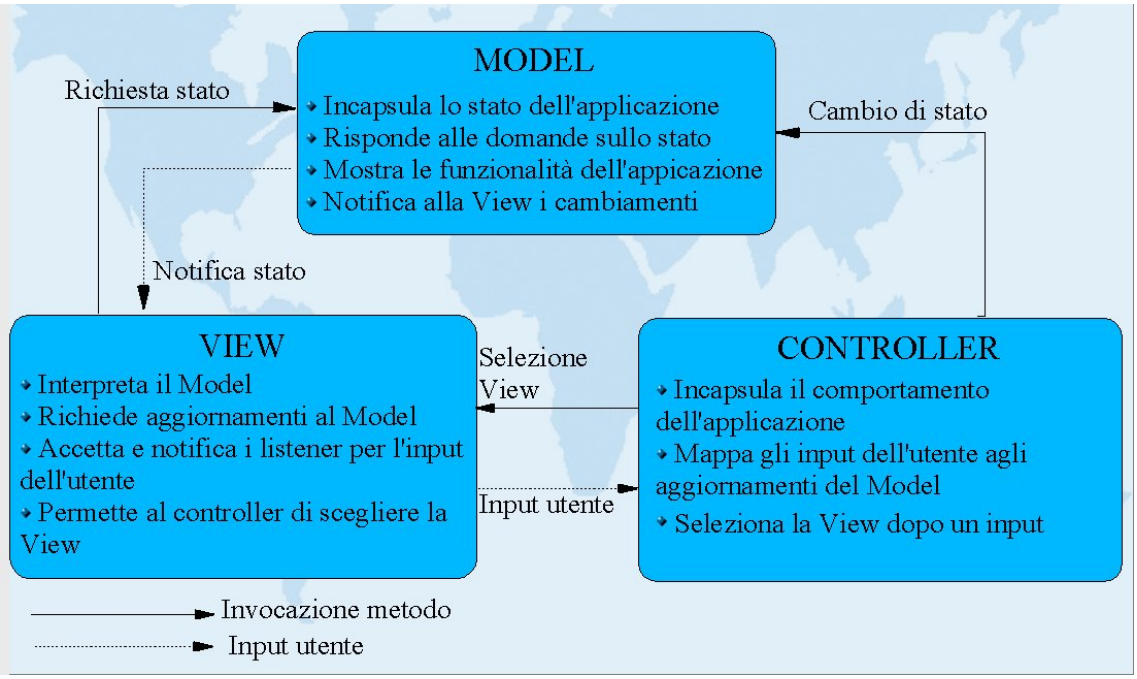

Figura 41: MVC - Diagramma di interazione

A.1.1.1 Model

Il *core* dell'applicazione viene implementato dal Model, che incapsulando lo stato della stessa definisce i dati e le operazioni che possono essere eseguite su questi. Quindi definisce le regole di business per l'interazione con i dati, esponendo alla View ed al Controller rispettivamente le funzionalità per l'accesso e l'aggiornamento. Per lo sviluppo del Model quindi è solito utilizzare le tipiche tecniche di progettazione *object oriented*, al fine di ottenere un componente software che astragga al meglio concetti importati dal mondo reale. Il Model può inoltre avere la responsabilità di notificare ai componenti della View eventuali aggiornamenti verificatisi in seguito a richieste del Controller, al fine di permettere alle View di presentare agli occhi degli utenti dati sempre aggiornati.

A.1.1.2 View

La logica di presentazione dei dati viene gestita solo e solamente dalla View. Ciò implica che questa deve fondamentalmente gestire la costruzione dell'interfaccia grafica che rappresenta il mezzo mediante il quale gli utenti interagiranno con il sistema. Ogni GUI_G può essere costituita da schermate diverse che presentano più modi di interagire con i dati dell'applicazione. Per far sì che i dati presentati siano sempre aggiornati è possibile adottare due strategie note come "push model" e "pull model". Il push model adotta il pattern **Observer**, registrando le View come osservatori del Model. Le View possono quindi richiedere gli aggiornamenti al Model in tempo reale grazie alla notifica di quest'ultimo. Benché questa rappresenti la strategia ideale, non è sempre applicabile, in quanto richiede che tutta l'applicazione sia sviluppata nella medesima tecnologia. In tali casi è possibile utilizzare il pull model, dove la View richiede gli aggiornamenti quando "lo ritiene opportuno". Inoltre la View delega al Controller l'esecuzione dei processi richiesti dall'utente dopo averne catturato gli input e la scelta delle eventuali schermate da presentare.

A.1.1.3 Controller

Questo componente ha la responsabilità di trasformare le interazioni dell'utente della View in azioni eseguite dal Model. Il Controller non rappresenta un semplice ponte tra View e Model. Realizzando la mappatura tra input dell'utente e processi eseguiti dal Model e selezionando la schermate della View richieste, il Controller implementa la logica di controllo dell'applicazione.

A.1.2 Dependency Injection

Dependency Injection (DI) è un pattern architetturale, nello specifico è una particolare forma del pattern Inversion Of Control. L'idea alla base della Dependency Injection è quella di avere un componente esterno (assembler) che si occupi della creazione degli oggetti e delle loro relative dipendenze e di assemblarle mediante l'utilizzo dell'injection. Con la Dependency Injection una classe o un sistema non è responsabile circa l'inizializzazione delle proprie dipendenze. L'obiettivo di tale tecnica è quello di allentare l'associazione tra un oggetto e ed un altro, inoltre la riduzione della dipendenza tra una classe e l'altra, facilita le fasi di unit testing.

Esistono diversi tipi di injection:

- **Constructor Injection**: dove la dipendenza viene iniettata tramite l'argomento del costruttore;
- **Setter Injection**: dove la dipendenza viene iniettata attraverso un metodo "set";

A.1.2.1 Constructor Injection

In questo caso la dipendenza viene iniettata all'interno la lista dei parametri del costruttore dell'oggetto, in questo caso si posso rilevare alcuni vantaggi:

- costruzione di oggetti validi fin dalla loro istanziazione, poiché passa dal suo costruttore;
- possibile costruzione di oggetti immutabili.

Tuttavia questo metodo presenta anche qualche aspetto negativo:

- telescoping, in caso di molte dipendenze il costruttore potrebbe avere un numero elevato di parametri;
- difficoltà con comprendere il significato di tutti i parametri.

A.1.2.2 Setter Injection

In questo caso la dipendenza viene iniettata attraverso i metodi setter dell'oggetto, in questo caso possibili vantaggi sono:

- individuazione precisa delle dipendenze e dei loro nomi;
- miglior feeling con le gerarchie di classi.

Come nel metodo precedente anche in questo caso si riscontra qualche aspetto negativo:

- componente costruita per passi;
- non è abilitante all'immutabilità, in un primo momento l'injector infatti, costruirà attraverso il costruttore di default un oggetto vuoto e successivamente andrà a risolvere le dipendenze.

A.1.3 Data access object

Data access object (DAO) è un pattern architetturale, che fornisce un'interfaccia astratta di un certo tipo di database o altro meccanismo di persistenza. DAO fornisce alcune operazioni sui dati specifici senza esporre i dettagli del database. Questo isolamento sostiene il principio di Single responsibility. Il vantaggio relativo all'uso del DAO è dunque il mantenimento di una rigida separazione tra le componenti di un'applicazione, le quali potrebbero essere il "Model" e il "Control" in un'applicazione basata sul paradigma MVC.

Questo pattern dunque fornisce numerosi vantaggi infatti, il suo facile utilizzo permette una rigorosa separazione tra due importanti parti dell'applicazione che non devono sapere nulla l'una dell'altra e che possono prevedere una evoluzione frequente e del tutto indipendente. Tutti i dettagli di storage sono nascosti dal resto dell'applicazione (information hiding). Pertanto, possibili modifiche al meccanismo di persistenza possono essere implementati solo modificando un'implementazione del DAO mentre il resto dell'applicazione non è interessata. DAO funge da intermediario tra l'applicazione e il database, muovendo i dati tra gli oggetti e i record della base di dati. All'interno dei DAO saranno presenti metodi di interrogazione e manipolazione della corrispondente classe di dominio, che conterrà le funzionalità base: CRUD

- **C**reate;
- **R**ead;
- **U**pdate;
- **D**elete.

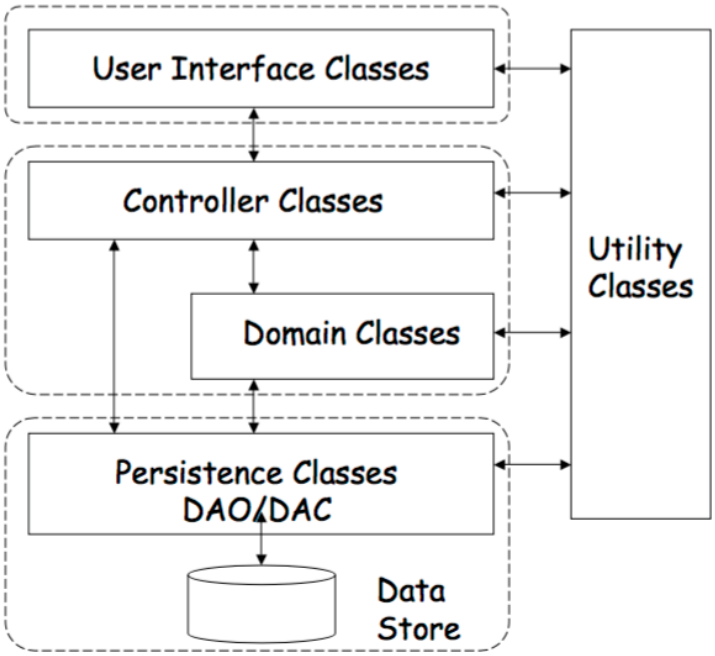

Figura 42: DAO - architettura con DAO