MaaP: MongoDB as an admin Platform

Specifica Tecnica

Versione	1.2.0
Data creazione	xxxx-xx-xx
Data ultima modifica	xxxx-xx-xx
Stato del Documento	Formale
Uso del Documento	Esterno
Redazione	nome1,nome2,...
Verifica	nome1,nome2,...
Approvazione	nome1,nome2,...
Distribuzione	Aperture Software

Sommario

Questo documento si propone di presentare la Specifica tecnica e architetturale per la Realizzazione del prodotto **MaaP**.

Diario delle modifiche

Versione	Data	Autore	Modifiche effettuate
1.2.0	2014-02-xx	... (RE)	Approvazione documento
1.1.0	2014-02-xx	... (VE)	Verifica documento
1.0.4	2014-02-xx	... (AN)	bla bla
1.0.3	2014-02-xx	... (AN)	bla bla
1.0.2	2014-02-xx	... (AN)	bla bla
1.0.0	2014-01-24	Giacomo Pinato (PR)	Prima stesura del documento

Tabella 1: Registro delle modifiche

Indice

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello del progetto **MaaP**, a partire dai requisiti individuati durante l'Analisi. Verrà presentata l'architettura generale secondo la quale saranno organizzate le varie componenti software, i *Design Pattern*$_G$ e le tecnologie utilizzate per poi descrivere più dettagliatamente le varie componenti e relative dipendenze.

1.2 Scopo del prodotto

Lo scopo del prodotto è produrre un framework per generare interfacce web di amministrazione dei dati di business basati sullo stack Node.js e MongoDB.
L'obiettivo è quello di semplificare il lavoro allo sviluppatore che dovrà rispondere in modo rapido e standard alle richieste degli esperti di business.

1.3 Glossario

Al fine di evitare ogni ambiguità nella comprensione del linguaggio utilizzato nel presente documento e, in generale, nella documentazione fornita dal gruppo Aperture Software, ogni termine tecnico, di difficile comprensione o di necessario approfondimento verrà inserito nel documento *Glossario_v1.2.0.pdf*.
Saranno in esso definiti e descritti tutti i termini in corsivo e allo stesso tempo marcati da una lettera "G" maiuscola in pedice nella documentazione fornita.

1.4 Riferimenti

1.4.1 Normativi

- **Analisi dei requisiti**: Analisi_dei_Requisiti_v1.2.0.pdf
- Norme di Progetto: Norme_di_Progetto_v1.2.0.pdf (allegato alla presente documentazione)

1.4.2 Informativi

- Learning Node: O'Reilly Shelley Powers
- AngularJS: O'Reilly Brad Green e Shyam Seshadri
- Software Engineering (8th edition), Ian Sommerville, Pearson Education — Addison-Wesley
- Design Patterns, E. Gamma, R. Helm, R. Johnson, J. Vlissides, Pearson Education — Addison-Wesley
- Dall'idea al codice con UML 2 L. Baresi, L. Lavazza, M. Pianciamore, Pearson Education

2 Tecnologie utilizzate

In questa sezione verrano elencate e descritte le tecnologie che si utilizzeranno durante lo sviluppo del progetto. In particolare la colonna portante del progetto sarà lo stack MEAN, ovvero MongoDB, Express, Angular e Node.js.

2.1 MongoDB

Il database con il quale la nostra applicazione dovrà interagire è realizzato con MongoDB, come specificato nel capitolato. Questa tecnologia offre i seguenti vantaggi:

- Facile indicizzazione: Ogni campo in MongoDB puo' diventare un indice;
- Bilanciamento di carico: MongoDB scala orizzontalmente molto facilmente grazie all'utilizzo di shard;
- Integrazione con Javascript: Query o altre funzioni scritte in Javascript possono essere eseguite direttamente dal dabatase;

2.2 Javascript

Si è deciso di utilizzare Javascript in quanto è il linguaggio su cui si basano tutte le altre tecnologie che andremo ad utilizzare, e offre quindi una facile integrazione, oltre ad essere un ottimo linguaggio per applicazioni web e client side.

2.3 NodeJs

Si è deciso di utilizzare il linguaggio Node.js in quanto consigliato dal capitolato e adatto al progetto. Le sue caratteristiche piu' vantaggiose sono:

- Modello event-driven: ovvero programmazione ad eventi, che si basa su un concetto semplice: il flusso del programma non segue un corso specifico ma è guidato dalle azioni dell'utilizzatore;
- Modello asincrono: grazie a questa caratteristica è possibile ridurre al minimo i tempi di morti in quanto, nell'attesa dell completamento di una operazione, si procede con altri flussi logici.
- Grande scalabilità: Grazie al modo in cui è implementato, Node.js riesce ad essere largamente scalabile con minimo sforzo.

2.4 jQuery

Per migliorare e semplificare la scrittura di funzioni nel linguaggio JavaScript, si è deciso di adottare il framework jQuery. I vantaggi sono:

- Semplicità: l'utilizzo di jQuery semplifica e facilita la scrittura di codice JavaScript, inoltre offre plug-in on-line per fornire nuove funzionalità.

Svantaggi:

- Pericolosità: jQuery offre funzionalità e plug-in molto utili, ma non tutto è compatibile con i vari browser, inoltre alcune funzionalità possono essere vecchie, non aggiornate o scritte male.

2.5 JSON

Rappresenta il tipo di messaggi con cui client e server si scambiano informazioni. I vantaggi offerti sono:

- Semplicità: i messaggi JSON sono più corti rispetto ad altri formati di interscambio, e vengono eseguiti più velocemente dal parser. JSON inoltre risulta più semplice e immediato rispetto ad esempio a XML.

Svantaggi:

- Restrittività: JSON è meno restrittivo rispetto ad XML, e questo può permettere di inserire errori nello scambio di messaggi.

2.6 AngularJs

- Two Way Data-Binding: Una delle caratteristiche principali di angular. Le modifiche apportate al model si rifletto direttamente sugli elementi del DOM, e le modifiche al DOM si ripercuotono automaticamente sul model. Questo alleggerisce tremendamente il codice necessario a controllare ad ascoltare e gestire il DOM, automatizzando il processo. E noi sappiamo che automatico è bene.
- Templates: I template HTML sono parsati dal browswe nel DOM,il quale costituisce poi l'input per il compilatore Angular. Quest'ultimo poi crea il data binding tra il DOM e lo scope dei dati. Uno dei piu' grandi vantaggi di questa tecnica e' che separa presentazione da implementazione, in quanto i template html possono modificati senza alterare il modo in cui sono inseriti i dati.
- Dependecy Injection: Angular possiede nativamente una "dependecy injection", che aiuta gli sviluppatori permettendo facilitando lo sviluppo, la compresione e il testing dell'applicazione.
- Directives: Le directives possono essere usate per definire tag HTML personalizzati che fungono da widget. Possono inoltre essere usate per "decorare" elementi con comportamenti personalizzati o per manipolare attributi del DOM.

2.7 HTML5

3 Descrizione architettura

3.1 Metodo e formalismo di specifica

Si è deciso di procedere utilizzando un approccio top-down per l'esposizione dell'architettura dell'applicazione, ovvero descrivendo inizialmente le componenti in generale per poi arrivare a trattarle al particolare. Si descriveranno i package e i componenti per poi dettagliare le singole classi, specificando per ciascuna di esse il tipo, l'obiettivo e la funzionalità. Poi si passerà ad illustrare degli esempi d'uso di Design Pattern (descritti approfonditamente nell' Appendice A) e le tecnologie utilizzate. Per facilitare la lettura dei diagrammi di package e di classe si farà uso di vari colori per distinguere le componenti.

3.2 Architettura generale

L'architettura generale del prodotto è basata sul Design Pattern MVC ed è quindi suddivisa in tre componenti principali: Model, View e Controller. Il Model e il Controller si trovano sul lato server dell'applicazione, mentre la View viene collocata a lato client. Quest'ultima a sua volta implementa il Design Pattern MVW, su cui si basa Angular, per il mantenimento temporaneo dei dati e la loro visualizzazione.

Il seguente diagramma rappresenta l'architettura ad alto livello del framework, indicando i package e le relazioni che intercorrono tra questi.
...TODO aggiungere diagramma dell'installer CLI comprensivo di descrizione...

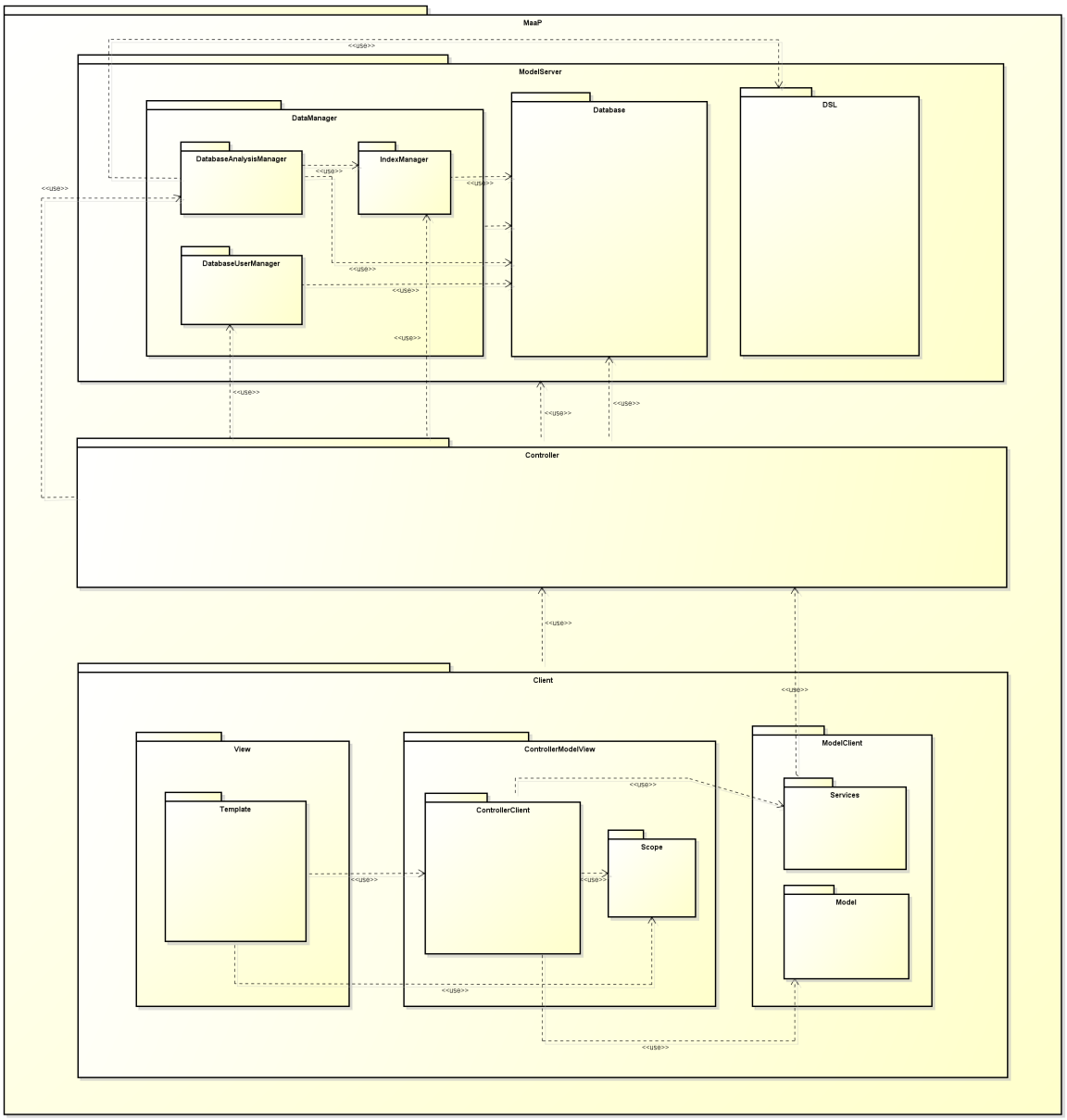

Figura 1: MaaP, Diagramma generale package

...TODO aggiungere descrizione...

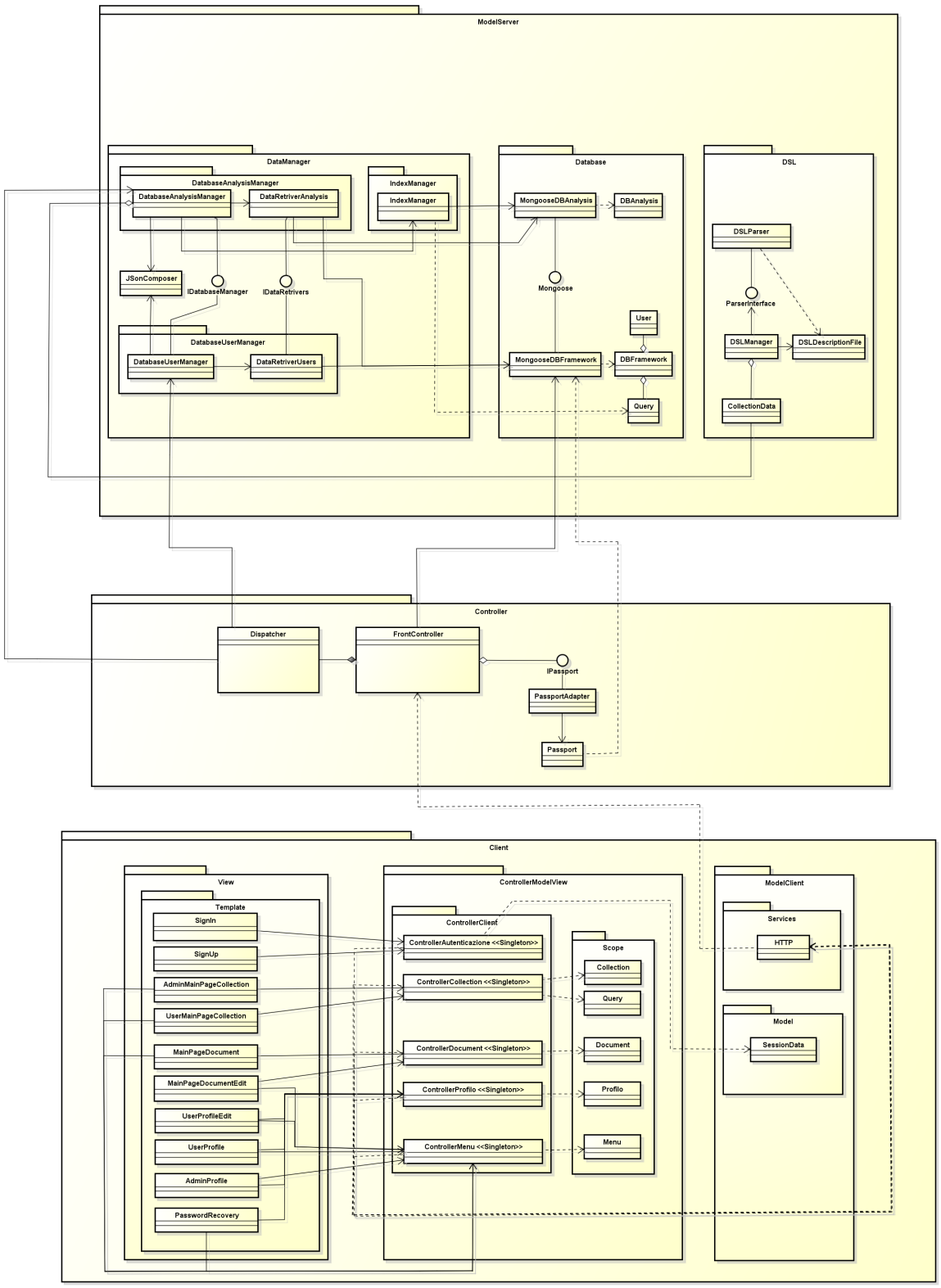

Figura 2: MaaP, Diagramma generale classi

...TODO aggiungere descrizione...

3.2.1 Model

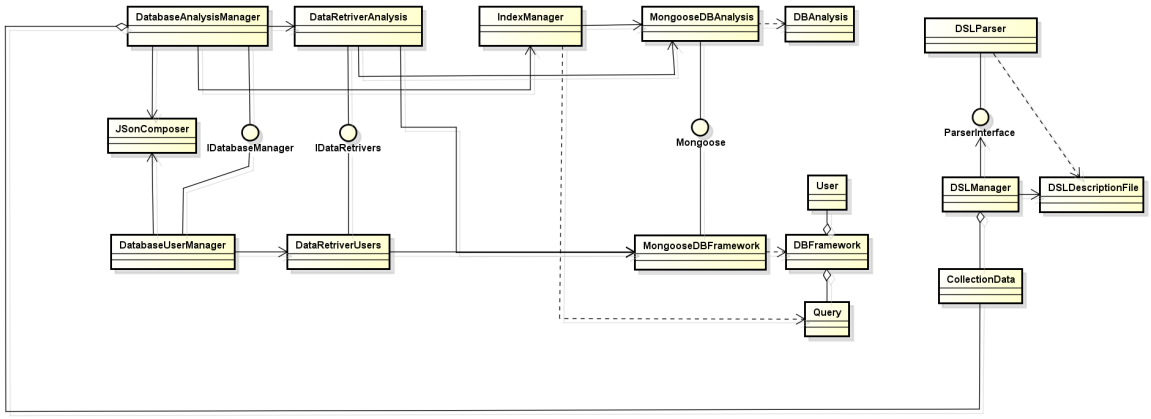

Figura 3: MaaP, Diagramma classi model

Il model contiene:

- Il database di analisi e quello degli utenti
- I file DSL e il parser che li interpreta
- I package dediti al recupero e alla gestione dei dati richiesti dal controller

Tutte le operazioni di gestione, modifica e recupero dei dati vengono messe a disposizione dal model. In tal modo il controller è responsabile solamente di gestire la logica dell'applicazione.

3.2.2 Controller

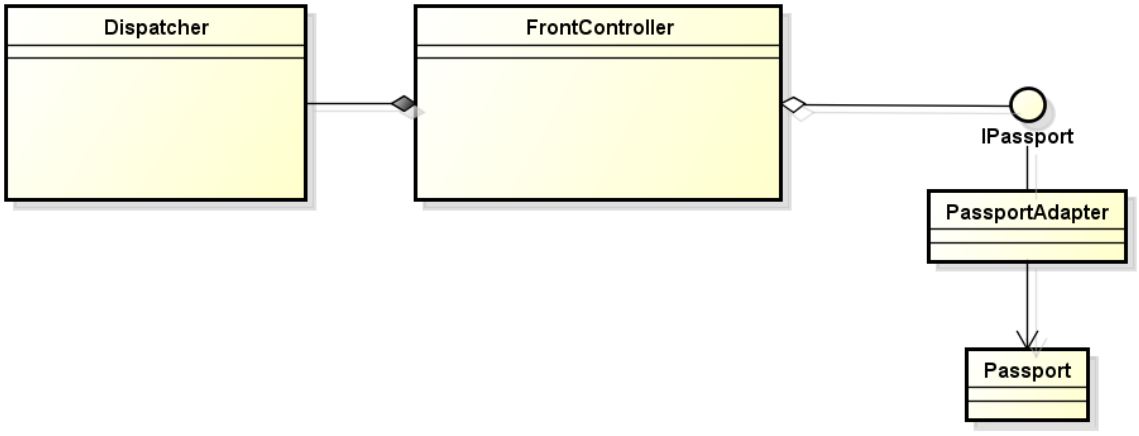

Figura 4: MaaP, Diagramma classi controller

Il controller è responsabile dell'autenticazione delle richieste e del loro routing da client a model e viceversa.

3.2.3 View

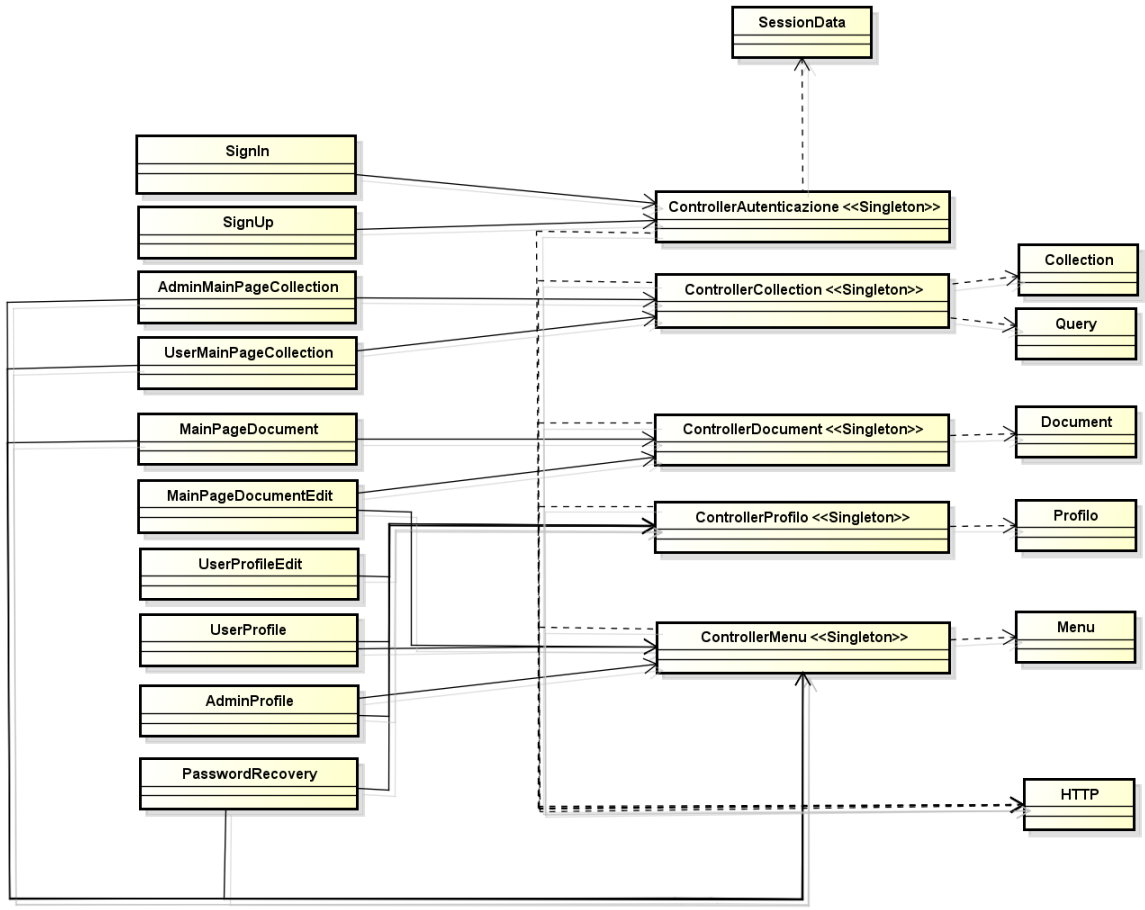

Figura 5: MaaP, Diagramma classi client

La View, essendo gestita da Angular, è a sua volta composta da una View, un Controller ed un Model. Scope e View sono collegati da un "two way data binding", ovvero le modifiche apportate ad uno dei due si riflettono immediatamente anche nell'altro. I dati ricevuti dal server vengono caricati sullo Scope, il quale aggiorna automaticamente la View. Quest'ultima aggiorna lo Scope in caso di modifiche da parte dell'utente finale.

4 Componenti e Classi

4.1 MaaP

4.1.1 Informazioni sul package

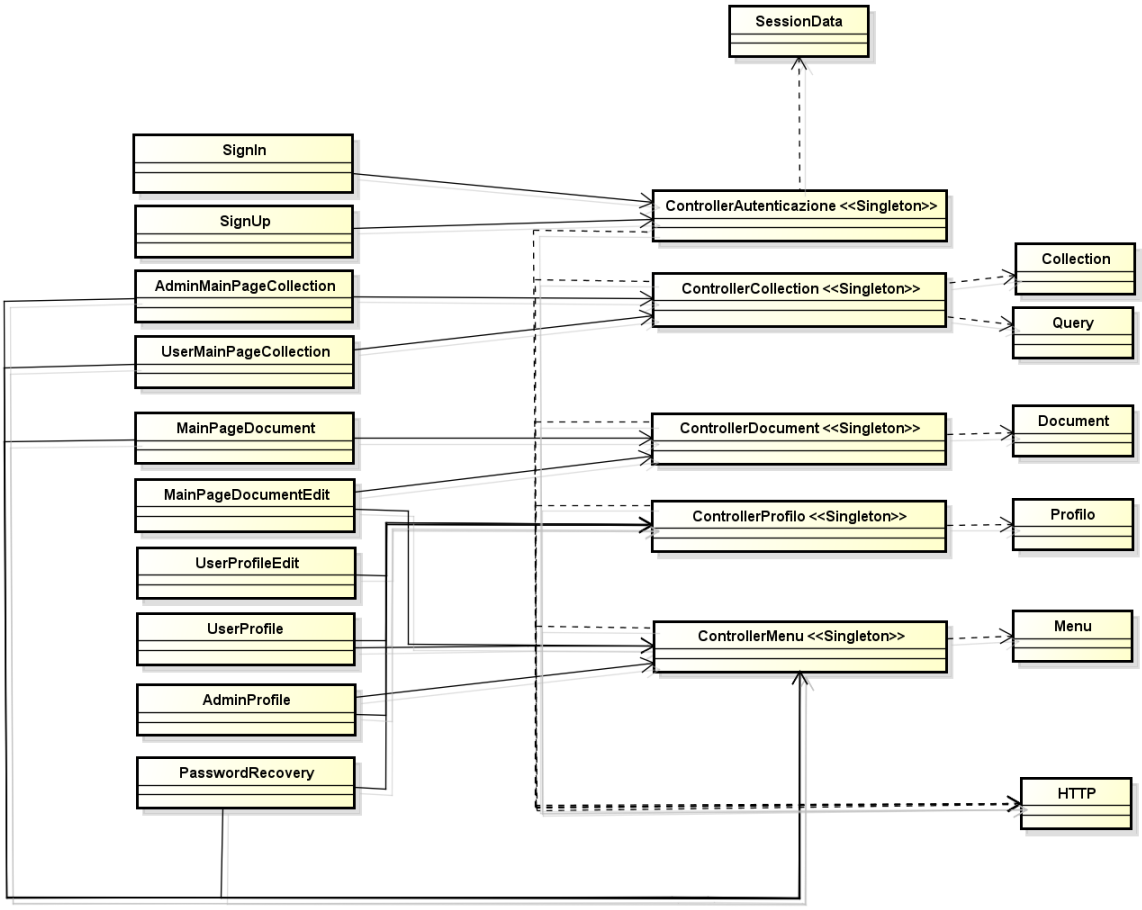

Figura 6: Componente MaaP

4.1.1.1 Descrizione

4.1.1.2 Sottocomponenti

- MaaP::bla1 bla1
- MaaP::bla2 bla2

5 Diagrammi di attività

6 Design Pattern Utilizzati

6.1 Design Pattern architetturali

- MVC

6.2 Design Pattern creazionali

- Singleton

6.3 Design Pattern comportamentali

- Factory

6.4 Design Pattern strutturali

- Facade
- Adapter

7 Stime di fattibilità e di bisogno di risorse

8 Tracciamento

8.1 Tracciamento componenti - requisiti

8.2 Tracciamento requisiti - componenti

A Descrizione Design Pattern

A.1 Design Pattern architetturali

A.2 Design Pattern 1

- **Scopo:**
- **Motivazione:**
- **Applicabilità:**

A.3 Design Pattern creazionali

A.4 Design Pattern 1

- **Scopo:**
- **Motivazione:**
- **Applicabilità:**

A.5 Design Pattern strutturali

A.6 Design Pattern 1

- **Scopo:**
- **Motivazione:**
- **Applicabilità:**

A.7 Design Pattern comportamentali

A.8 Design Pattern 1

- **Scopo:**
- **Motivazione:**
- **Applicabilità:**