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
1.0.1	2014-02-xx	... (AN)	Creazione 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.

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:

- Conoscenza: alcuni componenti del gruppo hanno già avuto esperienze passate con questa tecnologia;
- Novità: tecnologia nuova che si sta diffondendo nel mercato, nata per risolvere i problemi derivanti da database di tipo SQL.

2.2 Javascript

Si è deciso di utilizzare questo linguaggio in quanto è stato fortemente consigliato nel capitolato, i vantaggi offerti sono:

- Interpretazione: il linguaggio viene interpretato e non compilato;
- Compatibilità: tramite questo linguaggio è possibile interagire con alcune tecnologie, quali MongoDB e NodeJS;
- Conoscenza: tutti i membri del gruppo hanno già lavorato con questa tecnologia;
- Esecuzione: il codice JavaScript viene eseguito sul client, e questo sollecita meno i server.

Ha inoltre i seguenti svantaggi:

- Script: gli script hanno capacità limitata, non si può operare sull'hardware solo con il linguaggio JavaScript;
- Visibilità: il codice è visibile e leggibile da chiunque.

2.3 NodeJs

Si è deciso di utilizzare il linguaggio Node.js in quanto è stato richiesto nel capitolato, in oltre offre i seguenti vantaggi:

- Modello event-driven: ovvero programmazione ad eventi, che si basa su un concetto semplice: si lancia un'azione quando accade qualcosa;
- Azioni: ogni azione è asincrona, e grazie a questa caratteristica durante le attese di una certa azione il runtime può gestire dell'altro. Questa tecnologia è utilizzata per realizzare la componente server.

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

2.7 Mongoose

2.8 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, la funzionalità e le relazioni in ingresso ed uscita. Poi si passerà ad illustrare degli esempi d'uso di Design Pattern (descritti approfonditamente nell'Appendice A) e le tecnologie utilizzate.

3.2 Architettura generale

L'architettura del framework segue un modello di architettura in stile three-tier che prevede la suddivisione dell'applicazione in tre diversi strati dedicati rispettivamente all'interfaccia utente (Client), alla business logic (Controller) e alla gestione dei dati persistenti (Model). La parte Client segue il design pattern MVVM utilizzato da AngularJS ed è quindi suddivisa in Model, View, ViewModel.

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...

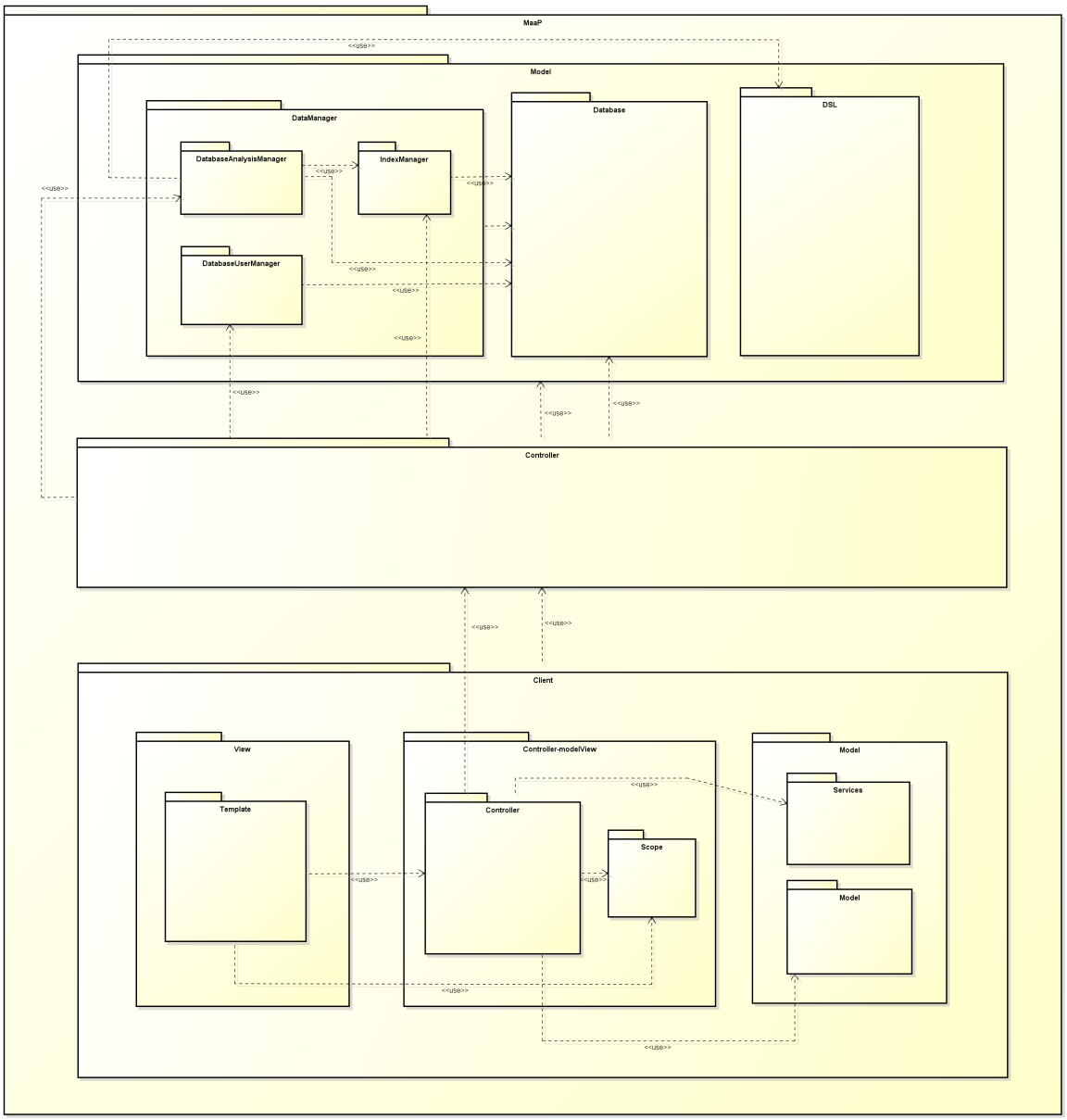

Figura 1: MaaP, Diagramma generale package

...TODO aggiungere descrizione...

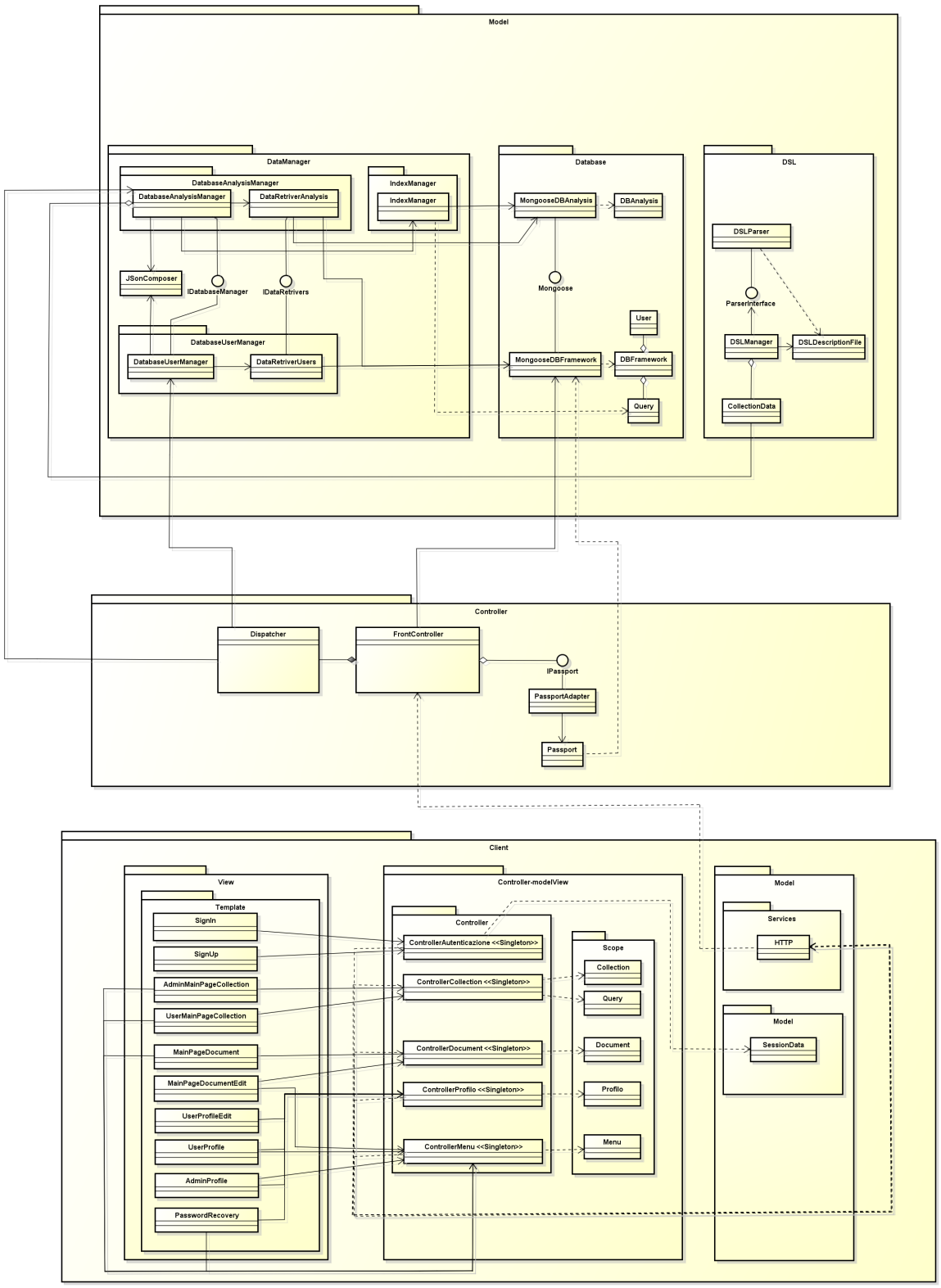

Figura 2: MaaP, Diagramma generale classi

...TODO aggiungere descrizione...

3.2.1 Model

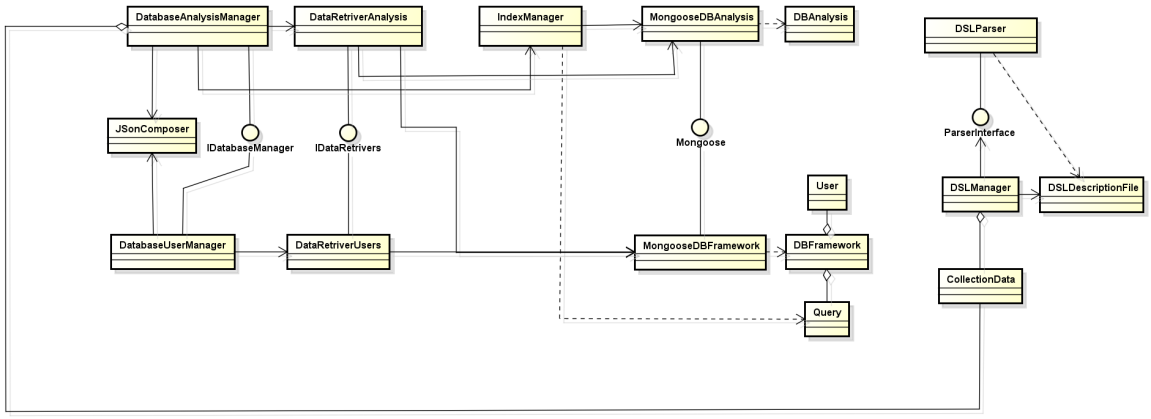

Figura 3: MaaP, Diagramma classi model

...TODO aggiungere descrizione...

3.2.2 Controller

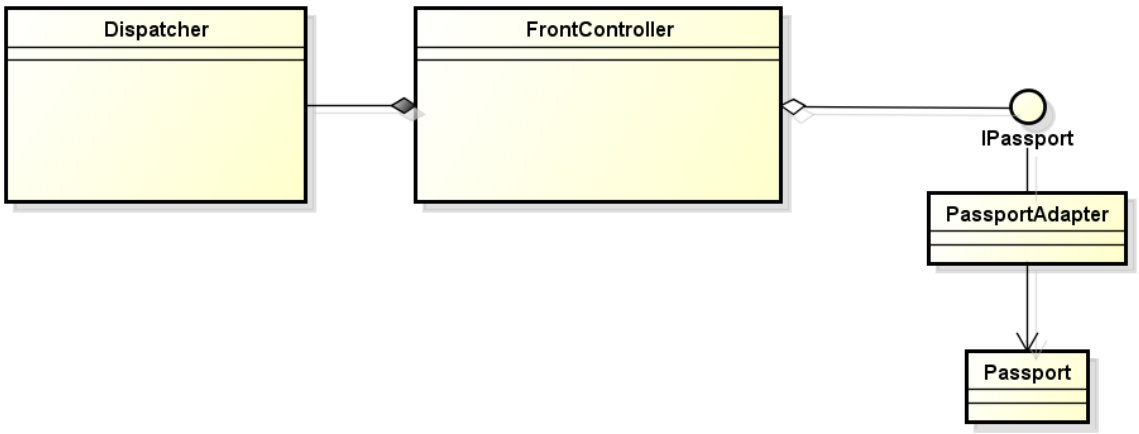

Figura 4: MaaP, Diagramma classi controller

...TODO aggiungere descrizione...

3.2.3 Client

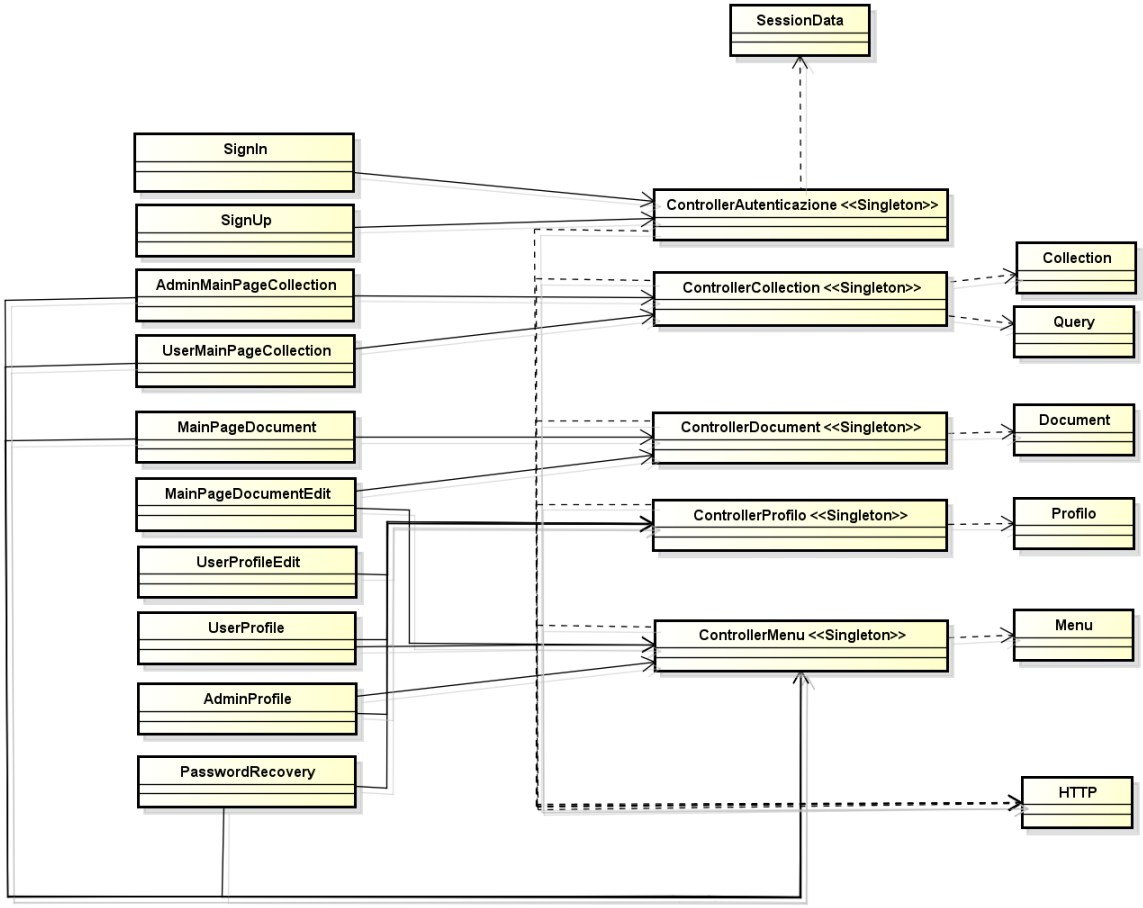

Figura 5: MaaP, Diagramma classi client

...TODO aggiungere descrizione...

4 Componenti e Classi

4.1 MaaP

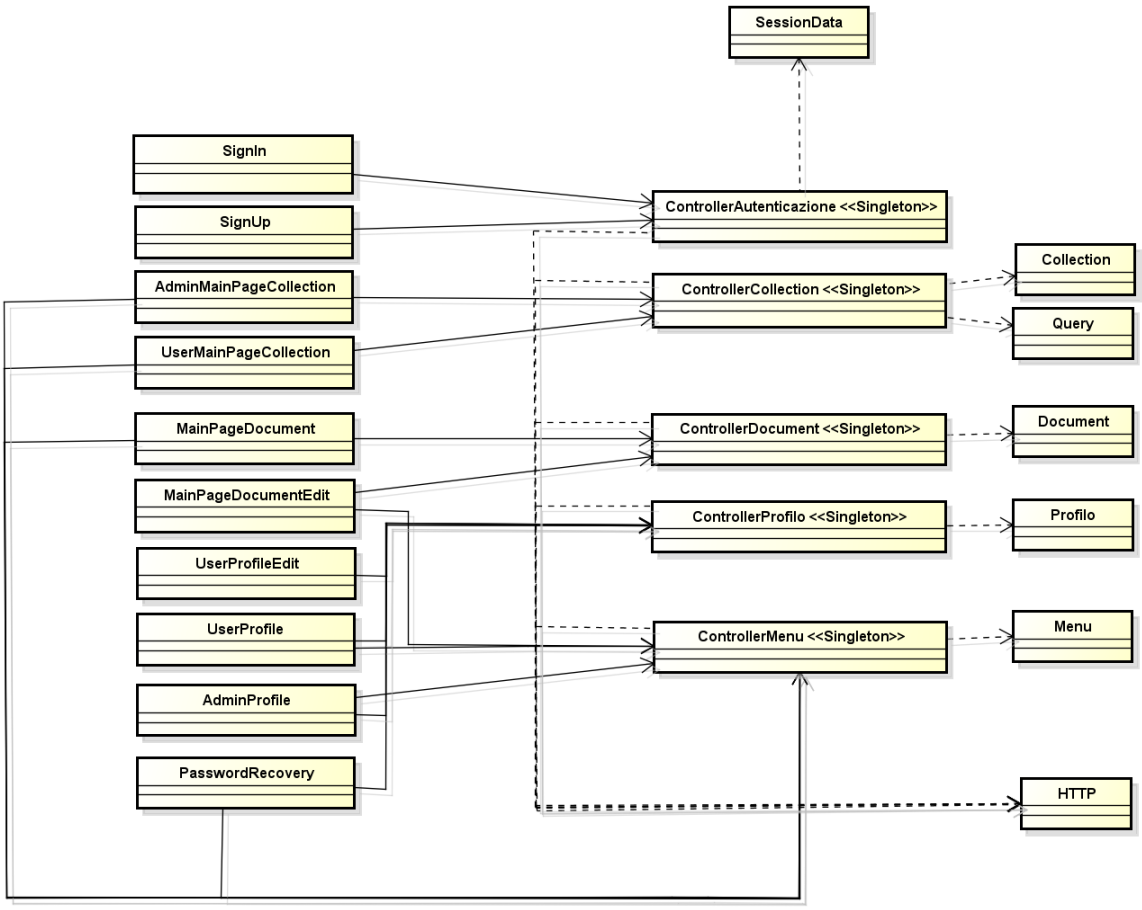

Figura 6: Componente MaaP

4.1.1 Informazioni sul package

4.1.1.1 Descrizione

4.1.1.2 Sottocomponenti

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

5 Diagrammi di attività

6 Design Pattern

6.1 Design Pattern architetturali

6.2 Design Pattern creazionali

6.3 Design Pattern comportamentali

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:**