



MaaP: MongoDB as an admin Platform

Specifica Tecnica

Versione	1.2.0
Data creazione	xxxx-xx-xx2014-01-24
Data ultima modifica	xxxx-xx-xx
Stato del Documento	Formale
Uso del Documento	Esterno
Redazione	Pinato Giacomo,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
1.0.0	2014-03-05	Giacomo Pinato (PR)	Componenti e Classi
1.0.0	2014-01-25	Giacomo Pinato (PR)	Tecnologie Utilizzate
1.0.0	2014-01-25	Fabio Miotto (PR)	Tecnologie Utilizzate
1.0.0	2014-01-24	Giacomo Pinato (PR)	Prima stesura del documento

Tabella 1: Registro delle modifiche

Indice

1	Introduzione	5
1.1	Scopo del documento	5
1.2	Scopo del prodotto	5
1.3	Glossario	5
1.4	Riferimenti	5
1.4.1	Normativi	5
1.4.2	Informativi	5
2	Tecnologie utilizzate	6
2.1	MongoDB	6
2.2	Javascript	6
2.3	NodeJs	6
2.4	jQuery	6
2.5	JSON	7
2.6	AngularJs	7
2.7	HTML5	7
3	Descrizione architettura	8
3.1	Metodo e formalismo di specifica	8
3.2	Architettura generale	8
3.2.1	Model	12
3.2.2	Controller	12
3.2.3	View	13
4	Componenti e Classi	14
4.1	MaaP	14
4.1.1	Informazioni sul package	14
4.2	Server	15
4.2.1	Informazioni sul package	15
4.3	Controller	16
4.3.1	Informazioni sul package	16
4.4	Model	17
4.4.1	Informazioni sul package	17
4.5	DSL	18
4.5.1	Informazioni sul package	18
4.6	Database	19
4.6.1	Informazioni sul package	19
4.7	DataManager	20
4.7.1	Informazioni sul package	20
4.8	DatabaseAnalysisManager	21
4.8.1	Informazioni sul package	21
4.9	DatabaseUserManager	22
4.9.1	Informazioni sul package	22
4.10	IndexManager	23
4.10.1	Informazioni sul package	23
4.11	Client	23
4.11.1	Informazioni sul package	23

4.12 View	24
4.12.1 Informazioni sul package	24
4.13 Template	25
4.13.1 Informazioni sul package	25
4.14 ControllerModelView	26
4.14.1 Informazioni sul package	26
4.15 ControllerClient	27
4.15.1 Informazioni sul package	27
4.16 Scope	28
4.16.1 Informazioni sul package	28
4.17 ModelClient	30
4.17.1 Informazioni sul package	30
4.18 Services	31
4.18.1 Informazioni sul package	31
4.19 Model	32
4.19.1 Informazioni sul package	32
5 Diagrammi di attività	33
6 Design Pattern Utilizzati	34
6.1 Design Pattern architetturali	34
6.2 Design Pattern creazionali	34
6.3 Design Pattern comportamentali	34
6.4 Design Pattern strutturali	34
7 Stime di fattibilità e di bisogno di risorse	35
8 Tracciamento	36
8.1 Tracciamento componenti - requisiti	36
8.2 Tracciamento requisiti - componenti	36
A Descrizione Design Pattern	37
A.1 Design Pattern architetturali	37
A.2 Design Pattern 1	37
A.3 Design Pattern creazionali	37
A.4 Design Pattern 1	37
A.5 Design Pattern strutturali	37
A.6 Design Pattern 1	37
A.7 Design Pattern comportamentali	37
A.8 Design Pattern 1	37

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* 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 verranno 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 può 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 database;

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 più 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 del 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 riflettono 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 browser 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 più grandi vantaggi di questa tecnica è che separa presentazione da implementazione, in quanto i template HTML possono essere modificati senza alterare il modo in cui sono inseriti i dati.
- Dependency Injection: Angular possiede nativamente una "dependency injection", che aiuta gli sviluppatori facilitando la creazione, la comprensione 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...

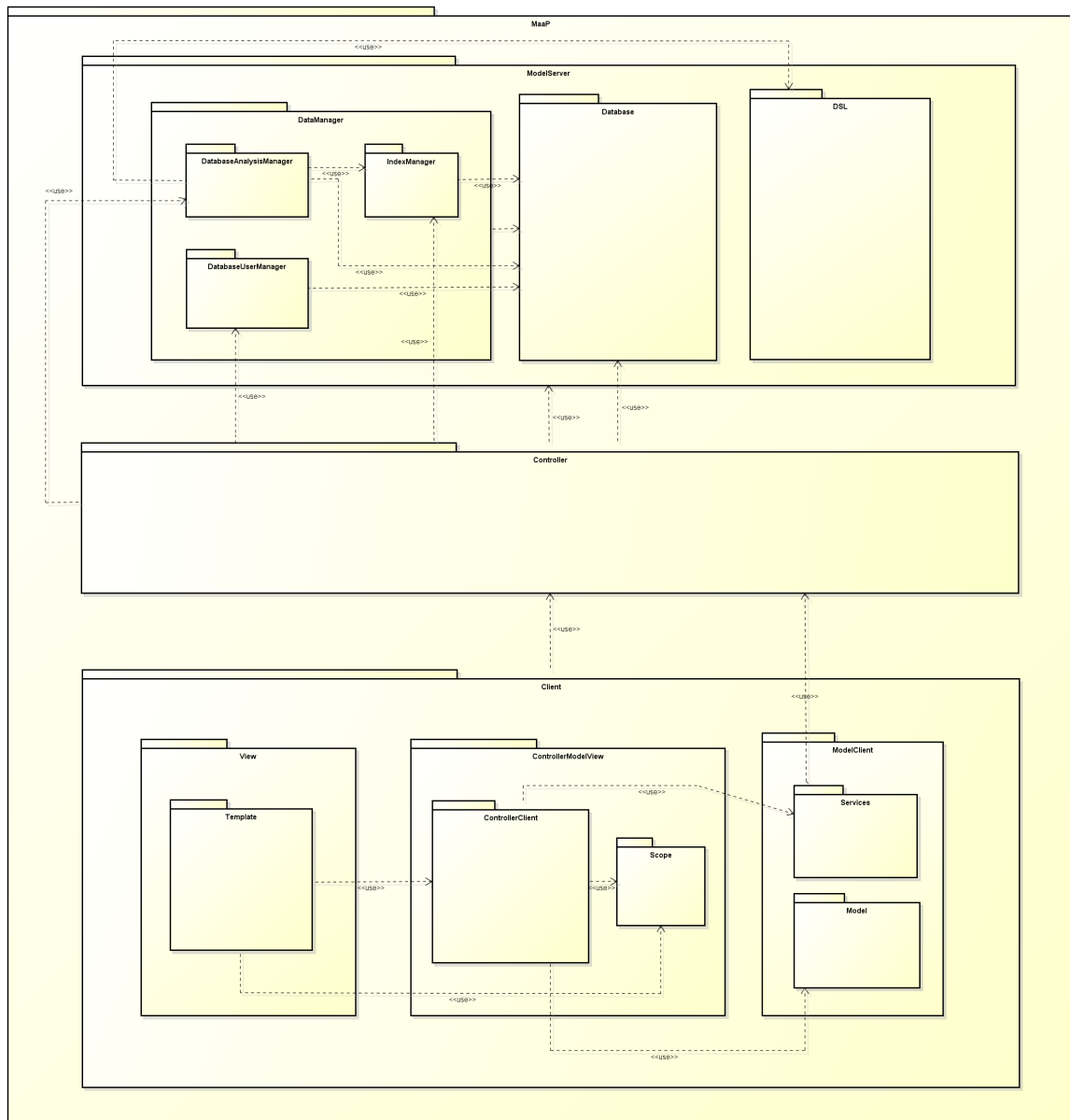


Figura 1: MaaP, Diagramma generale package

...TODO aggiungere descrizione...

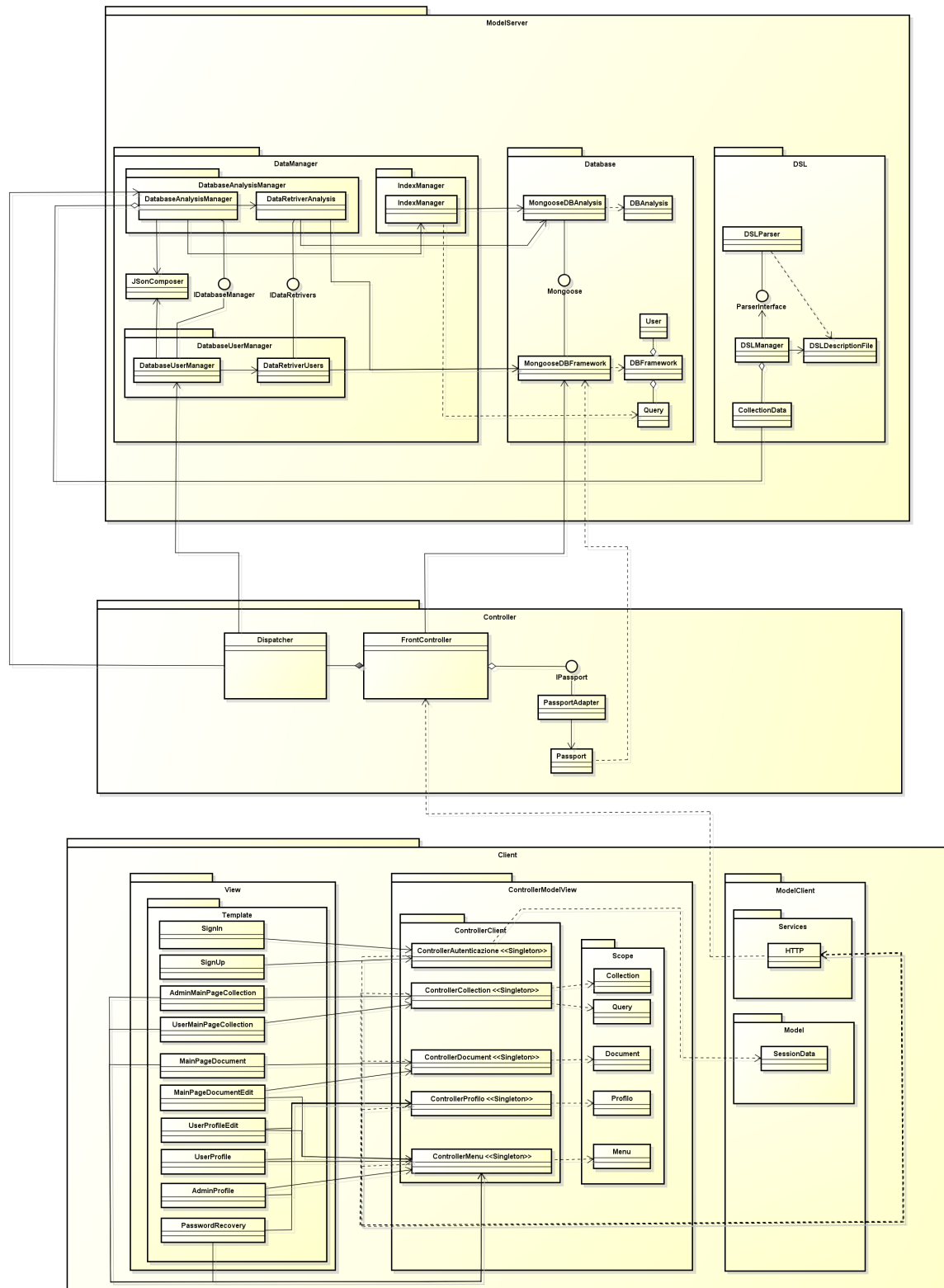


Figura 2: MaaP, Diagramma generale classi

...TODO aggiungere descrizione...

3.2.1 Model

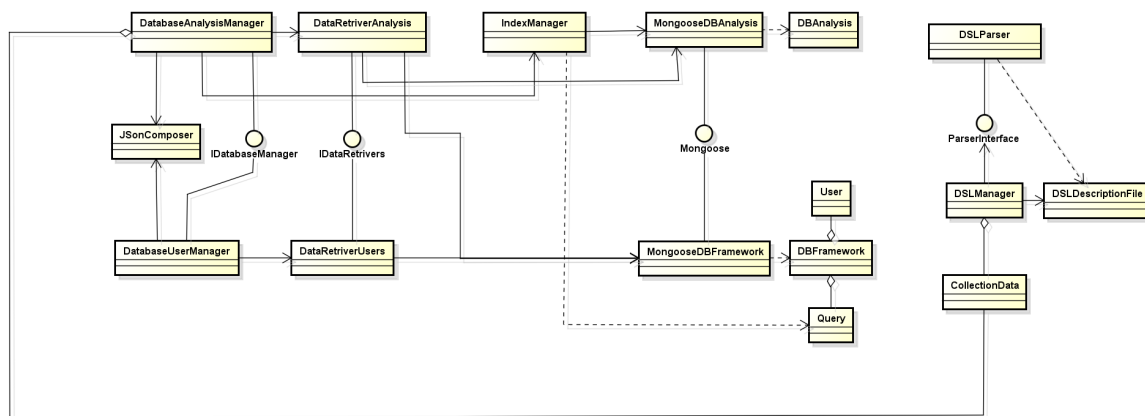


Figura 3: MaaP, Diagramma classi model

Il model contiene:

- Il database di analisi e quello degli utenti
- I file DSL e il parser che li interpreta
- I package dedicati 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

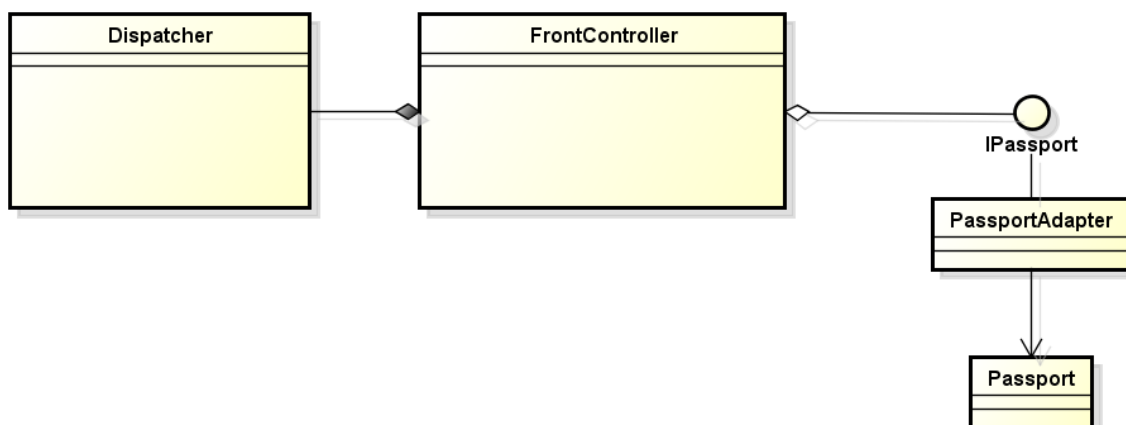


Figura 4: MaaP, Diagramma classi controller

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

3.2.3 View

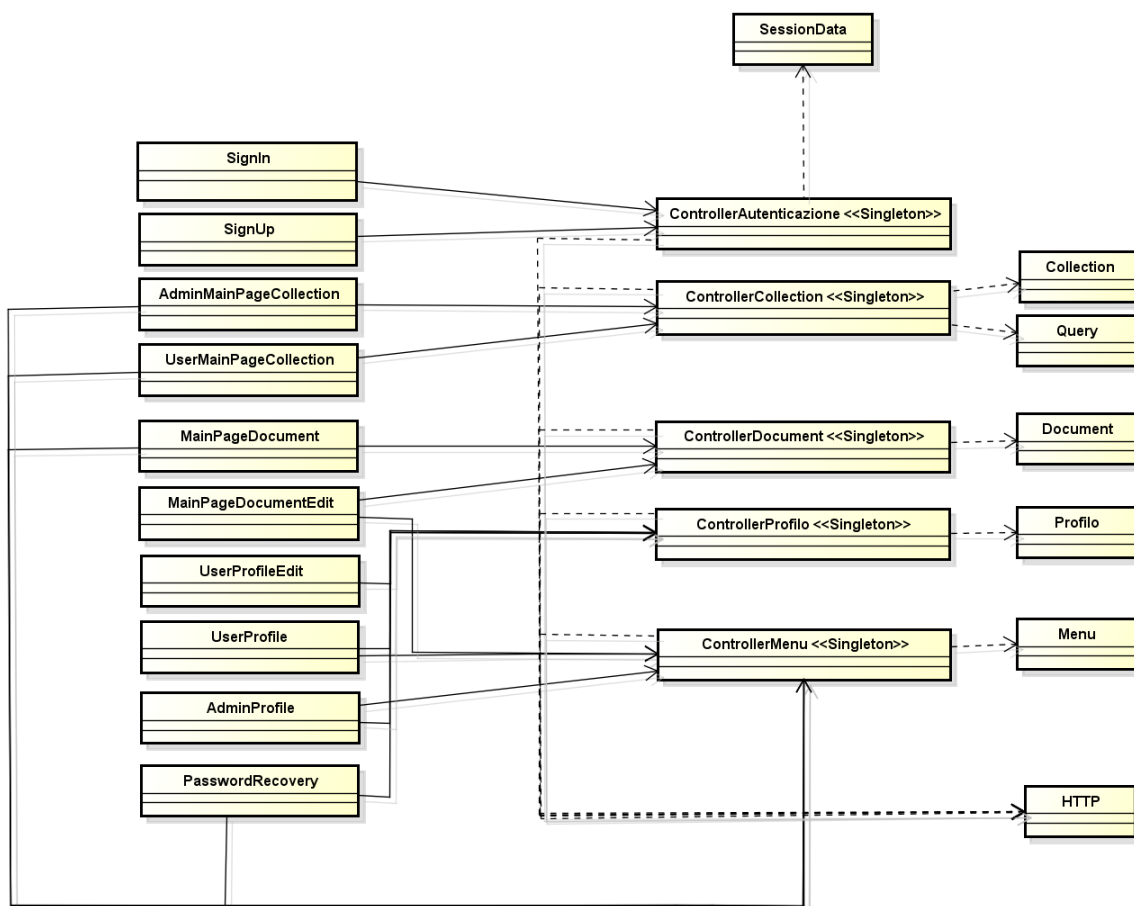


Figura 5: MaaP, Diagramma classi client

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

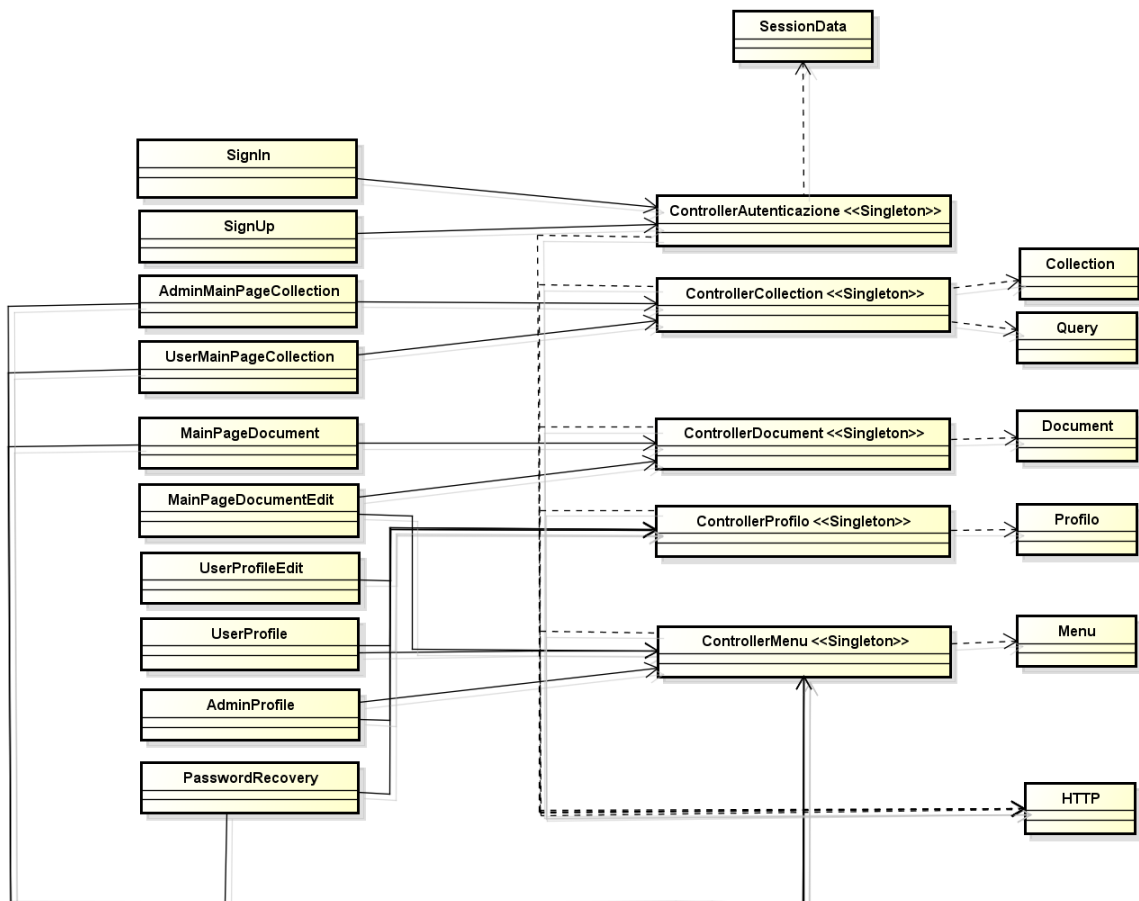


Figura 6: Componente MaaP

4.1.1.1 Descrizione Package (?) globale del progetto.

4.1.1.2 Sottocomponenti

- MaaP::Server
- MaaP::Client

4.2 Server

4.2.1 Informazioni sul package

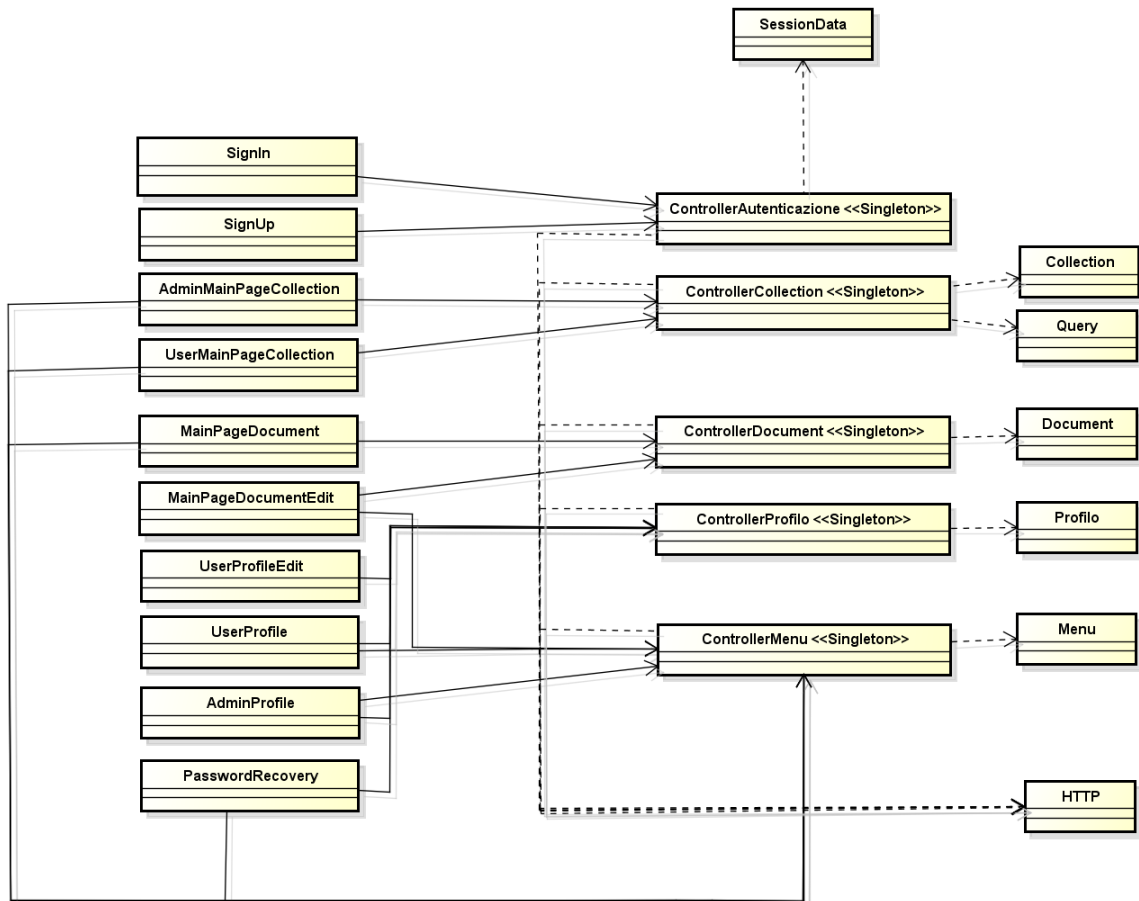


Figura 7: Componente MaaP

4.2.1.1 Descrizione Package contenente la componente server-side del progetto. Contiene il model ed il controller.

4.2.1.2 Sottocomponenti

- MaaP::Server::Controller
- MaaP::Server::Model

4.3 Controller

4.3.1 Informazioni sul package

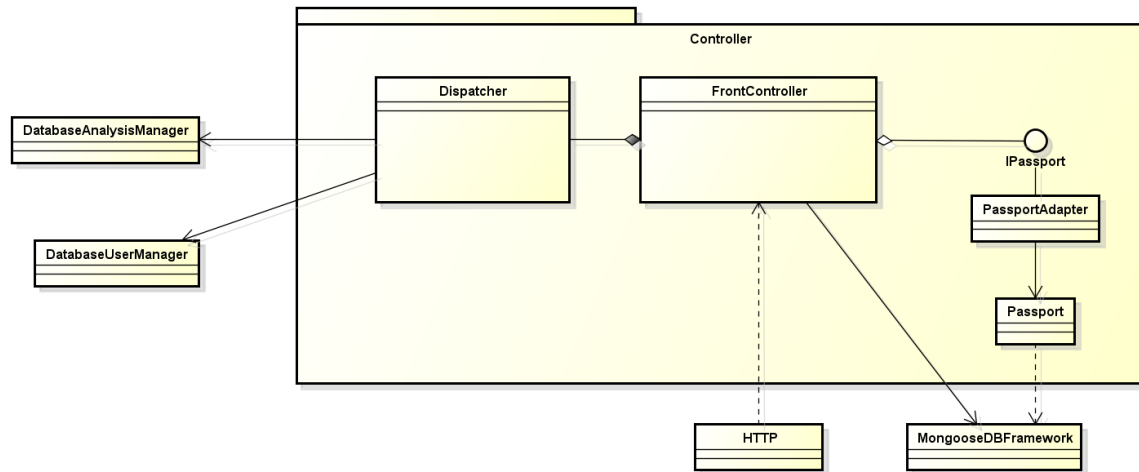


Figura 8: Componenti Controller

4.3.1.1 Descrizione Package contenente il controller dell'applicazione MaaP. Il controller si occupa del routing delle richieste provenienti da e dirette al client, oltre alla loro autenticazione.

4.3.1.2 Classi

- MaaP::Server::Controller::Dispatcher
- MaaP::Server::Controller::FrontController
- MaaP::Server::Controller::PassportAdapter
- MaaP::Server::Controller::Passport (?)

4.3.1.3 Iterazioni con altri componenti

- MaaP::Server::Model
- MaaP::Client::Controller

4.4 Model

4.4.1 Informazioni sul package

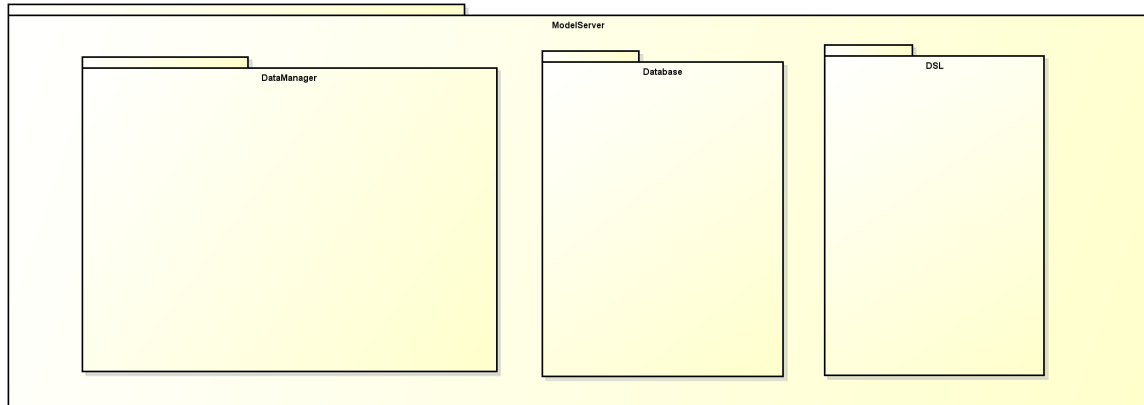


Figura 9: Componenti Model

4.4.1.1 Descrizione Package del model server-side. Il model contiene i database di analisi e degli utenti, il parser e i file di descrizione DSL e ulteriori file di configurazione, oltre a tutta la logica di recupero e gestione dei dati dei database.

4.4.1.2 Sottocomponenti

- MaaP::Server::Model::DataManager
- MaaP::Server::Model::Database
- MaaP::Server::Model::DSL

4.5 DSL

4.5.1 Informazioni sul package

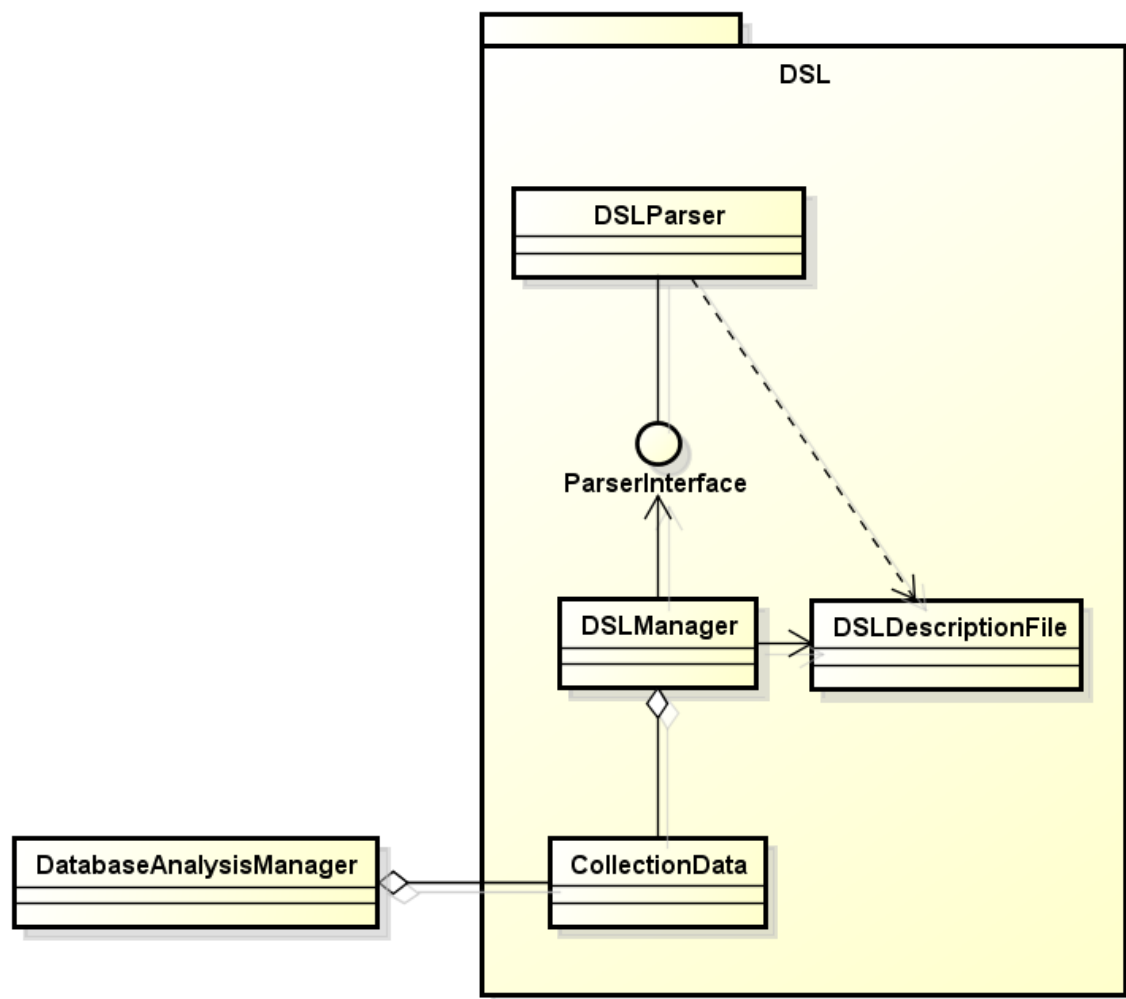


Figura 10: Componenti DSL

4.5.1.1 Descrizione Package della gestione DSL. Contiene i file di descrizione per le collection e il parser per generare la loro struttura partendo dagli stessi.

4.5.1.2 Classi

- MaaP::Server::Model::DSL::DSLParser
- MaaP::Server::Model::DSL::DSLManager

- MaaP::Server::Model::DSL::CollectionData
- MaaP::Server::Model::DSL::DSLDescriptionFile
- MaaP::Server::Model::DSL::ParserInterface (?)

4.6 Database

4.6.1 Informazioni sul package

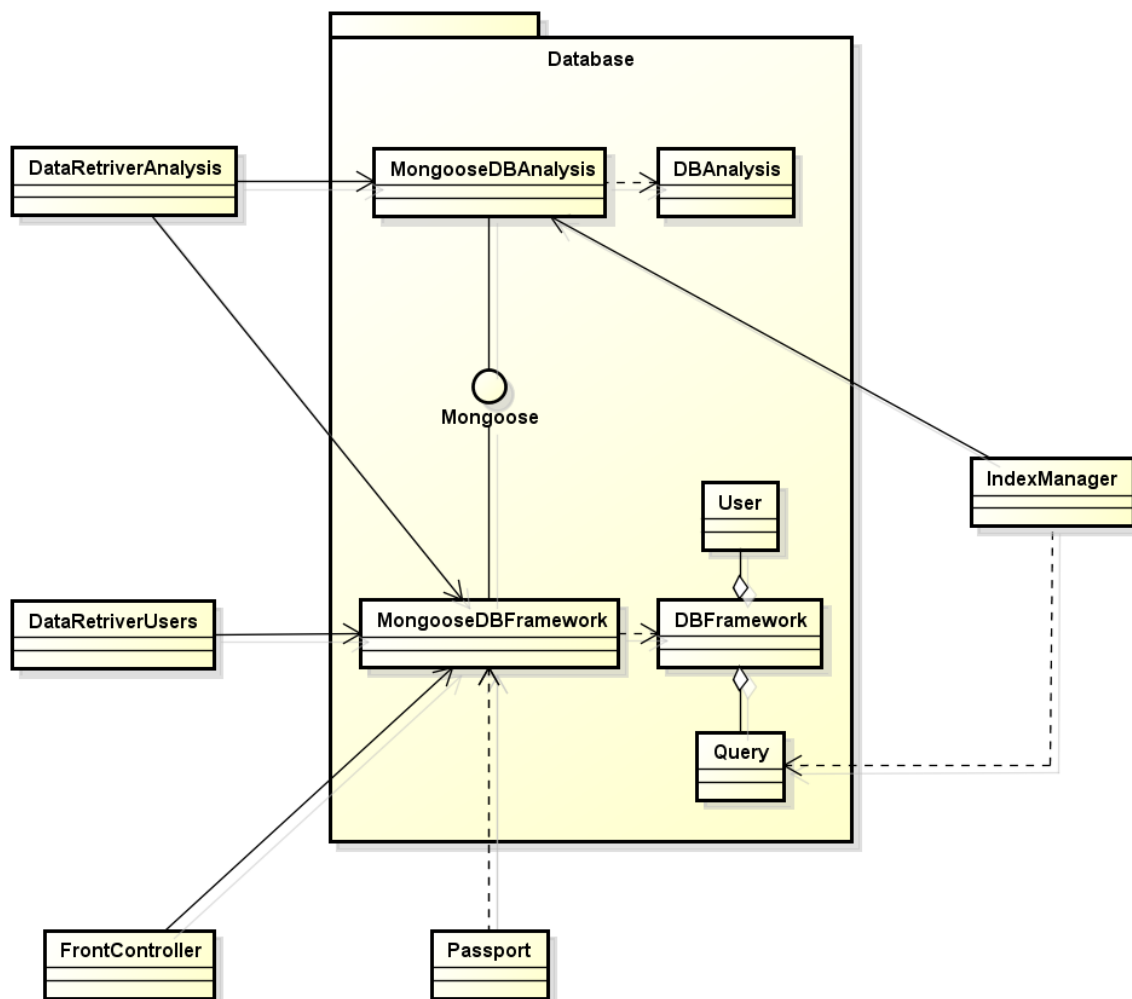


Figura 11: Componenti Database

4.6.1.1 Descrizione Package dei database presenti nel model. Contiene il database degli utenti, il database di analisi e la raccolta delle query effettuate più frequentemente.ù

4.6.1.2 Classi

- MaaP::Server::Model::Database::MongooseDBAnalysis
- MaaP::Server::Model::Database::DBAnalysis
- MaaP::Server::Model::Database::Mongoose (?)
- MaaP::Server::Model::Database::MongooseDBFramework
- MaaP::Server::Model::Database::DBFramework
- MaaP::Server::Model::Database::USer
- MaaP::Server::Model::Database::Query

4.7 DataManager

4.7.1 Informazioni sul package

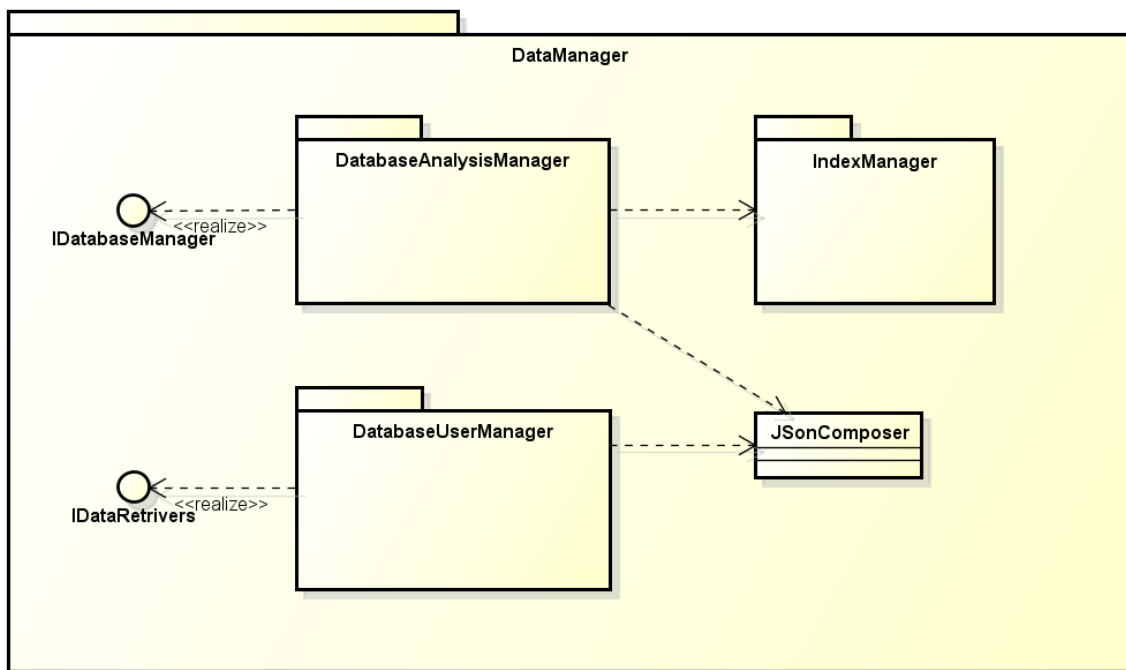


Figura 12: Componenti DataManager

4.7.1.1 Descrizione Package del DataManager. Contiene la logica per la gestione dei database degli utenti e di analisi, per la gestione degli indici e per la composizione dei JSON.

4.7.1.2 Sottocomponenti

- MaaP::Server::Model::DataManager::DatabaseAnalysisManager
- MaaP::Server::Model::DataManager::DatabaseUserManager
- MaaP::Server::Model::DataManager::IndexManager

4.7.1.3 Classi

- MaaP::Server::Model::DataManager::JSoNComposer
- MaaP::Server::Model::DataManager::IDatabaseManager
- MaaP::Server::Model::DataManager::IDataRetrievers

4.8 DatabaseAnalysisManager

4.8.1 Informazioni sul package

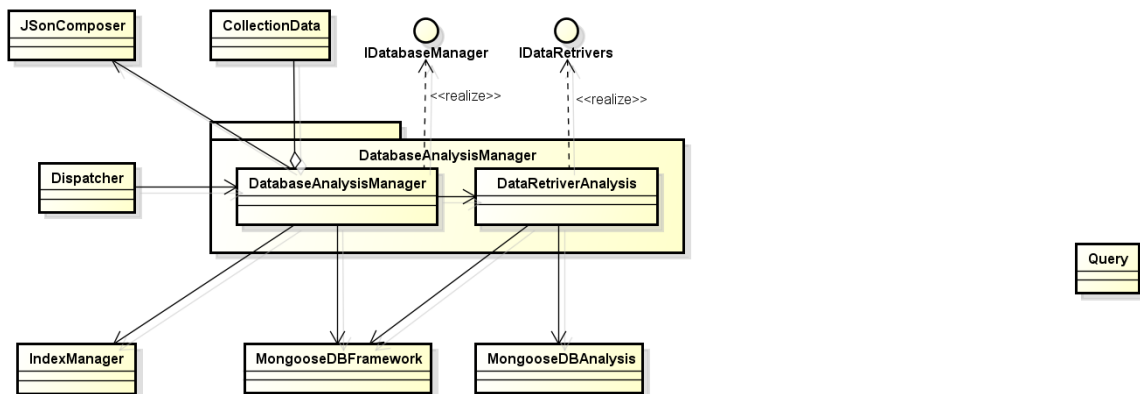


Figura 13: Componenti DatabaseAnalysisManager

4.8.1.1 Descrizione Package del gestore del database di analisi.

4.8.1.2 Classi

- MaaP::Server::Model::DataManager::DatabaseAnalysisManager::DatabaseAnalysisManager
- MaaP::Server::Model::DataManager::DatabaseAnalysisManager::DataRetrieverAnalysis

4.9 DatabaseUserManager

4.9.1 Informazioni sul package

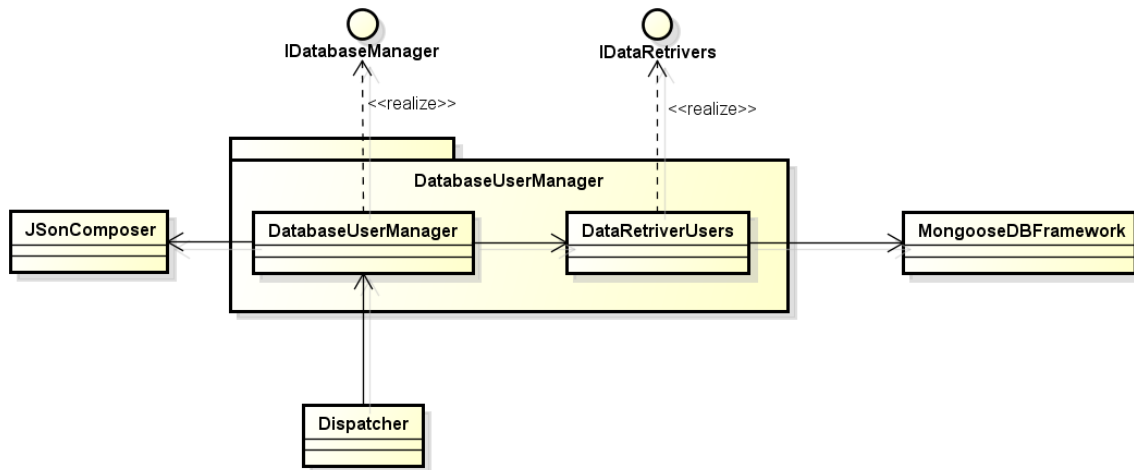


Figura 14: Componenti DatabaseUserManager

4.9.1.1 Descrizione Package del gestore del database degli utenti.

4.9.1.2 Classi

- MaaP::Server::Model::DataManager::DatabaseUserManager::DatabaseUserManager
- MaaP::Server::Model::DataManager::DatabaseUserManager::DataRetrieverUser

4.10 IndexManager

4.10.1 Informazioni sul package

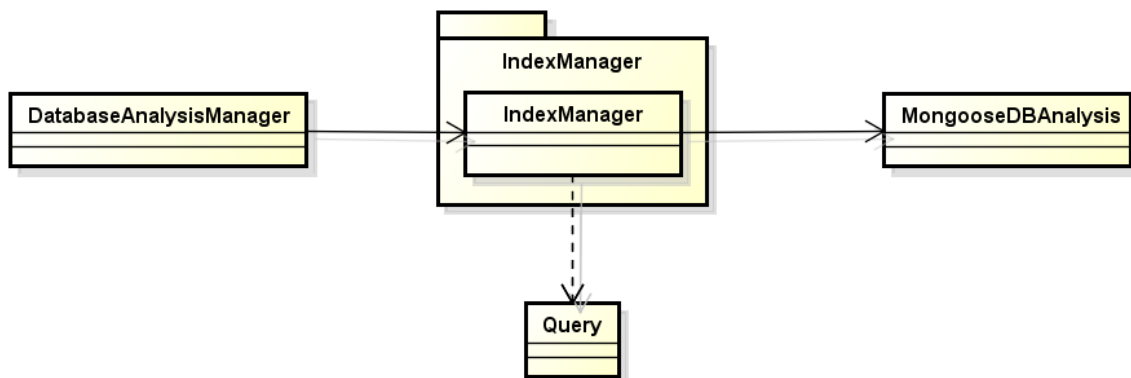


Figura 15: Componenti IndexManager

4.10.1.1 Descrizione Package del gestore degli indici.

4.10.1.2 Classi

- MaaP::Server::Model::DataManager::IndexManager::IndexManager

4.11 Client

4.11.1 Informazioni sul package

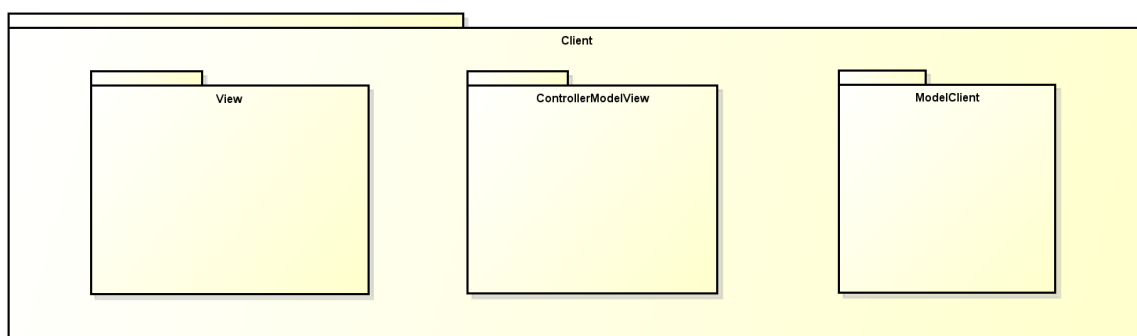


Figura 16: Componenti Client

4.11.1.1 Descrizione Package contenente la componente client-side del progetto.

4.11.1.2 Sottocomponenti

- MaaP::Client::View
- MaaP::Client::ControllerModelView
- MaaP::Client::ModelClient

4.12 View

4.12.1 Informazioni sul package

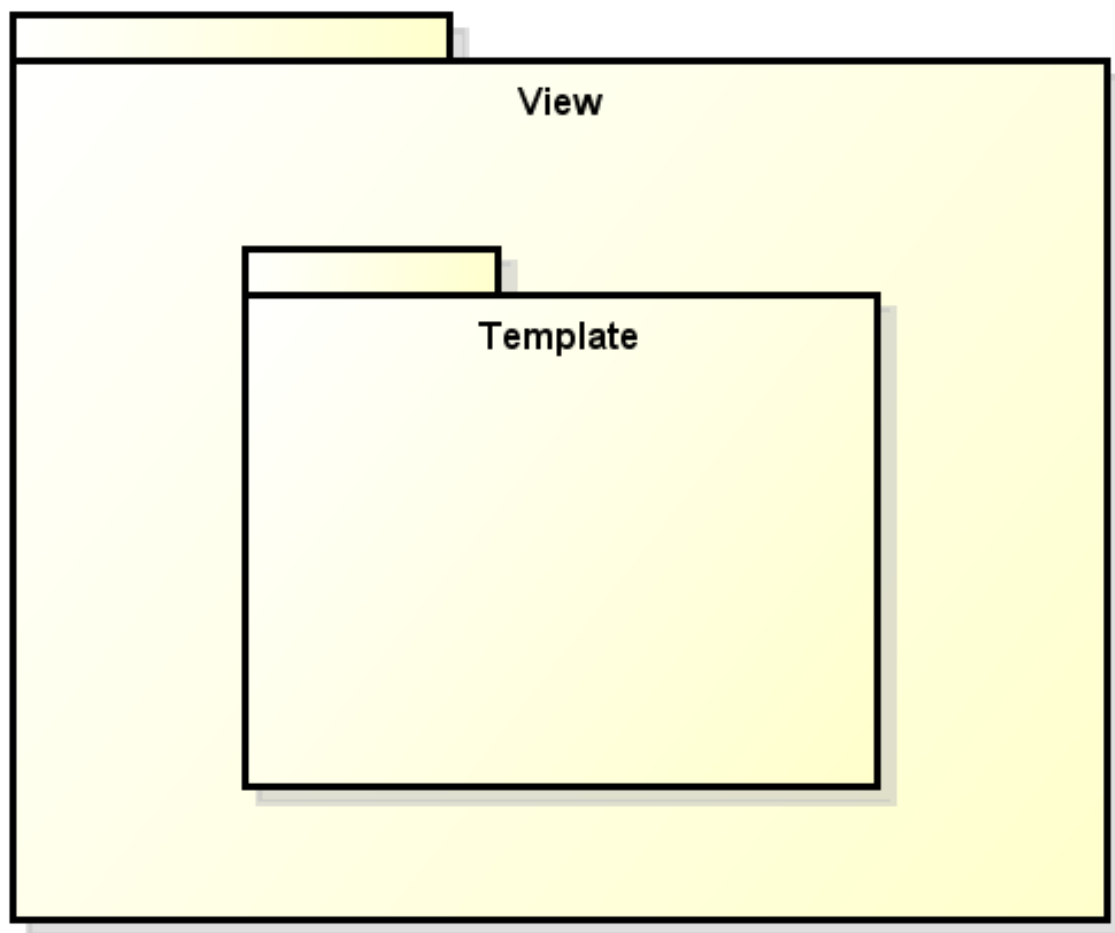


Figura 17: Componenti View

4.12.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.12.1.2 Sottocomponenti

- MaaP::Client::View::Template

4.13 Template

4.13.1 Informazioni sul package

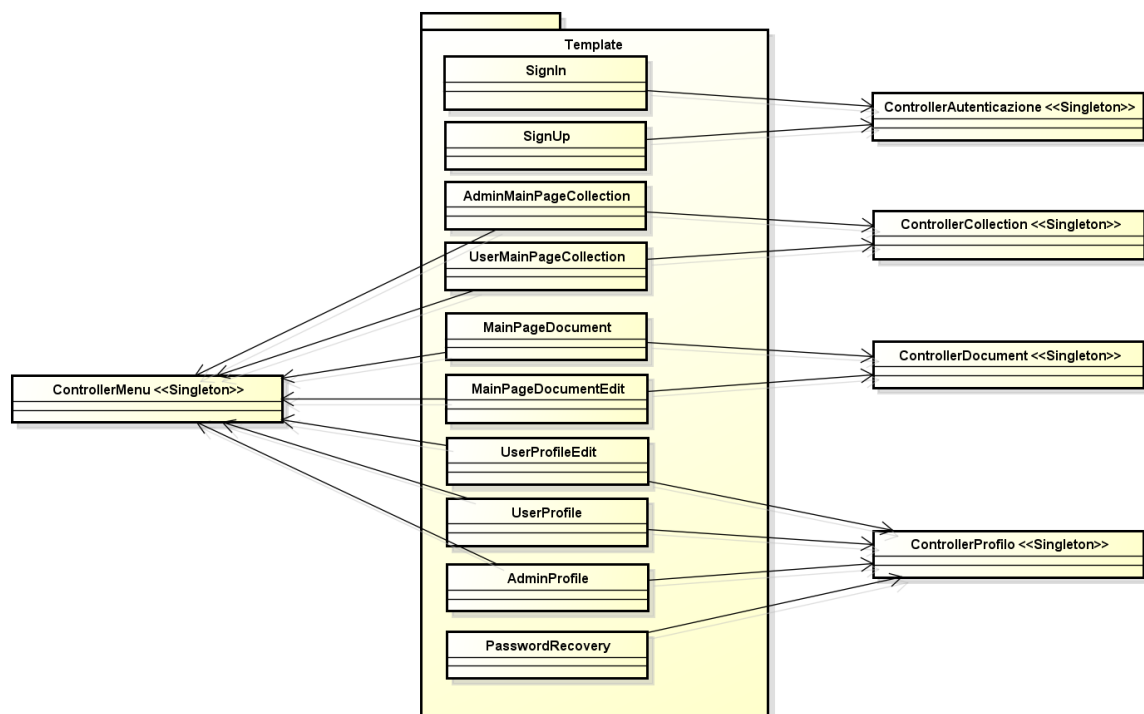


Figura 18: Componenti Template

4.13.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.13.1.2 Classi

- MaaP::Client::View::Template::SignIn
- MaaP::Client::View::Template::SignUp
- MaaP::Client::View::Template::AdminMainPageCollection
- MaaP::Client::View::Template::UserMainPageCollection
- MaaP::Client::View::Template::MainPageDocument
- MaaP::Client::View::Template::MainPageDocumentEdit

- MaaP::Client::View::Template::UserProfileEdit
- MaaP::Client::View::Template::UserProfile
- MaaP::Client::View::Template::AdminProfile
- MaaP::Client::View::Template::PasswordRecovery

4.14 ControllerModelView

4.14.1 Informazioni sul package

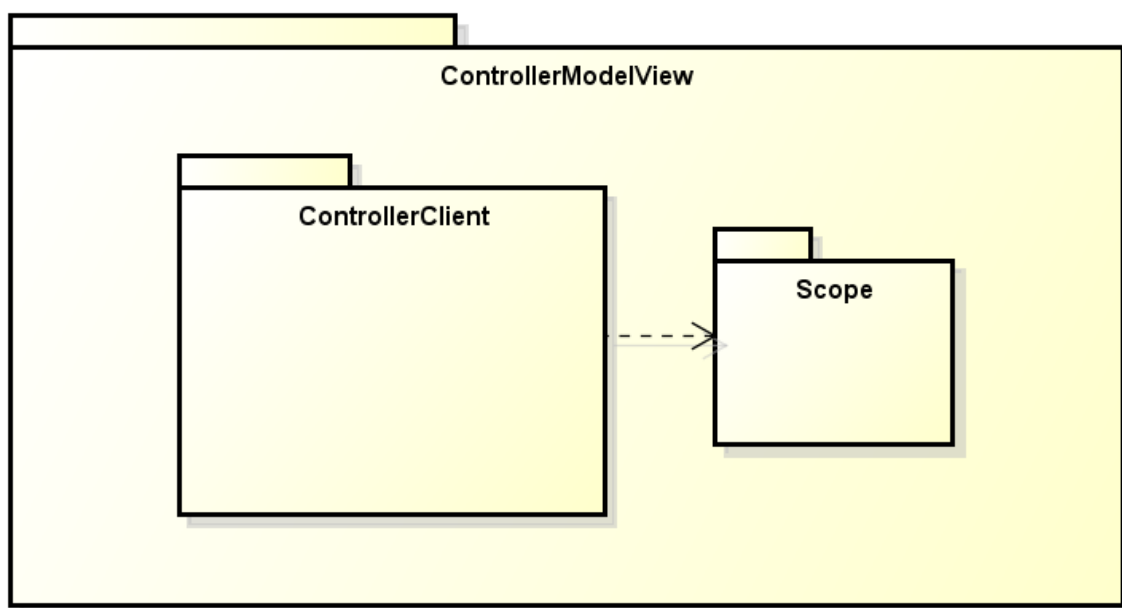


Figura 19: Componenti ControllerModelView

4.14.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.14.1.2 Sottocomponenti

- MaaP::Client::ControllerModelView::ControllerClient
- MaaP::Client::ControllerModelView::Scope

4.15 ControllerClient

4.15.1 Informazioni sul package

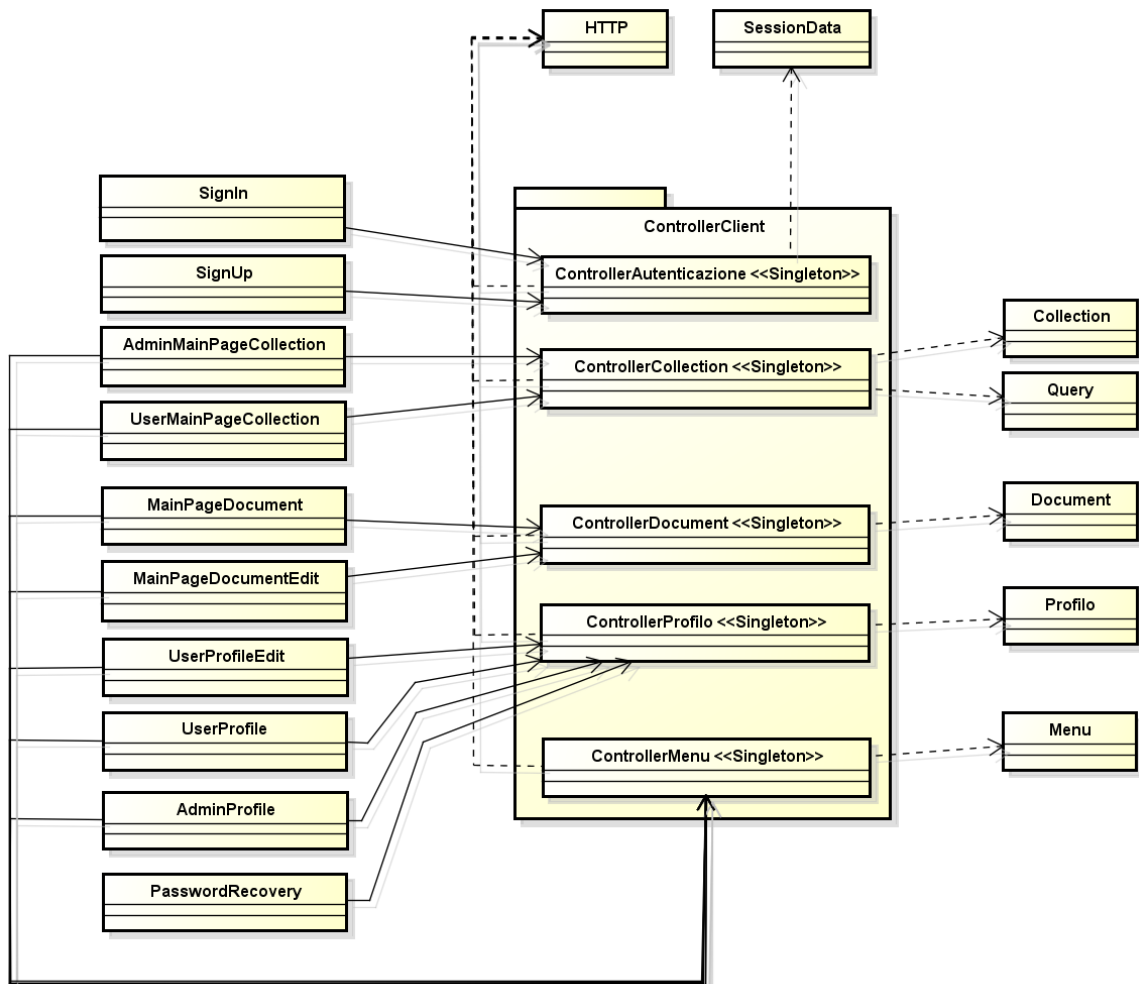


Figura 20: Componenti ControllerClient

4.15.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.15.1.2 Classi

- MaaP::Client::ControllerModelView::ControllerClient::ControllerAutenticazione
- MaaP::Client::ControllerModelView::ControllerClient::ControllerCollection

- MaaP::Client::ControllerModelView::ControllerClient::ControllerDocument
- MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo
- MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu

4.16 Scope

4.16.1 Informazioni sul package

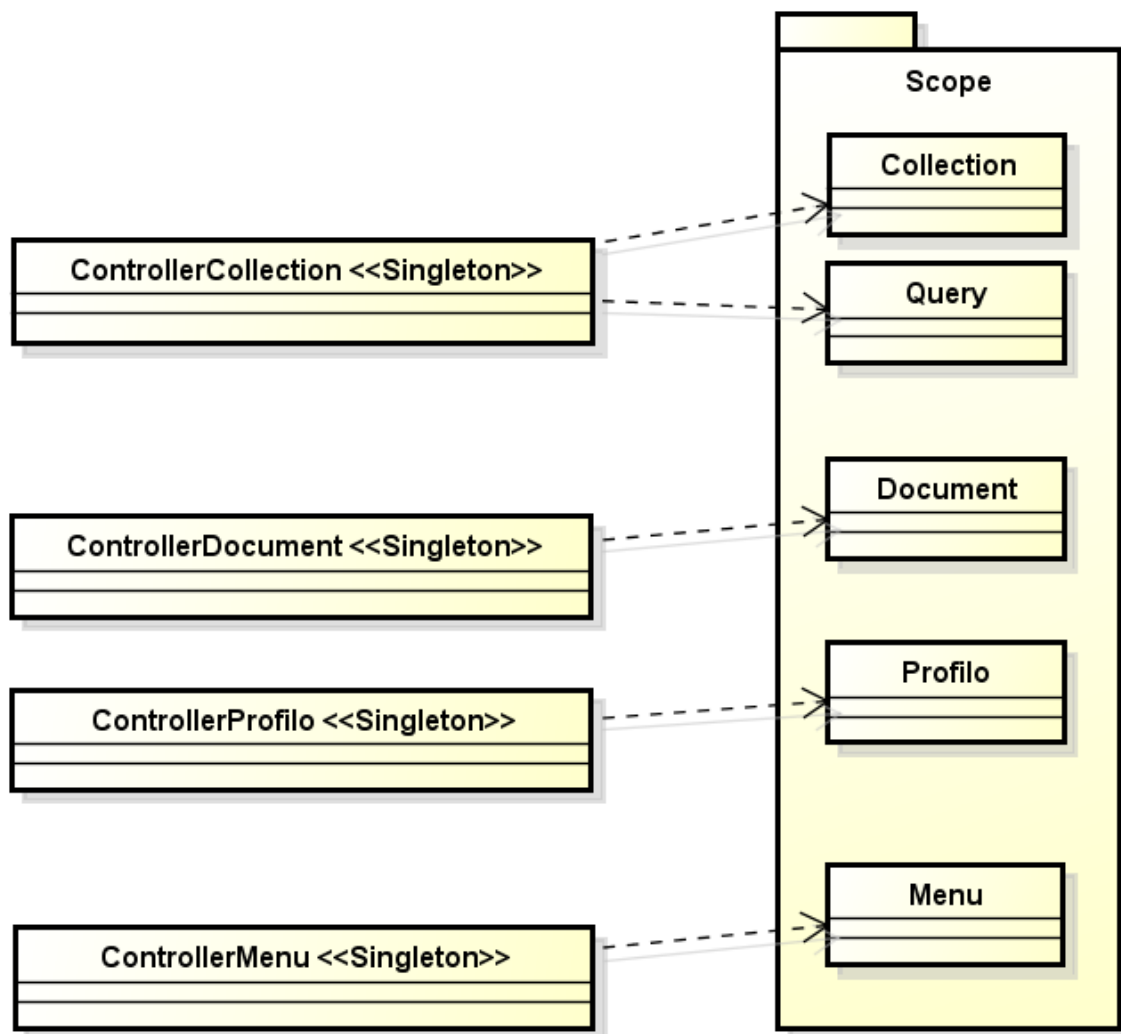


Figura 21: Componenti Scope

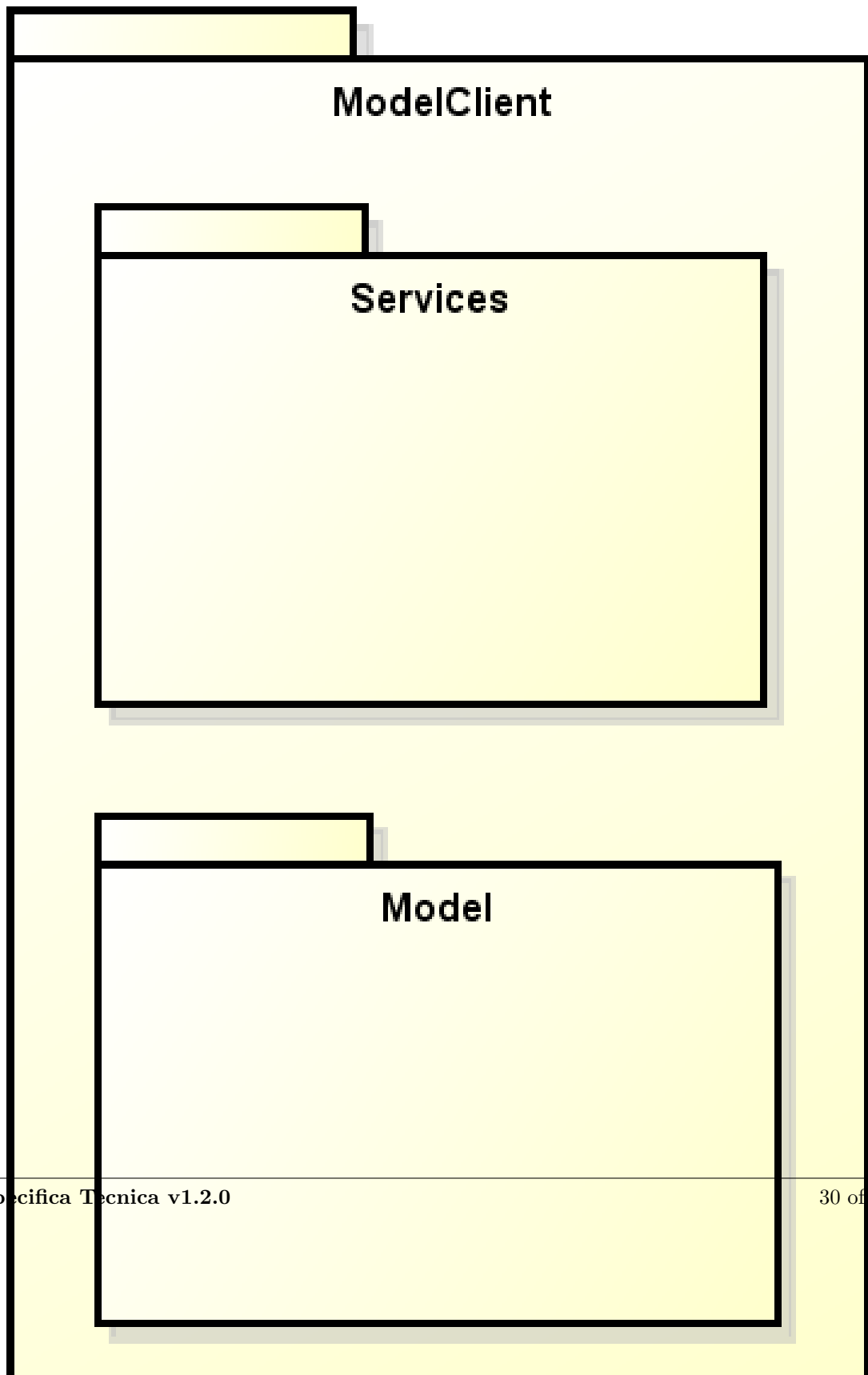
4.16.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.16.1.2 Classi

- MaaP::Client::ControllerModelView::Scope::Collection
- MaaP::Client::ControllerModelView::Scope::Query
- MaaP::Client::ControllerModelView::Scope::Document
- MaaP::Client::ControllerModelView::Scope::Profilo
- MaaP::Client::ControllerModelView::Scope::Menu

4.17 ModelClient

4.17.1 Informazioni sul package



4.17.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.17.1.2 Sottocomponenti

- MaaP::Client::ModelClient::Model
- MaaP::Client::ModelClient::Services

4.18 Services

4.18.1 Informazioni sul package

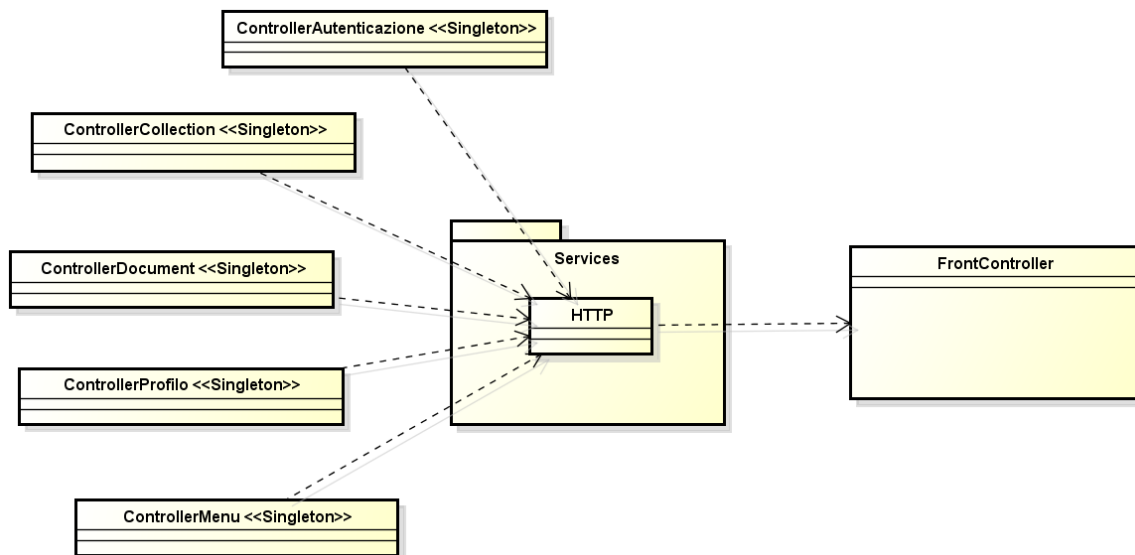


Figura 23: Componenti Services

4.18.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.18.1.2 Classi

- MaaP::Client::ModelClient::Services::HTTP

4.19 Model

4.19.1 Informazioni sul package

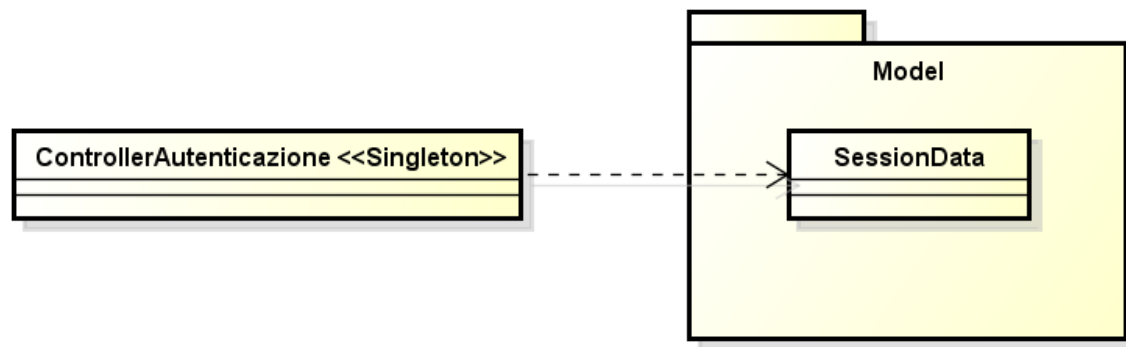


Figura 24: Componenti Model

4.19.1.1 Descrizione Package contenente la componente i template delle pagine html generate dal framework.

4.19.1.2 Classi

- MaaP::Client::ModelClient::Model::SessionData

5 Diagrammi di attività

6 Design Pattern Utilizzati

6.1 Design Pattern architetturali

- MVC: Utilizzato per l'architettura generale dell'applicazione.

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:**