MaaP: MongoDB as an admin Platform

Specifica Tecnica

Versione	3.2.0
Data creazione	2014-01-24
Data ultima modifica	2014-03-xx
Stato del Documento	Formale
Uso del Documento	Esterno
Redazione	Pinato Giacomo,nome2,...
Verifica	nome1,nome2,...
Approvazione	nome1,nome2,...
Distribuzione	Aperture Software
	Prof. Vardanega Tullio
	Prof. Cardin Riccardo

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

Elenco delle figure

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_v3.2.0.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_v3.2.0.pdf
- Norme di Progetto: Norme_di_Progetto_v3.2.0.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, AngularJS 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 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 $parser_G$. 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.

2.5 AngularJs

- *Two Way Data-Binding*$_G$: 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 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 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.

2.6 HTML5

L'*HTML5*$_G$ è un linguaggio di markup per la strutturazione delle pagine web.
Nel progetto MaaP è stato scelto di utilizzare l'HTML5 perché introduce novità finalizzate soprattutto a migliorare il *disaccoppiamento*$_G$ tra struttura, definita dal markup, caratteristiche di resa (tipo di carattere, colori, eccetera), definite dalle direttive di stile, e contenuti di una pagina web, definiti dal testo vero e proprio.
Inoltre l'HTML5 prevede il supporto per la memorizzazione locale di grosse quantità di dati scaricati dal web browser, ideale per consentire l'utilizzo di applicazioni web e quindi per il framework MaaP.

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.

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.

I seguenti diagrammi rappresentano l'architettura ad alto livello del framework, indicando i package e le relazioni che intercorrono tra questi.

3.2.1 MaaPCLI

3.2.1.1 Informazioni sul package

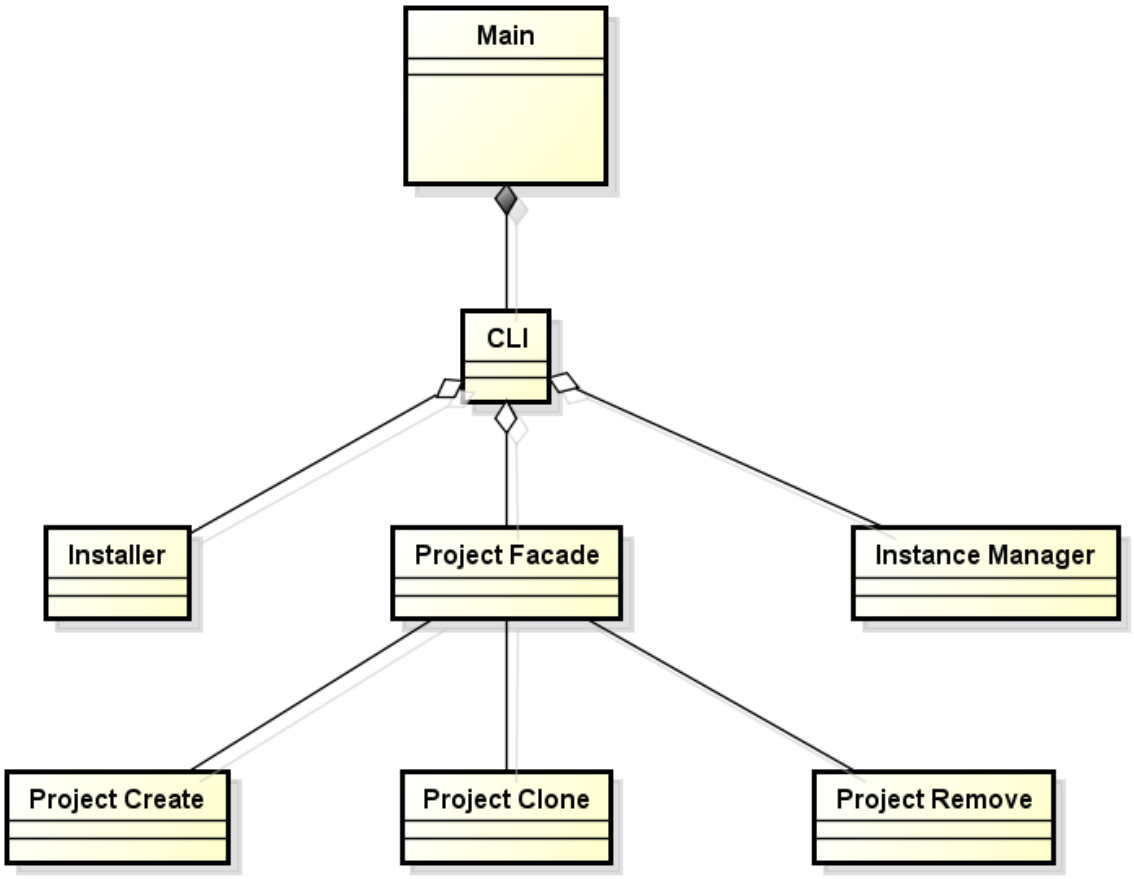

Figura 1: Diagramma delle classi relativo alla gestione del framework da parte dell'utente sviluppatore

3.2.1.2 Descrizione

Descrizione...

3.2.1.3 Classi

3.2.1.3.0.1 CLI

Nome
MaaPCLI::CLI
Descrizione
Classe che rappresenta l'interfaccia a riga di comando.

Utilizzo
Viene utilizzata dall'utente sviluppatore per interagire con il framework.
Relazioni con altre classi

- **MaaPCLI::Installer**
  Relazione uscente, contiene un riferimento ad un oggetto di tipo Installer per avviare l'installazione del framework;
- **MaaPCLI::ProjectFacade**
  Relazione uscente, contiene un riferimento ad un oggetto di tipo ProjectFacade per creare un nuovo progetto, clonare uno esistente oppure eliminarlo;
- **MaaPCLI::InstanceManager**
  Relazione uscente, contiene un riferimento ad un oggetto di tipo InstanceManager per istanziare una istanza di progetto MaaP precedentemente creato.

3.2.1.3.0.2 Installer

Nome
MaaPCLI::Installer
Descrizione
Classe che rappresenta lo script di installazione del framework.
Utilizzo
Viene utilizzata dall'utente sviluppatore per installare il framework e relative dipendenze nel sistema in uso.
Relazioni con altre classi

- **MaaPCLI::CLI**
  Relazione entrante, interazione con l'interfaccia a riga di comando.

3.2.1.3.0.3 InstanceManager

Nome
MaaPCLI::InstanceManager
Descrizione
Classe che rappresenta lo script per l'avvio di un'istanza di un progetto esistente.
Utilizzo
Viene utilizzata dall'utente sviluppatore per avviare il server caricando un determinato progetto.
Relazioni con altre classi

- **MaaPCLI::CLI**
  Relazione entrante, interazione con l'interfaccia a riga di comando.

3.2.1.3.0.4 ProjectFacade

Nome
MaaPCLI::ProjectFacade
Descrizione
Classe che rappresenta la classe Facade nel design pattern Facade
Utilizzo
Viene utilizzata dall'utente sviluppatore per interagire con il framework per la creazione di un nuovo progetto e/o per la clonazione, eliminazione di un progetto esistente.
Relazioni con altre classi

- **MaaPCLI::ProjectCreate**
  Relazione uscente, contiene un riferimento ad un oggetto di tipo ProjectCreate per avviare la creazione di un nuovo progetto;
- **MaaPCLI::ProjectClone**
  Relazione uscente, contiene un riferimento ad un oggetto di tipo ProjectClone per avviare la clonazione di un progetto esistente;
- **MaaPCLI::ProjectRemove**
  Relazione uscente, contiene un riferimento ad un oggetto di tipo ProjectRemove per eliminare un progetto esistente;

3.2.1.3.0.5 ProjectCreate

Nome
MaaPCLI::ProjectCreate
Descrizione
Classe che rappresenta una classe del design patter Facade.
Utilizzo
Viene utilizzata dall'utente sviluppatore per avviare la creazione di un nuovo progetto.
Relazioni con altre classi

- **MaaPCLI::ProjectFacade**
  Relazione entrante, interazione con la facciata ProjectFacade.

3.2.1.3.0.6 ProjectClone

Nome
MaaPCLI::ProjectClone
Descrizione
Classe che rappresenta una classe del design patter Facade.
Utilizzo
Viene utilizzata dall'utente sviluppatore per avviare la clonazione di un progetto esistente.
Relazioni con altre classi

- **MaaPCLI::ProjectFacade**
  Relazione entrante, interazione con la facciata ProjectFacade.

3.2.1.3.0.7 ProjectRemove

Nome
MaaPCLI::ProjectRemove
Descrizione
Classe che rappresenta una classe del design patter Facade.
Utilizzo
Viene utilizzata dall'utente sviluppatore per eliminare un progetto esistente.
Relazioni con altre classi

- **MaaPCLI::ProjectFacade**
  Relazione entrante, interazione con la facciata ProjectFacade.

3.2.2 Package

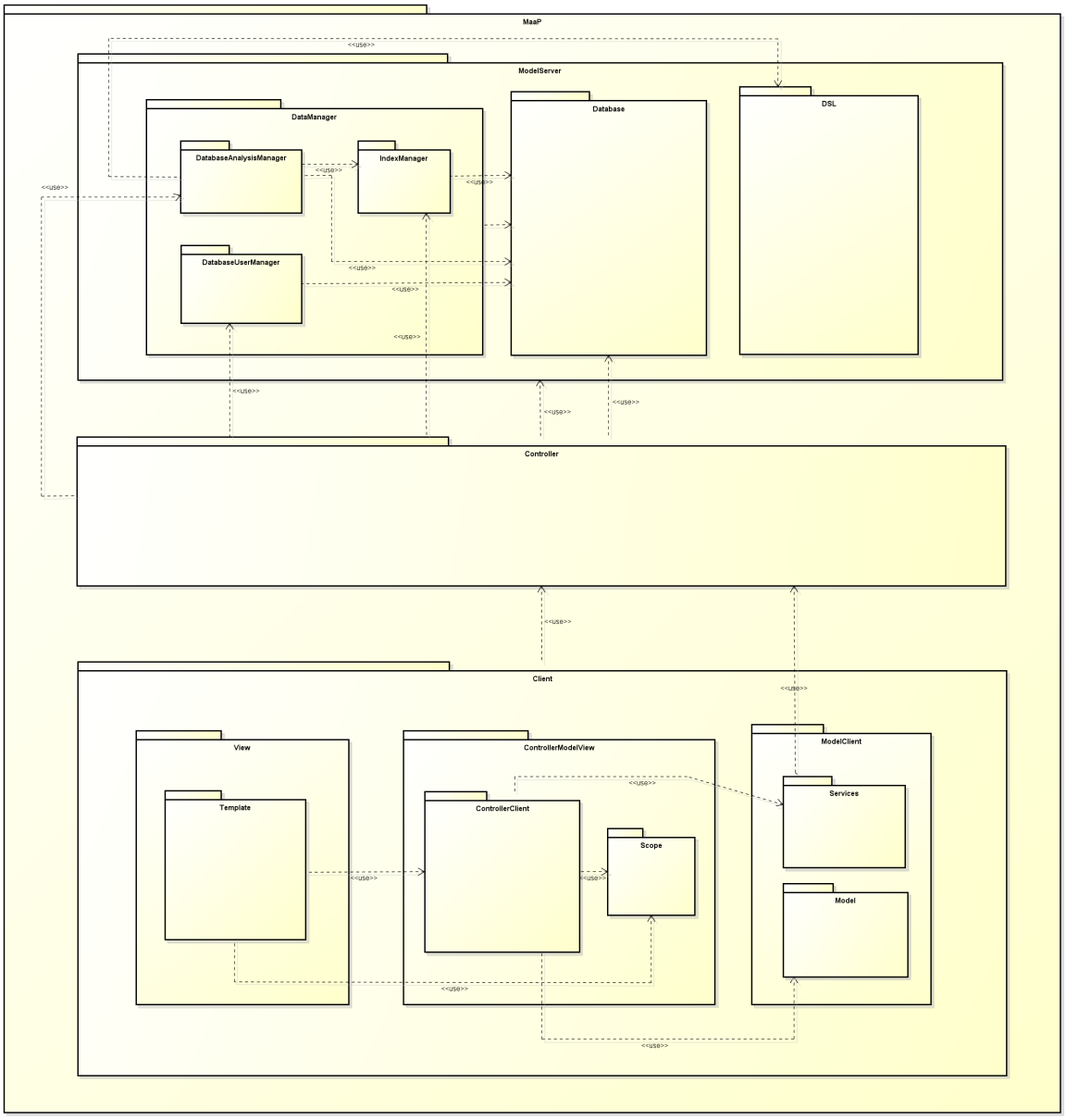

Figura 2: Architettura generale del software - vista package

Nel precedente diagramma sono presenti le relazioni tra i package Client, Controller e ModelServer. Vengono inoltre presentati tutti i sotto-package così da facilitare la comprensione dell'intero sistema.

3.2.3 Classi

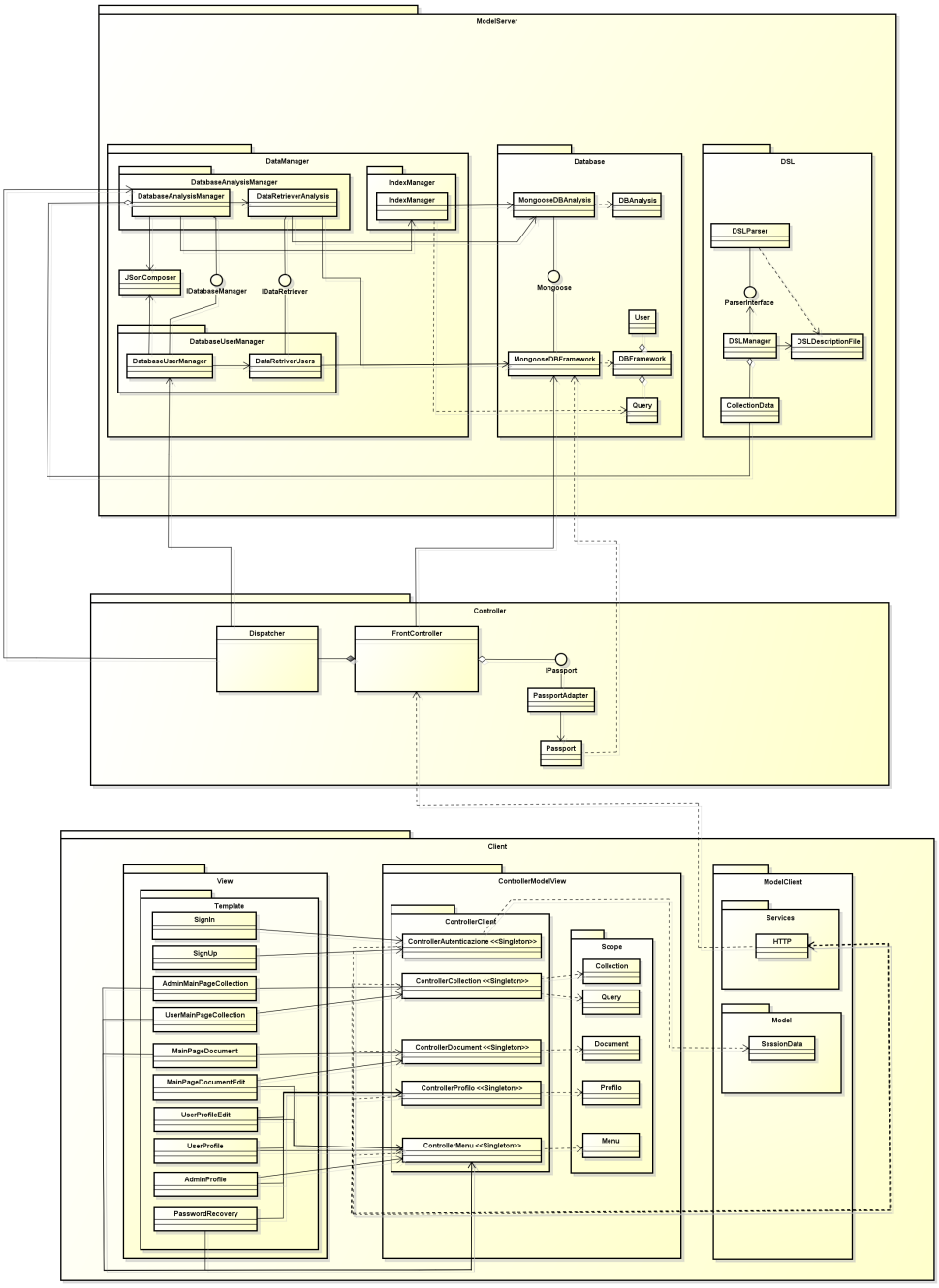

Figura 3: Architettura generale del software

Nel precedente diagramma è presente l'architettura ad alto livello del software e vengono indicate le classi fondamentali per rappresentare le relazioni del modello Three-tier. I diagrammi di sequenza relativi allo scambio di segnali, lo scopo ed il contesto di utilizzo sono presenti nella sezione ????.

3.2.3.1 ModelServer

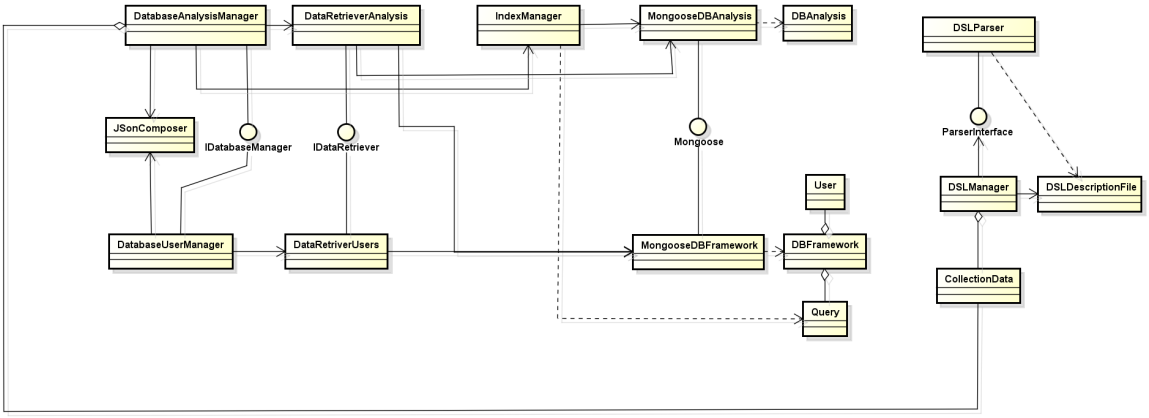

Figura 4: Diagramma delle classi del ModelServer

Nel ModelServer sono presenti oggetti che rappresentano:

- Il database di analisi e quello degli utenti;
- La gestione del file DSL e il suo parsing;
- La 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.

3.2.3.2 Controller

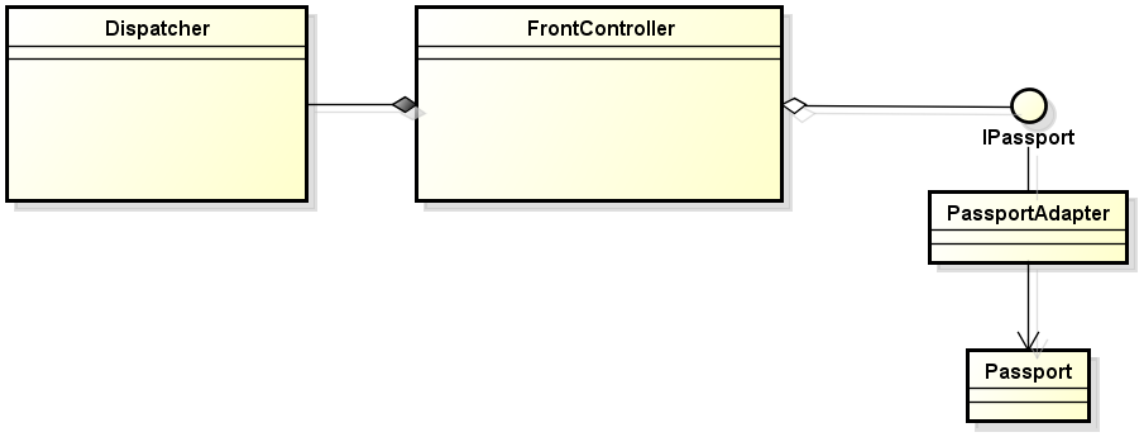

Figura 5: Diagramma delle classi del Controller

Il controller è responsabile dell'autenticazione delle richieste e del loro routing da Client a Model-Server e viceversa.

3.2.3.3 Client

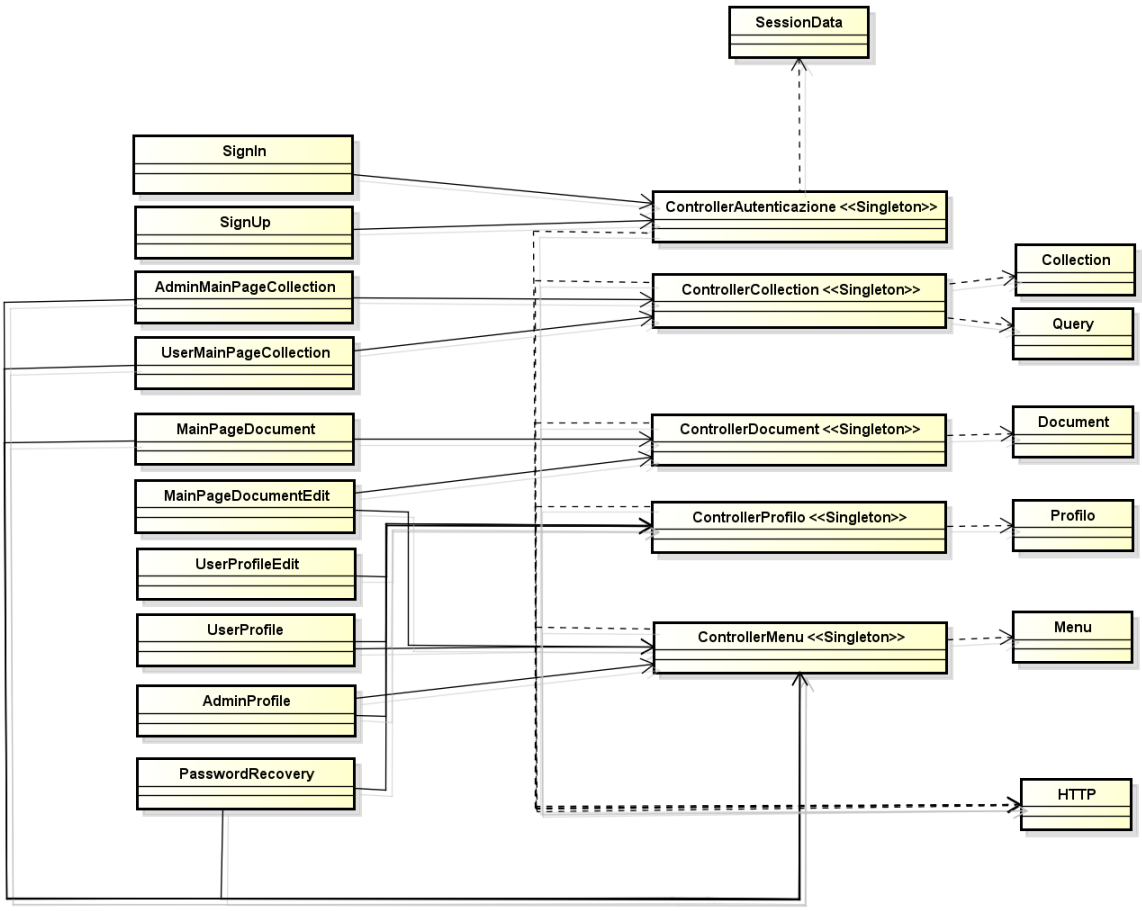

Figura 6: Diagramma delle classi del Client

Nel Client sono presenti oggetti che rappresentano:

- I template per le pagine web;
- I Controller per la gestione dei template;
- Lo Scope per l'aggiornamento dei dati dei template;
- I Servizi utilizzati dai Controller.

4 Componenti e Classi

4.1 MaaP

4.1.1 Informazioni sul package

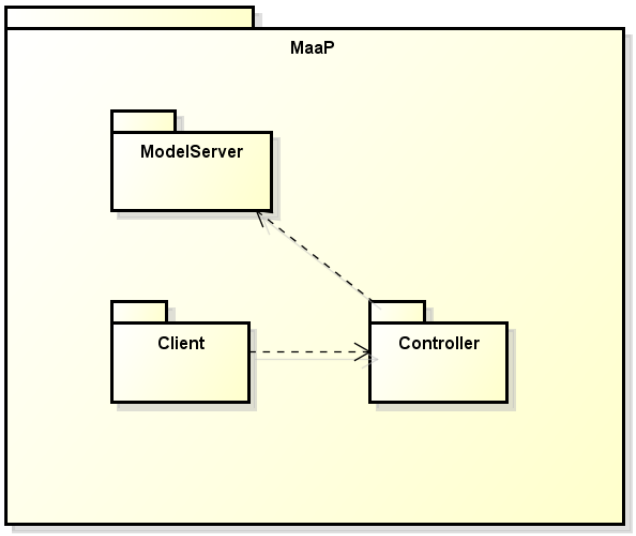

Figura 7: Componenti MaaP

4.1.1.1 Descrizione

Namespace globale per il progetto. Le relazioni tra i package ModelServer, Controller e Client identificano il modello di architettura Three-tier.

4.1.1.2 Sotto-componenti

- MaaP::ModelServer
- MaaP::Controller
- MaaP::Client

4.2 MaaP::ModelServer

4.2.1 Informazioni sul package

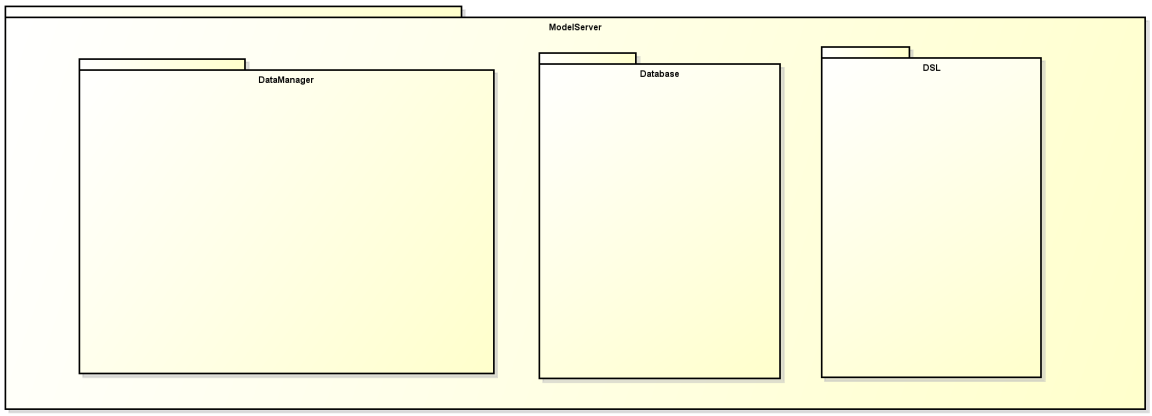

Figura 8: Componente MaaP::ModelServer

4.2.1.1 Descrizione

Package per il componente ModelServer del modello di architettura Three-tier.

4.2.1.2 Sottocomponenti

- MaaP::ModelServer::DataManager;
- MaaP::ModelServer::Database;
- MaaP::ModelServer::DSL.

4.2.2 MaaP::ModelServer::DataManager

4.2.2.1 Informazioni sul package

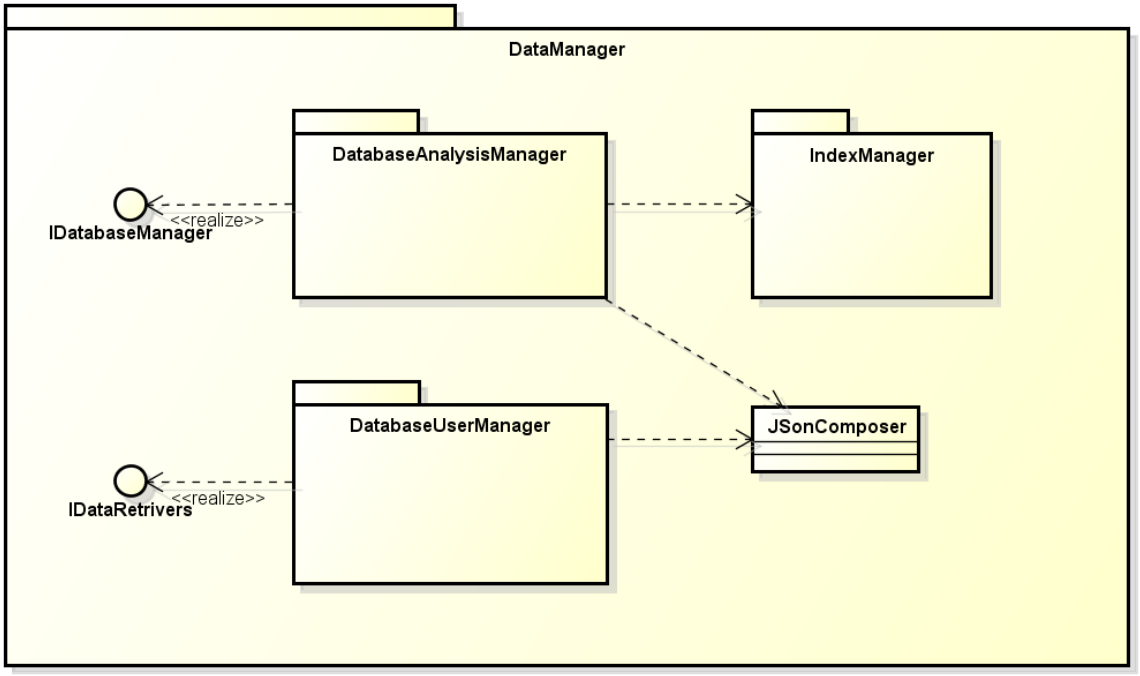

Figura 9: Componente MaaP::ModelServer::DataManager

4.2.2.2 Descrizione

Componente parte del ServerModel per la gestione dei dati.

4.2.2.3 Sotto-componenti

- MaaP::ModelServer::DataManager::DataBaseAnalysisManager;
- MaaP::ModelServer::DataManager::DatabaseUserManager;
- MaaP::ModelServer::DataManager::IndexManager.

4.2.2.4 Classi

4.2.2.4.1 JSonComposer

Nome
MaaP::ModelServer::DataManager::JSonComposer
Descrizione
Classe che costruisce un file JSON a partire dalla struttura di una Collection, o di un Document, e dai suoi dati.

Utilizzo
Viene utilizzata dai DatabaseManager per costruire il file JSON da inviare al Controller.

4.2.2.4.2 IDatabaseManager

Nome
MaaP::ModelServer::DataManager::IDatabaseManager
Descrizione
Interfaccia che rappresenta il gestore dei database. Contiene tutte le operazioni che si possono effettuare sul database e l'elaborazione dei dati recuperati da essi.
Utilizzo
Viene utilizzata per la gestione delle richieste inoltrate dal Controller.
Classi che ereditano

- MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DatabaseAnalysisManager;
- MaaP::ModelServer::DataManager::DatabaseUserManager::DatabaseUserManager.

4.2.2.4.3 IDataRetriever

Nome
MaaP::ModelServer::DataManager::IDataRetriever
Descrizione
Interfaccia attraverso cui i DatabaseManager dialogano con i batabase. Contiene le operazioni di lettura e scrittura nei database.
Utilizzo
Viene utilizzata per recuperare e inserire dati, sui database, su richiesta dei DataManager.
Classi che ereditano

- MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DataRetrieverAnalysis;
- MaaP::ModelServer::DataManager::DatabaseUserManager::DataRetrieverUsers.

4.2.2.5 MaaP::ModelServer::DataManager::DatabaseAnalysisManager

4.2.2.5.1 Informazioni sul package

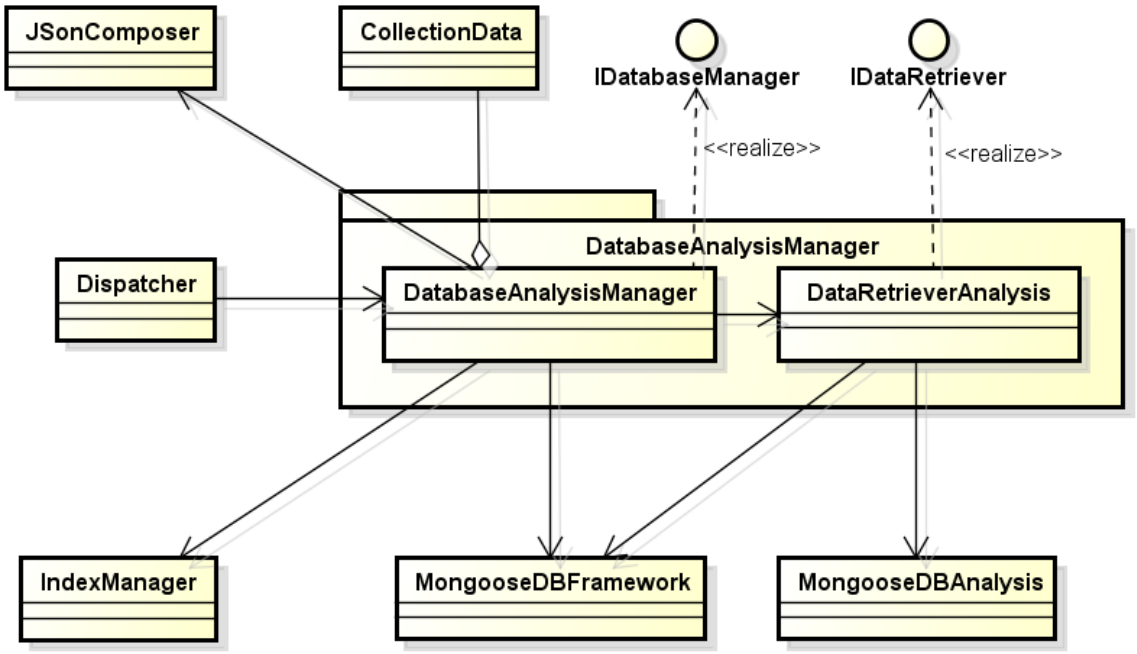

Figura 10: Componente MaaP::ModelServer::DataManager::DatabaseAnalysisManager

4.2.2.5.2 Descrizione

Componente parte del DataManager per la gestione dei dati del database di analisi.

4.2.2.5.3 Classi

4.2.2.5.3.1 DatabaseAnalysisManager

Nome
MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DatabaseAnalysisManager
Descrizione
Classe che rappresenta il gestore dei database di analisi. Contiene tutte le operazioni che si possono effettuare sul database di analisi e l'elaborazione dei dati recuperati da essi.
Utilizzo
Viene utilizzata per la gestione delle richieste, relative al database di analisi, inoltrate dal Controller.
Classi da cui eredita

- MaaP::ModelServer::DataManager::IDatabaseManager;

Relazioni con altre classi

- **MaaP::ModelServer::DataManager::JSonComposer**
  Relazione uscente, utilizza un riferimento a un oggetto di tipo JsonComposer per ottenere il JSON da spedire;
- **MaaP::ModelServer::DSL::CollectionData**
  Relazione uscente, utilizza un riferimento a un oggetto CollectionData che contiene la struttura di un file di descrizione;
- **MaaP::ModelServer::DataManager::DataAnalysisManager::DataRetrieverAnalysis**
  Relazione uscente, utilizza un riferimento a un oggetto DataRetrieverAnalysis per relazionarsi con il database di analisi;
- **MaaP::ModelServer::DataManager::IndexManager::IndexManager**
  Relazione uscente, utilizza un riferimento a un oggetto IndexManager per la creazione degli indici;
- **MaaP::Controller::Dispatcher**
  Relazione entrante, interazioni con le funzionalità del gestore del database di analisi.

4.2.2.5.3.2 DatabaseRetrieverAnalysis

Nome
MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DatabaseRetrieverAnalysis
Descrizione
Classe che rappresenta l'oggetto per interagire con i database.
Utilizzo
Viene utilizzata per inserire e leggere dati sui database di analisi e framework.
Classi da cui eredita

- MaaP::ModelServer::DataManager::IDataRetriever;

Relazioni con altre classi

- **MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DatabaseAnalysisManager**
  Relazione entrante, interazione con il database;
- **MaaP::ModelServer::Database::MongooseDBAnalysis**
  Relazione uscente, utilizza un riferimento a un oggetto di tipo MongooseDBAnalysis per creare lo schema dei dati del database di analisi e per interagire con essi;
- **MaaP::ModelServer::Database::MongooseDBFramework**
  Relazione uscente, utilizza un riferimento a un oggetto di tipo MongooseDBFramework per creare lo schema dei dati del database del framework e per interagire con essi;

4.2.2.6 MaaP::ModelServer::DataManager::DatabaseUserManager

4.2.2.6.1 Informazioni sul package

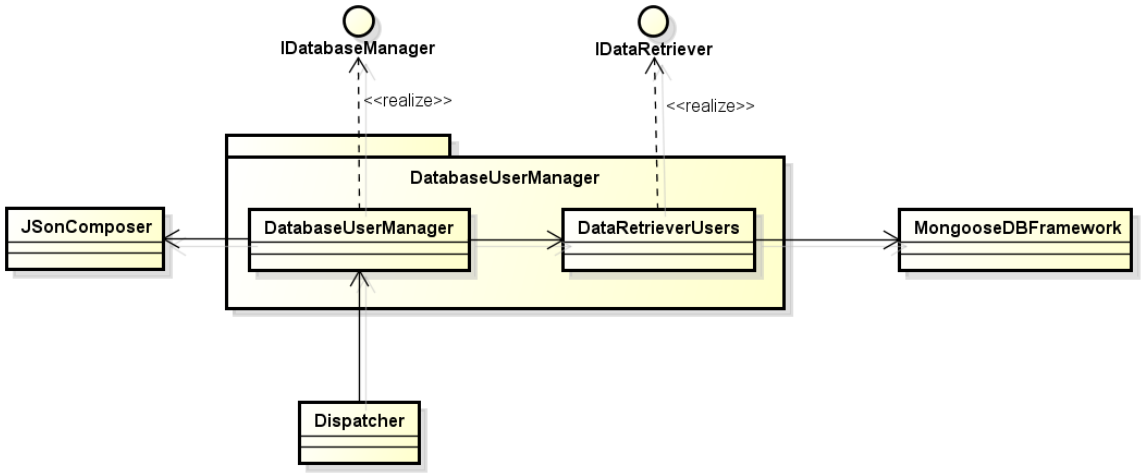

Figura 11: Componente MaaP::ModelServer::DataManager::DatabaseUserManager

4.2.2.6.2 Descrizione

Componente parte del DataManager per la gestione dei dati del database del framwork che comprende sia dati utente che impostazioni del sistema.

4.2.2.6.3 Classi

4.2.2.6.3.1 DatabaseUserManager

Nome
MaaP::ModelServer::DataManager::DatabaseUserManager::DatabaseUserManager
Descrizione
Classe che rappresenta il gestore del database del framework. Contiene tutte le operazioni che si possono effettuare sul database del framework e l'elaborazione dei dati recuperati da esso.
Utilizzo
Viene utilizzata per la gestione delle richieste relative al database del framework inoltrate dal Controller.
Classi da cui eredita

- MaaP::ModelServer::DataManager::IDatabaseManager;

Relazioni con altre classi

- **MaaP::ModelServer::DataManager::JSonComposer**
  Relazione uscente, utilizza un riferimento a un oggetto di tipo JsonComposer per ottenere il JSON da spedire;

- **MaaP::ModelServer::DataManager::DataUserManager::DataRetrieverUsers**
  Relazione uscente, utilizza un riferimento a un oggetto DataRetrieverUsers per relazionarsi con il database del framework;
- **MaaP::Controller::Dispatcher**
  Relazione entrante, interazioni con le funzionalità del gestore del database di analisi.

4.2.2.6.3.2 DataRetrieverUsers

Nome
MaaP::ModelServer::DataManager::DatabaseUserManager::DataRetrieverUsers
Classe che rappresenta l'oggetto per interagire con il database del framework.
Utilizzo
Viene utilizzata per inserire e leggere dati sul database del framework.
Classi da cui eredita

- MaaP::ModelServer::DataManager::IDataRetriever;

Relazioni con altre classi

- **MaaP::ModelServer::DataManager::DatabaseUserManager::DatabaseUserManager**
  Relazione entrante, interazione con il database;
- **MaaP::ModelServer::Database::MongooseDBFramework**
  Relazione uscente, utilizza un riferimento a un oggetto di tipo MongooseDBFramework per creare lo schema dei dati del database del framework e per interagire con essi.

4.2.2.7 MaaP::ModelServer::DataManager::IndexManager

4.2.2.7.1 Informazioni sul package

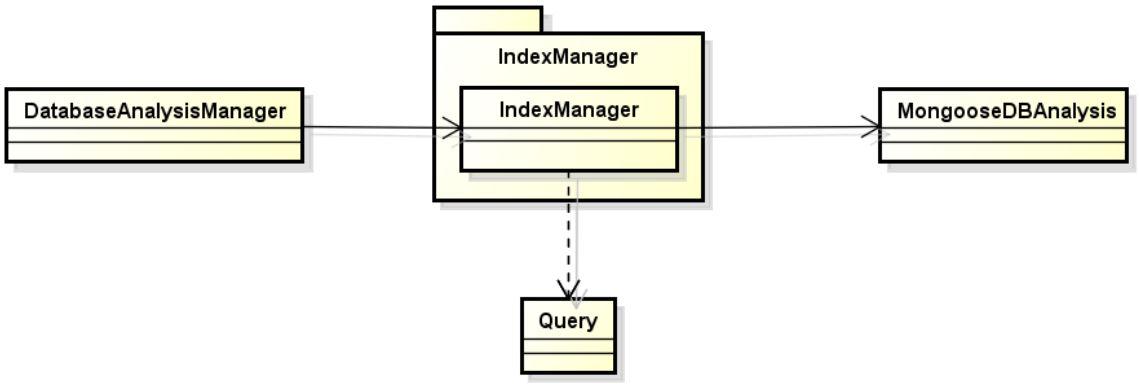

Figura 12: Componente MaaP::ModelServer::DataManager::IndexManager

4.2.2.7.2 Descrizione

Componente parte del DataManager per la creazione e gestione degli indici.

4.2.2.7.3 Classi

4.2.2.7.3.1 IndexManager

Nome
MaaP::ModelServer::DataManager::IndexManager::IndexManager
Descrizione
Classe che rappresenta il gestore degli indici. Contiene tutte le operazioni per la creazione degli indici.
Utilizzo
Viene utilizzata per la creazione di indici personalizzati su richiesta del DatavaseAnalysisManager.
Relazioni con altre classi

- **MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DatabaseAnalysisManager**
  Relazione entrante, interazione con il database;
- **MaaP::ModelServer::DataManager::Database::MongooseDBAnalysis**
  Relazione uscente, utilizza un riferimento ad un oggetto di tipo MongooseDBAnalysis per creare lo schema dei dati del database di analisi e per interagire con essi;
- **MaaP::ModelServer::DataManager::Database::Query**
  Relazione uscente debole, utilizza un riferimento ad un oggetto Query per il recupero delle query più utilizzate.

4.2.3 MaaP::ModelServer::Database

4.2.3.1 Informazioni sul package

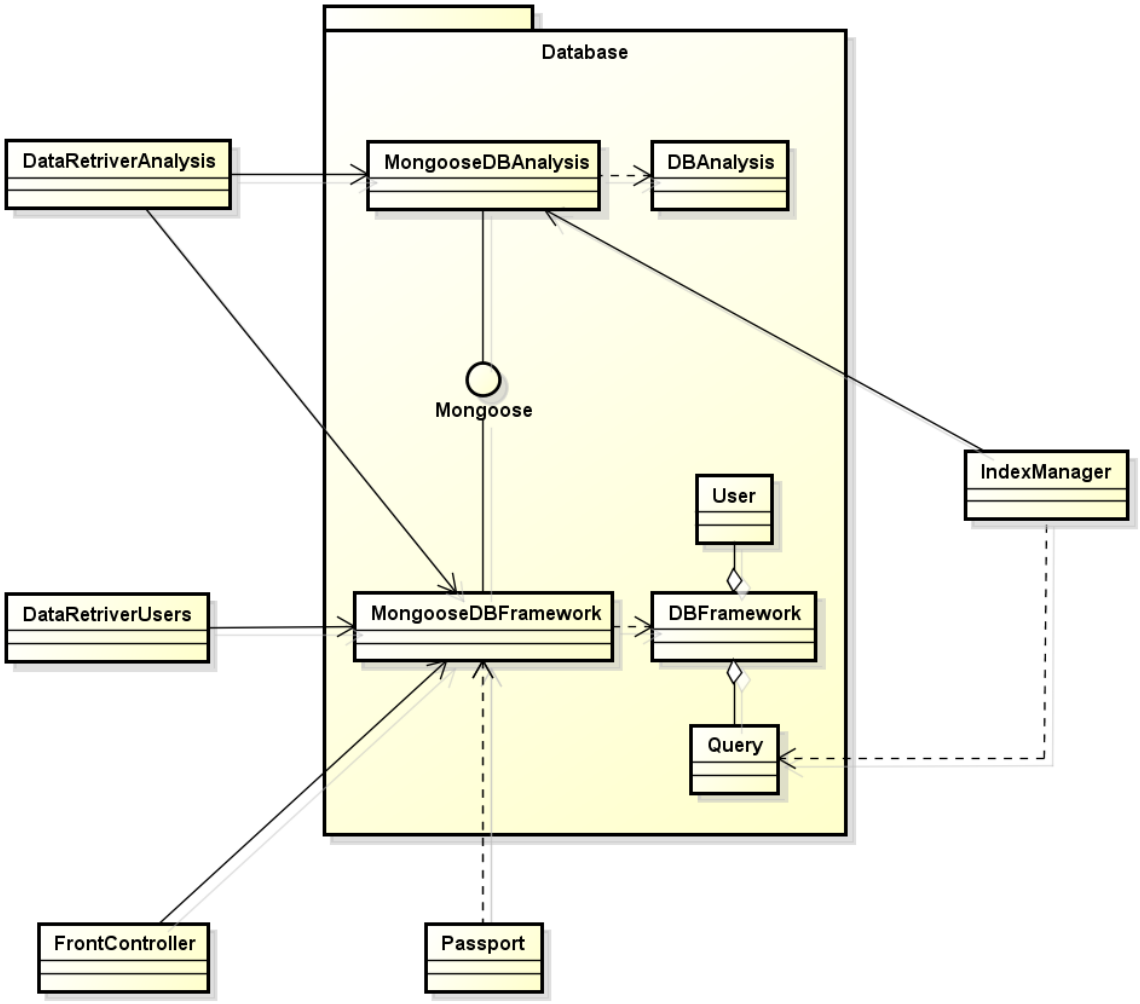

Figura 13: Componente MaaP::ModelServer::Database

4.2.3.2 Descrizione

Componente parte del ModelServer per la gestione dei dati.

4.2.3.3 Classi

4.2.3.3.1 MongooseDBAnalysis

Nome
MaaP::ModelServer::Database::MongooseDBAnalysis

Descrizione
Classe che rappresenta l'interfaccia di connessione con il database di analisi.
Utilizzo
Viene utilizzata per interfacciarsi con il database di analisi fornendo uno schema adeguato.
Classi da cui eredita

- MaaP::ModelServer::Database::Mongoose;

Relazioni con altre classi

- **MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DataRetrieverAnalysis**
  Relazione entrante, interazione con il database di analisi;
- **MaaP::ModelServer::DataManager::IndexManager::IndexManager**
  Relazione entrante, interazione con il database di analisi;
- **MaaP::ModelServer::Database::DBAnalysis**
  Relazione uscente debole, utilizza un riferimento al database di analisi a cui connettersi.

4.2.3.3.2 DBAnalysis

Nome
MaaP::ModelServer::Database::DBAnalysis
Descrizione
Classe che rappresenta il database di analisi.
Utilizzo
Viene utilizzata per contenere i dati di analisi.
Relazioni con altre classi

- **MaaP::ModelServer::Database::MongooseDBAnalysis**
  Relazione entrante debole, interazione con il database di analisi.

4.2.3.3.3 Mongoose

Nome
MaaP::ModelServer::Database::Mongoose
Descrizione
Interfaccia che permette di dialogare con i database utilizzando Mongoose.
Utilizzo
Viene utilizzata per interfacciarsi con i vari database.
Classi che ereditano

- MaaP::ModelServer::Database::MongooseDBAnalysis;
- MaaP::ModelServer::Database::MongooseDBFramework.

4.2.3.3.4 MongooseDBFramework

Nome
MaaP::ModelServer::Database::MongooseDBMongooseDBFramework
Descrizione
Classe che rappresenta l'interfaccia di connessione con il database del framework.
Utilizzo
Viene utilizzata per interfacciarsi con il database del framework fornendo uno schema adeguato.
Classi da cui eredita

- MaaP::ModelServer::Database::Mongoose;

Relazioni con altre classi

- **MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DataRetrieverAnalysis**
  Relazione entrante, interazione con il database del framework;
- **MaaP::ModelServer::DataManager::DatabaseUserManager::DataRetrieverUsers**
  Relazione entrante, interazione con il database del framework;
- **MaaP::ModelServer::Database::DBFramework**
  Relazione uscente debole, utilizza un riferimento al database del framework a cui connettersi.

4.2.3.3.5 DBFramework

Nome
MaaP::ModelServer::Database::DBFramework
Descrizione
Classe che rappresenta il database del framework.
Utilizzo
Viene utilizzata per contenere i dati utente ed impostazioni varie del sistema.
Relazioni con altre classi

- **MaaP::ModelServer::Database::User**
  Relazione uscente, utilizza un riferimento ad un oggetto User per gestire i dati utente.
- **MaaP::ModelServer::Database::Query**
  Relazione uscente, utilizza un riferimento ad un oggetto Query per gestire la lista di query fin'ora effettuate dal sistema;
- **MaaP::Controller::FrontController**
  Relazione entrante, interazione con il database del framework;
- **MaaP::Controller::Passport**
  Relazione entrante debole, interazione con il database del framework.

4.2.3.3.6 User

Nome
MaaP::ModelServer::Database::User
Descrizione
Classe che rappresenta la parte contenuta nel database del framework relativa ai dati utenti.
Utilizzo
Viene utilizzata per contenere i dati utente.
Relazioni con altre classi

- **MaaP::ModelServer::Database::DBFramework**
  Relazione entrante, interazione con i dati utente.

4.2.3.3.7 Query

Nome
MaaP::ModelServer::Database::Query
Descrizione
Classe che rappresenta la parte contenuta nel database del framework relativa alle query effettuate del sistema.
Utilizzo
Viene utilizzata per contenere le query effettuate del sistema.
Relazioni con altre classi

- **MaaP::ModelServer::Database::DBFramework**
  Relazione entrante, interazione con le query effettuate del sistema.
- **MaaP::ModelServer::DataManager::IndexManager::IndexManager**
  Relazione entrante debole, interazione con le query effettuate del sistema.

4.2.4 MaaP::ModelServer::DSL

4.2.4.1 Informazioni sul package

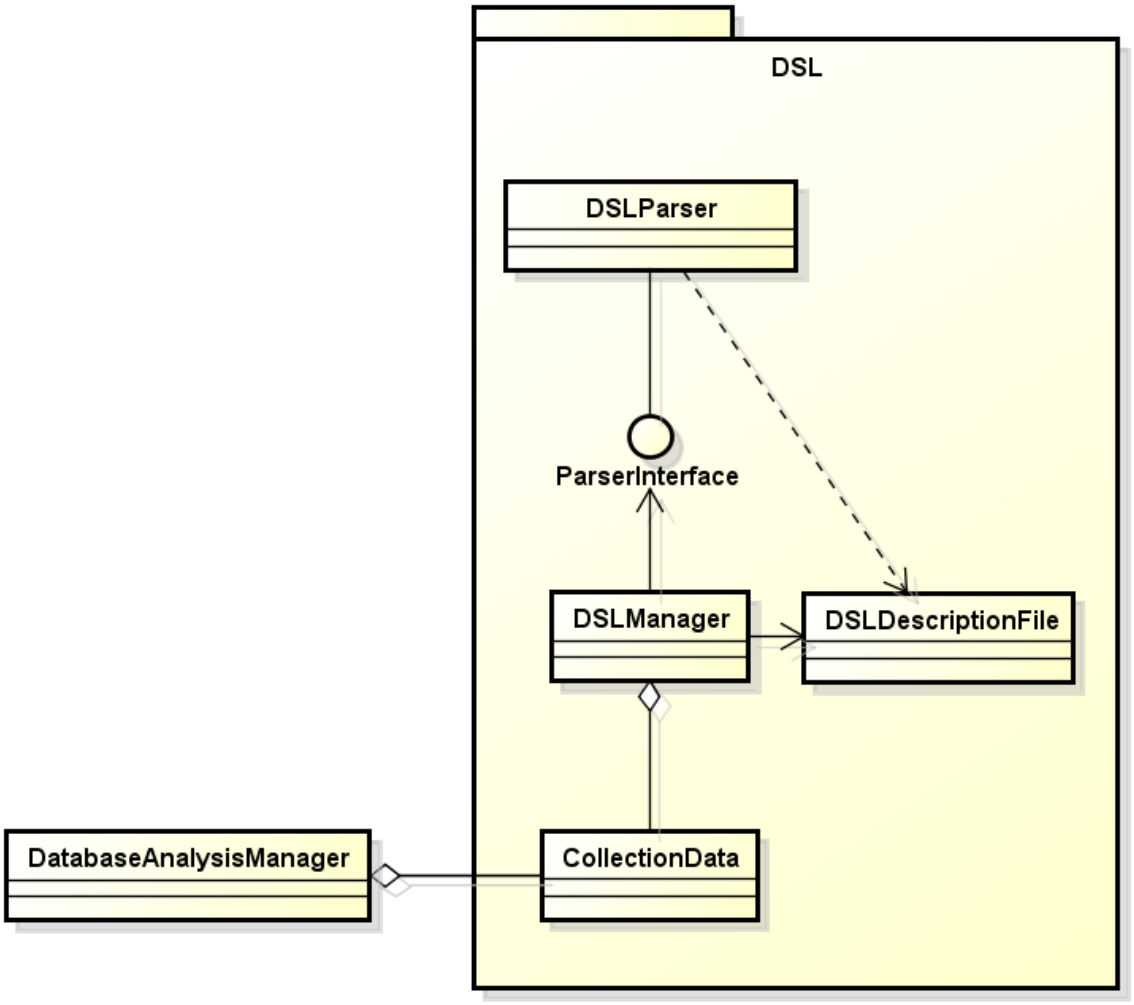

Figura 14: Componente MaaP::ModelServer::DSL

4.2.4.2 Descrizione

Componente parte del ServerModel per la gestione dei file di descrizione.

4.2.4.3 Sotto-componenti

4.2.4.4 Classi

4.2.4.4.1 ParserInterface

Nome
MaaP::ModelServer::DSL::ParserInterface
Descrizione
Interfaccia che rappresenta la componente interfaccia del design pattern strategy per il parser di un linguaggio DSL.
Utilizzo
Viene utilizzata per la effettuare il parsing di un file di descrizione.
Classi che ereditano

- MaaP::ModelServer::DSL::DSLParser.

4.2.4.4.2 DSLParser

Nome
MaaP::ModelServer::DSL::DSLParser
Descrizione
Classe che rappresenta l'algoritmo per il parser DSL del design pattern strategy.
Utilizzo
Viene utilizzata all'avvio del sistema per eseguire il parsing dei file di descrizione. **Classi da cui eredita**

- MaaP::ModelServer::DSL::ParserInterface;

Relazioni con altre classi

- **MaaP::ModelServer::DSL::DSLDescriptionFile**
  Relazione uscente debole, utilizza un riferimento ad un oggetto DSLDescriptionFile per leggere il file di descrizione;

4.2.4.4.3 DSLManager

Nome
MaaP::ModelServer::DSL::DSLManager
Descrizione
Classe che rappresenta il gestore dei file di descrizione. Contiene tutte le operazioni per eseguire il parsing dei file di descrizione e per salvare il risultato su appositi file di tipo CollectionData
Utilizzo
Viene utilizzata all'avvio del sistema per eseguire il parsing dei file di descrizione e salvare il risultato su file. **Classi da cui eredita Relazioni con altre classi**

- **MaaP::ModelServer::DSL::ParserInterface**
  Relazione uscente, utilizza un riferimento ad un oggetto ParserInterface per eseguire il parsing del file di descrizione;
- **MaaP::ModelServer::DSL::DSLDescriptionFile**
  Relazione uscente, utilizza un riferimento ad un oggetto DSLDescriptionFile per leggere il file di descrizione;

- **MaaP::ModelServer::DSL::CollectionData**
  Relazione uscente, utilizza un riferimento ad un oggetto CollectionData per salvare i risultati dell'operazione di parsing.

4.2.4.4.4 CollectionData

Nome
MaaP::ModelServer::DSL::CollectionData
Descrizione
Classe che rappresenta il file contenente il risultato dell'operazione di parsing.
Utilizzo
Viene utilizzata all'avvio del sistema per salvare il risultato dell'operazione di parsing del file di descrizione. **Relazioni con altre classi**

- **MaaP::ModelServer::DSL::DSLManager**
  Relazione entrante, interazione con il file;
- **MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DatabaseAnalysisManager**
  Relazione entrante, interazione con il file.

4.3 MaaP::Controller

4.3.1 Informazioni sul package

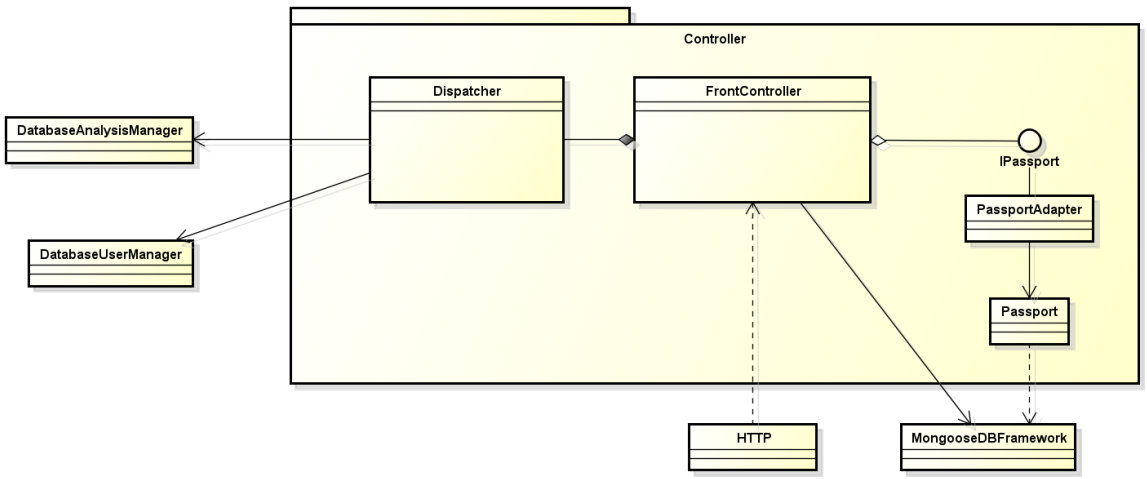

Figura 15: Componente MaaP::Controller

4.3.1.1 Descrizione

Package per il componente Controller del modello di architettura three-tier.

4.3.1.2 Classi

4.3.1.2.1 IPassport

Nome
MaaP::Controller::IPassport
Descrizione
Interfaccia che rappresenta il componente target del design pattern object adapter.
Utilizzo
Viene utilizzata per gestire l'autenticazione utente. **Relazioni con altre classi**

- **MaaP::Controller::FrontController**
  Relazione entrante, interazione con il gestore dell'autenticazione.

Classi che ereditano

- MaaP::Controller::PassportAdapter.

4.3.1.2.2 PassportAdapter

Nome
MaaP::Controller::PassportAdapter
Descrizione
Classe che rappresenta il componente adapter del design pattern object adapter.
Utilizzo
Viene utilizzata per gestire l'autenticazione utente. **Classi da cui eredita**

- MaaP::Controller::IPassport.

Relazioni con altre classi

- **MaaP::Controller::Passport**
  Relazione uscente, utilizza un riferimento ad un oggetto di tipo Passport per gestire l'autenticazione utente.

4.3.1.2.3 Passport

Nome
MaaP::Controller::Passport
Descrizione
Classe che rappresenta il componente adaptee del design patter object adapter.
Utilizzo
Viene utilizzata per gestire l'autenticazione utente. **Relazioni con altre classi**

- **MaaP::ModelServer::Database::MongooseDBFramework**
  Relazione uscente debole, utilizza un riferimento ad un oggetto MongooseDBFramework per accedere ai dati utente.

4.3.1.2.4 FrontController

Nome
MaaP::Controller::FrontController
Descrizione
Classe che rappresenta il componente controller del design patter Front Controller.

Utilizzo
Viene utilizzata per gestire le richieste del client ed inoltrarle al dispatcher. **Relazioni con altre classi**

- **MaaP::Controller::IPassport**
  Relazione uscente, contiene un riferimento ad un oggetto IPassport per gestire l'autenticazione utente;
- **MaaP::Controller::Dispatcher**
  Relazione uscente, contiene un riferimento ad un oggetto Dispatcher per smistare le richieste del client ai vari manager;
- **MaaP::ModelServer::Database::MongooseDBFramework**
  Relazione uscente, contiene un riferimento ad un oggetto MongooseDBFramework per inserire nuovi dati nel database del framework relativi a nuovi utenti;
- **MaaP::Client::ModelClient::Services::HTTP**
  Relazione entrante debole, interazione con il servizio HTTP.

4.3.1.2.5 Dispatcher

Nome
MaaP::Controller::Dispatcher
Descrizione
Classe che rappresenta il componente dispatcher del design patter Front Controller.
Utilizzo
Viene utilizzata per smistare le richieste del client ai vari gestori dei dati. **Relazioni con altre classi**

- **MaaP::Controller::FrontController**
  Relazione entrante, interazione con il FrontController;
- **MaaP::ModelServer::DataManager::DatabaseAnalysisManager::DatabaseAnalysisManager**
  Relazione uscente, contiene un riferimento ad un oggetto DatabaseAnalysisManager per richiedere azioni relative ai dati di analisi;
- **MaaP::ModelServer::DataManager::DatabaseUserManager::DatabaseUserManager**
  Relazione uscente, contiene un riferimento ad un oggetto DatabaseUserManager per richiedere azioni relative ai dati utenti ed impostazioni di sistema.

4.4 MaaP::Client

4.4.1 Informazioni sul package

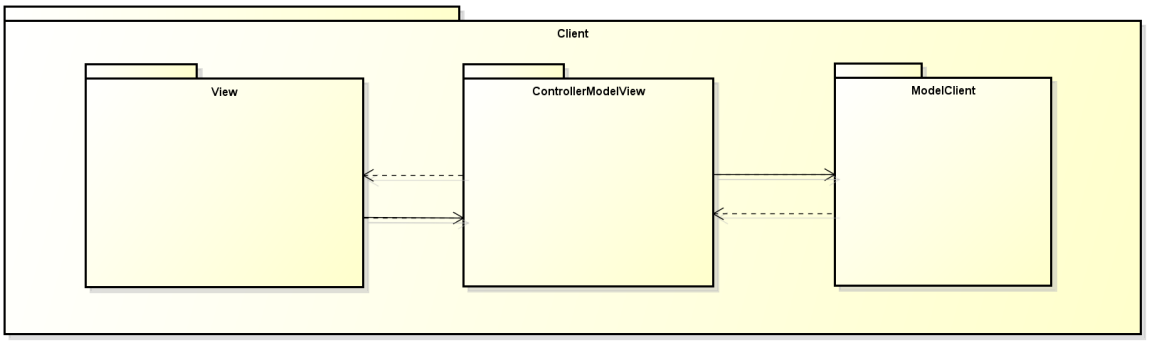

Figura 16: Componente MaaP::Client

4.4.1.1 Descrizione

Package per il componente Client del modello di architettura three-tier.

4.4.1.2 Sottocomponenti

- MaaP::Client::View;
- MaaP::Client::ControllerModelView;
- MaaP::Client::ModelClient.

4.4.2 MaaP::Client::View

4.4.2.1 Informazioni sul package

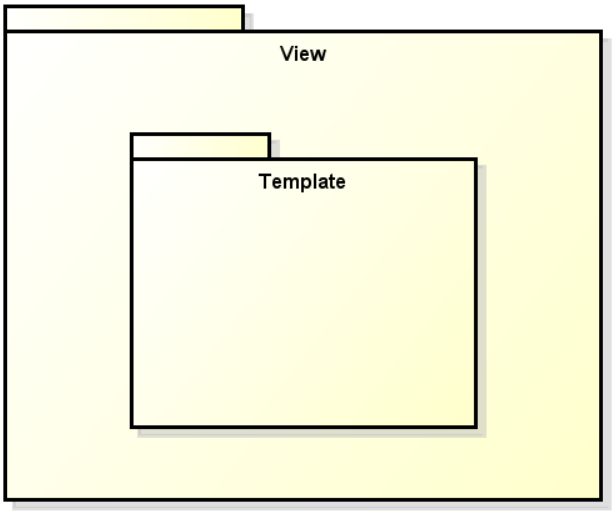

Figura 17: Componente MaaP::Client::View

4.4.2.2 Descrizione

Componente view del design pattern MVVM.

4.4.2.3 Sotto-componenti

- MaaP::Client::Template.

4.4.2.4 MaaP::Client::View::Template

4.4.2.5 Informazioni sul package

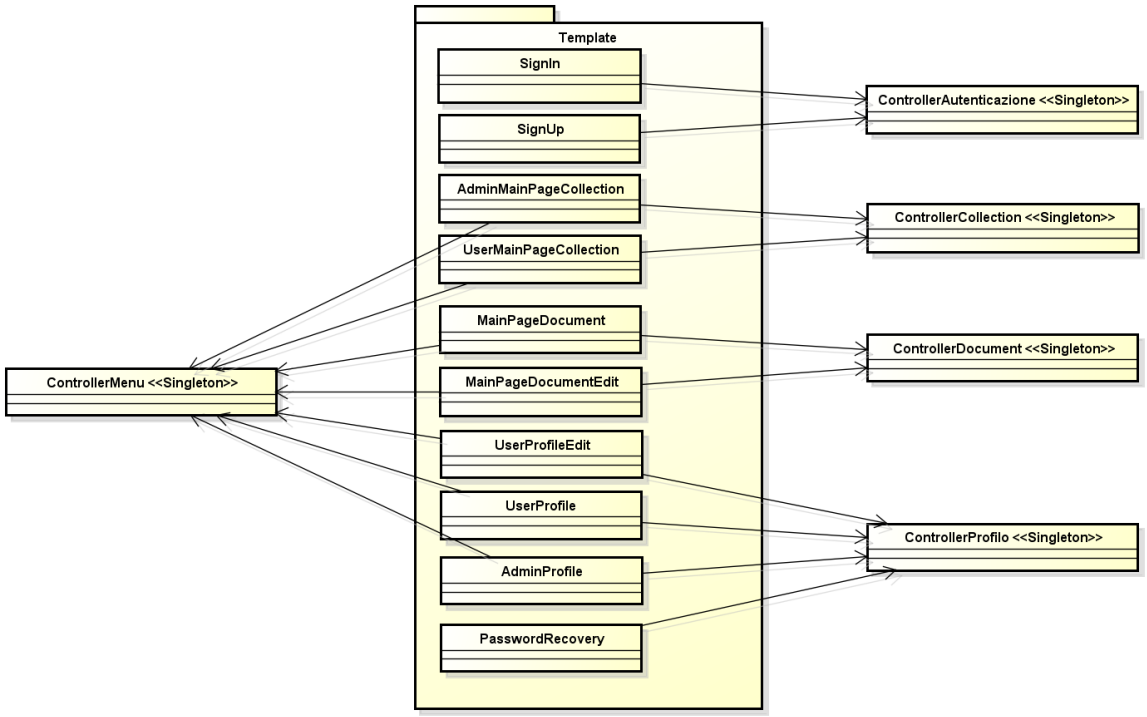

Figura 18: Componente MaaP::Client::View::Template

4.4.2.6 Descrizione

Componente che contiene i template per la visualizzazione delle pagine web.

4.4.2.7 Classi

4.4.2.7.1 SignIn

Nome
MaaP::Client::View::Template::SignIn
Descrizione
Classe che rappresenta il template per la pagina di login.
Utilizzo
Viene utilizzata per renderizzare la pagina web di login.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerAutenticazione**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerAutenticazione per gestire il login utente.

4.4.2.7.2 SignUp

Nome
MaaP::Client::View::Template::SignUp
Descrizione
Classe che rappresenta il template per la pagina di registrazione.
Utilizzo
Viene utilizzata per renderizzare la pagina web di registrazione utente. **Relazioni con altre classi**

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerAutenticazione**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerAutenticazione per gestire la registrazione di un nuovo utente.

4.4.2.7.3 AdminMainPageCollection

Nome
MaaP::Client::View::Template::AdminMainPageCollection
Descrizione
Classe che rappresenta il template per la pagina di visualizzazione Collection per l'utente amministratore.
Utilizzo
Viene utilizzata per renderizzare la pagina web di visualizzazione Collection per l'utente amministratore.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerCollection**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerCollection per gestire la visualizzazione della pagina Collection;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.2.7.4 UserMainPageCollection

Nome
MaaP::Client::View::Template::UserMainPageCollection
Descrizione
Classe che rappresenta il template per la pagina di visualizzazione Collection per l'utente.
Utilizzo
Viene utilizzata per renderizzare la pagina web di visualizzazione Collection per l'utente.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerCollection**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerCollection per gestire la visualizzazione della pagina Collection;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.2.7.5 MainPageDocument

Nome
MaaP::Client::View::Template::MainPageDocument
Descrizione
Classe che rappresenta il template per la pagina di visualizzazione Document.
Utilizzo
Viene utilizzata per renderizzare la pagina web di visualizzazione del Document.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerDocument**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerDocument per gestire la visualizzazione della pagina Document;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.2.7.6 MainPageDocumentEdit

Nome
MaaP::Client::View::Template::MainPageDocumentEdit
Descrizione
Classe che rappresenta il template per la pagina di modifica dei Document.
Utilizzo
Viene utilizzata per renderizzare la pagina web di modifica dei Document.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerDocument**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerDocument per gestire la visualizzazione della pagina di modifica dei Document;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.2.7.7 UserProfileEdit

Nome
MaaP::Client::View::Template::UserProfileEdit
Descrizione
Classe che rappresenta il template per la pagina di modifica del profilo utente.
Utilizzo
Viene utilizzata per renderizzare la pagina web di modifica del profilo utente.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerProfilo per gestire la visualizzazione della pagina di modifica del profilo utente;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.2.7.8 UserProfile

Nome
MaaP::Client::View::Template::UserProfile
Descrizione
Classe che rappresenta il template per la pagina di visualizzazione del profilo utente.
Utilizzo
Viene utilizzata per renderizzare la pagina web di visualizzazione del profilo utente.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerProfilo per gestire la visualizzazione della pagina del profilo utente;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.2.7.9 AdminProfile

Nome
MaaP::Client::View::Template::AdminProfile
Descrizione
Classe che rappresenta il template per la pagina di visualizzazione del profilo utente amministratore.
Utilizzo
Viene utilizzata per renderizzare la pagina web di visualizzazione del profilo utente amministratore.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerProfilo per gestire la visualizzazione della pagina del profilo utente amministratore;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.2.7.10 PasswordRecovery

Nome
MaaP::Client::View::Template::UserProfile
Descrizione
Classe che rappresenta il template per la pagina di recupero password.
Utilizzo
Viene utilizzata per renderizzare la pagina web recupero password. **Relazioni con altre classi**

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerProfilo per gestire la visualizzazione della pagina di recupero password;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione uscente, contiene un riferimento ad un oggetto ControllerMenu per gestire la visualizzazione del menù.

4.4.3 MaaP::Client::ControllerModelView

4.4.3.1 Informazioni sul package

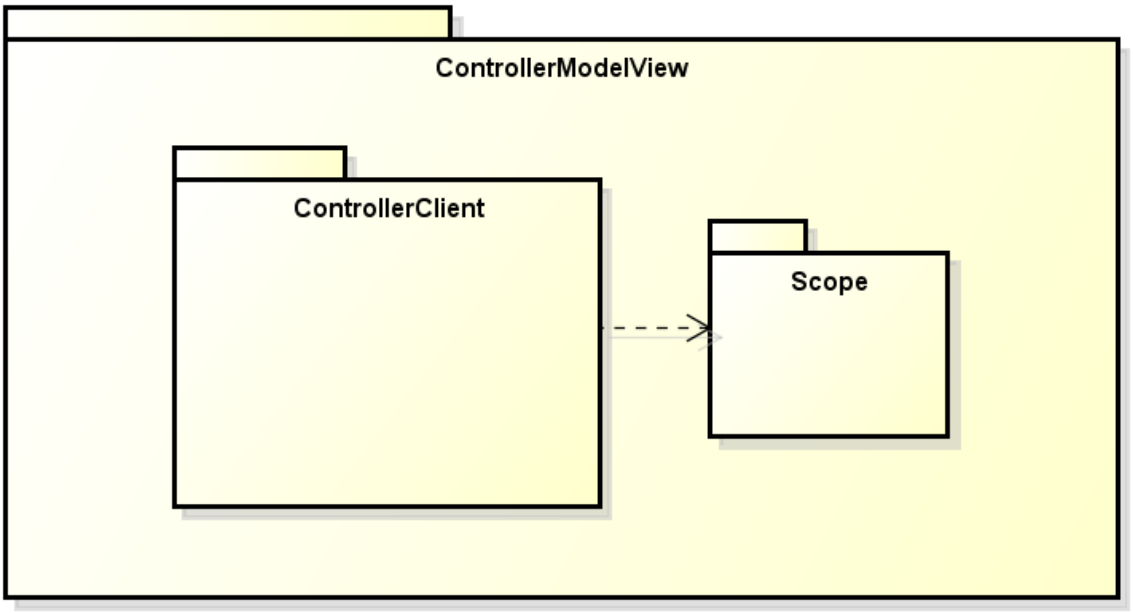

Figura 19: Componente MaaP::Client::ControllerModelView

4.4.3.2 Descrizione

Componente ModelView del design pattern MVVM.

4.4.3.3 Sotto-componenti

- MaaP::Client::ControllerModelView::ControllerClient;
- MaaP::Client::ControllerModelView::Scope.

4.4.3.4 MaaP::Client::ControllerModelView::ControllerClient

4.4.3.4.1 Informazioni sul package

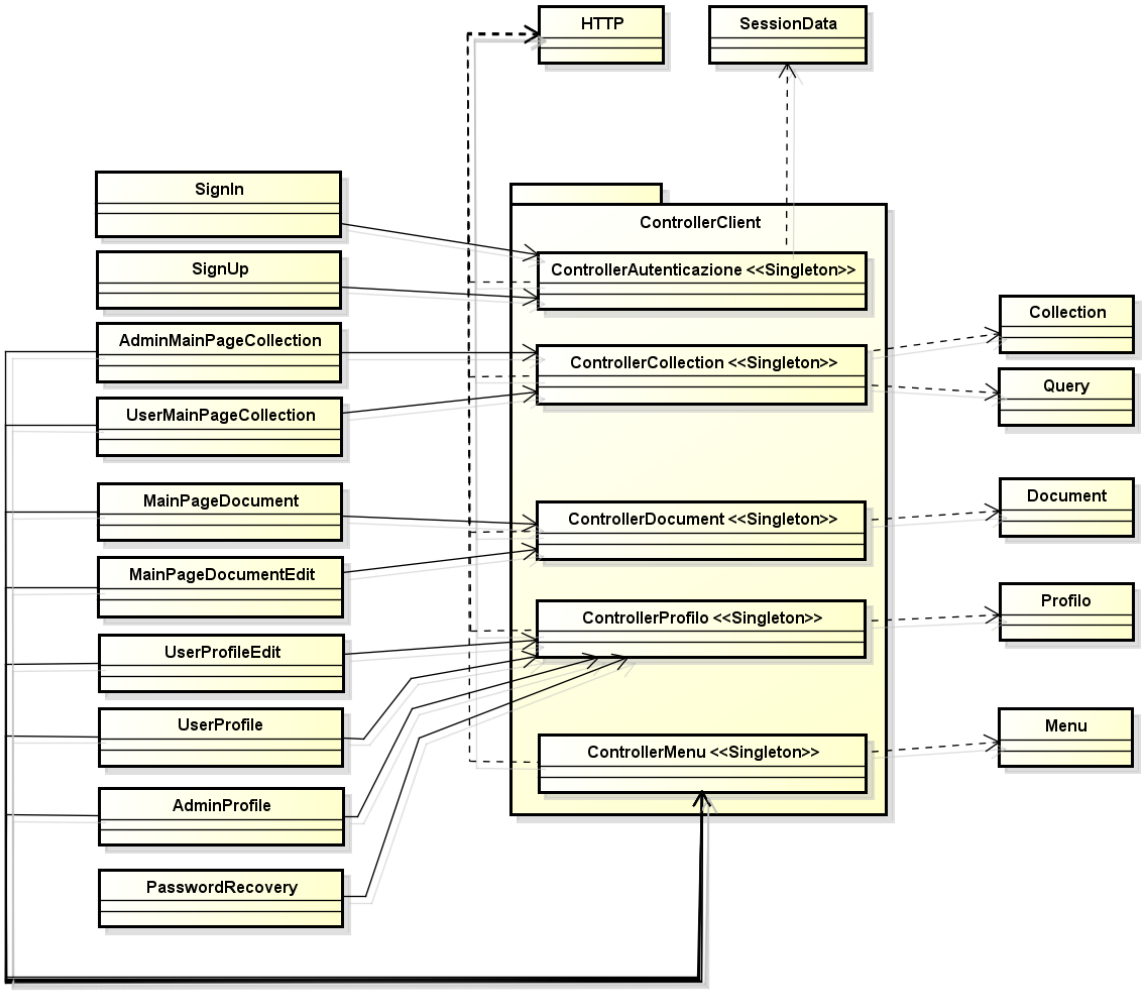

Figura 20: Componente MaaP::Client::ControllerModelView::ControllerClient

4.4.3.4.2 Descrizione

Componente parte del ControllerModelView contenente i vari controller.

4.4.3.4.3 Classi

4.4.3.4.3.1 ControllerAutenticazione

Nome
MaaP::Client::ControllerModelView::ControllerClient::ControllerAutenticazione

Descrizione
Classe che rappresenta il controller per indirizzare le richieste di autenticazione e registrazione.
Utilizzo
Viene utilizzata per la indirizzare le richieste di autenticazione e registrazione.
Relazioni con altre classi

- **MaaP::Client::ModelClient::Services::HTTP**
  Relazione uscente debole, contiene un riferimento ad un oggetto HTTP per utilizzare il relativo servizio;
- **MaaP::Client::ModelClient::Model::SessionData**
  Relazione uscente debole, contiene un riferimento ad un oggetto SessionData per utilizzare i dati di sessione;
- **MaaP::Client::View::Template::SignIn**
  Relazione entrante, interazione con il template;
- **MaaP::Client::View::Template::SignUp**
  Relazione entrante, interazione con il template.

4.4.3.4.3.2 ControllerCollection

Nome
MaaP::Client::ControllerModelView::ControllerClient::ControllerCollection
Descrizione
Classe che rappresenta il controller per indirizzare le richieste di visualizzazione di una pagina Collection.
Utilizzo
Viene utilizzata per indirizzare le richieste di visualizzazione di una pagina Collection.
Relazioni con altre classi

- **MaaP::Client::ModelClient::Services::HTTP**
  Relazione uscente debole, contiene un riferimento ad un oggetto HTTP per utilizzare il relativo servizio;
- **MaaP::Client::ControllerModelView::Scope::Collection**
  Relazione uscente debole, contiene un riferimento ad un oggetto Collection per accedere allo scope relativo ai dati di una Collection;
- **MaaP::Client::ControllerModelView::Scope::Query**
  Relazione uscente debole, contiene un riferimento ad un oggetto Query per accedere allo scope relativo ai dati relativi alle query;
- **MaaP::Client::View::Template::AdminMainPageCollection**
  Relazione entrante, interazione con il template;
- **MaaP::Client::View::Template::UserMainPageCollection**
  Relazione entrante, interazione con il template.

4.4.3.4.3.3 ControllerDocument

Nome
MaaP::Client::ControllerModelView::ControllerClient::ControllerDocument

Descrizione
Classe che rappresenta il controller per indirizzare le richieste di visualizzazione di una pagina Document.
Utilizzo
Viene utilizzata per indirizzare le richieste di visualizzazione di una pagina Document.
Relazioni con altre classi

- **MaaP::Client::ModelClient::Services::HTTP**
  Relazione uscente debole, contiene un riferimento ad un oggetto HTTP per utilizzare il relativo servizio;
- **MaaP::Client::ControllerModelView::Scope::Document**
  Relazione uscente debole, contiene un riferimento ad un oggetto Document per accedere allo scope relativo ai dati di un Document;
- **MaaP::Client::View::Template::MainPageDocument**
  Relazione entrante, interazione con il template;
- **MaaP::Client::View::Template::MainPageDocumentEdit**
  Relazione entrante, interazione con il template.

4.4.3.4.3.4 ControllerProfilo

Nome
MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo
Descrizione
Classe che rappresenta il controller per indirizzare le richieste di visualizzazione di una pagina profilo utente.
Utilizzo
Viene utilizzata per indirizzare le richieste di visualizzazione di una pagina profilo utente.
Relazioni con altre classi

- **MaaP::Client::ModelClient::Services::HTTP**
  Relazione uscente debole, contiene un riferimento ad un oggetto HTTP per utilizzare il relativo servizio;
- **MaaP::Client::ControllerModelView::Scope::Profilo**
  Relazione uscente debole, contiene un riferimento ad un oggetto Profilo per accedere allo scope relativo ai dati del profilo;
- **MaaP::Client::View::Template::UserProfileEdit**
  Relazione entrante, interazione con il template;
- **MaaP::Client::View::Template::UserProfile**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::AdminProfile**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::PasswordRecovery**
  Relazione entrante, interazione con il template.

4.4.3.4.3.5 ControllerMenu

Nome
MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu
Descrizione
Classe che rappresenta il controller per indirizzare le richieste di visualizzazione della parte di pagina relativa al menù.
Utilizzo
Viene utilizzata per indirizzare le richieste di visualizzazione della parte di pagina relativa al menù.
Relazioni con altre classi

- **MaaP::Client::ModelClient::Services::HTTP**
  Relazione uscente debole, contiene un riferimento ad un oggetto HTTP per utilizzare il relativo servizio;
- **MaaP::Client::ControllerModelView::Scope::Menu**
  Relazione uscente debole, contiene un riferimento ad un oggetto Menu per accedere allo scope relativo ai dati del menù;
- **MaaP::Client::View::Template::AdminMainPageCollection**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::UserMainPageCollection**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::MainPageDocument**
  Relazione entrante, interazione con il template;
- **MaaP::Client::View::Template::MainPageDocumentEdit**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::UserProfileEdit**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::UserProfile**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::AdminProfile**
  Relazione entrante, interazione con il template.
- **MaaP::Client::View::Template::PasswordRecovery**
  Relazione entrante, interazione con il template.

4.4.3.5 MaaP::Client::ControllerModelView::Scope

4.4.3.5.1 Informazioni sul package

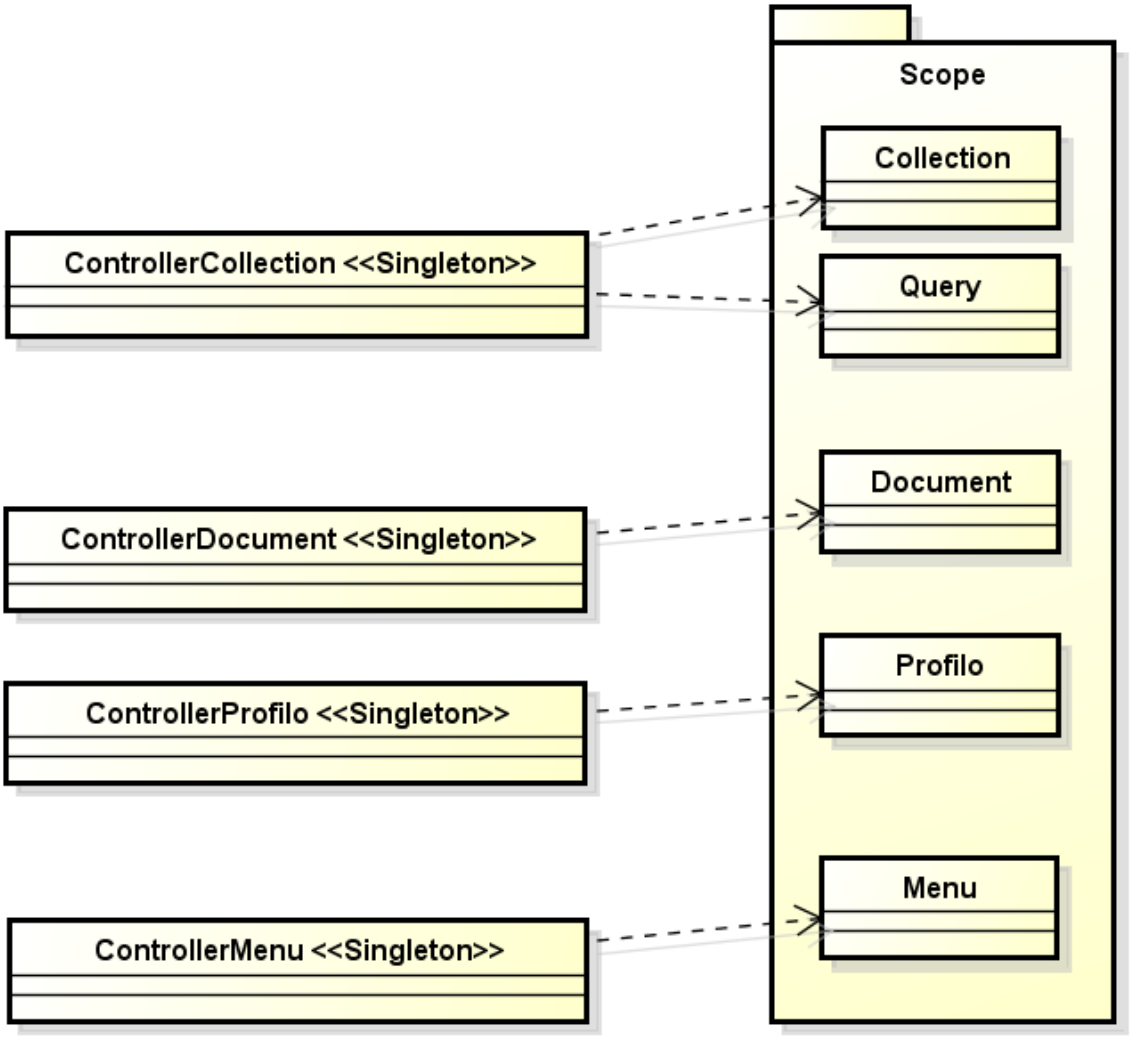

Figura 21: Componente MaaP::Client::ControllerModelView::Scope

4.4.3.5.2 Descrizione

Componente parte del ControllerModelView contenente i dati per renderizzare i template.

4.4.3.5.3 Classi

4.4.3.5.3.1 Collection

Nome
MaaP::Client::ControllerModelView::Scope::Collection
Descrizione
Classe che rappresenta i dati relativi alla Collection da visualizzare.
Utilizzo
Viene utilizzata per memorizzare i dati relativi alla Collection da visualizzare i quali saranno sucessivamente visualizzati nella pagina web.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerCollection**
  Relazione entrante debole, interazione con il controller della Collection;

4.4.3.5.3.2 Query

Nome
MaaP::Client::ControllerModelView::Scope::Query
Descrizione
Classe che rappresenta i dati relativi alle query più utilizzare.
Utilizzo
Viene utilizzata per memorizzare i dati relativi alle query più utilizzate, le quali saranno sucessivamente visualizzate nella pagina web.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerCollection**
  Relazione entrante debole, interazione con il controller della Collection;

4.4.3.5.3.3 Document

Nome
MaaP::Client::ControllerModelView::Scope::Document
Descrizione
Classe che rappresenta i dati relativi al Document da visualizzare.
Utilizzo
Viene utilizzata per memorizzare i dati relativi al Document da visualizzare i quali saranno sucessivamente visualizzati nella pagina web.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerDocument**
  Relazione entrante debole, interazione con il controller del Document;

4.4.3.5.3.4 Profilo

Nome
MaaP::Client::ControllerModelView::Scope::Profilo
Descrizione
Classe che rappresenta i dati relativi al profilo utente da visualizzare.
Utilizzo
Viene utilizzata per memorizzare i dati relativi al profilo utente da visualizzare i quali saranno

sucessivamente visualizzati nella pagina web.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo**
  Relazione entrante debole, interazione con il controller del profilo;

4.4.3.5.3.5 Menu

Nome
MaaP::Client::ControllerModelView::Scope::Menu
Descrizione
Classe che rappresenta i dati relativi al menù da visualizzare.
Utilizzo
Viene utilizzata per memorizzare i dati relativi al menù da visualizzare i quali saranno sucessivamente visualizzati nella pagina web.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione entrante debole, interazione con il controller del menù;

4.4.4 MaaP::Client::ModelClient

4.4.4.1 Informazioni sul package

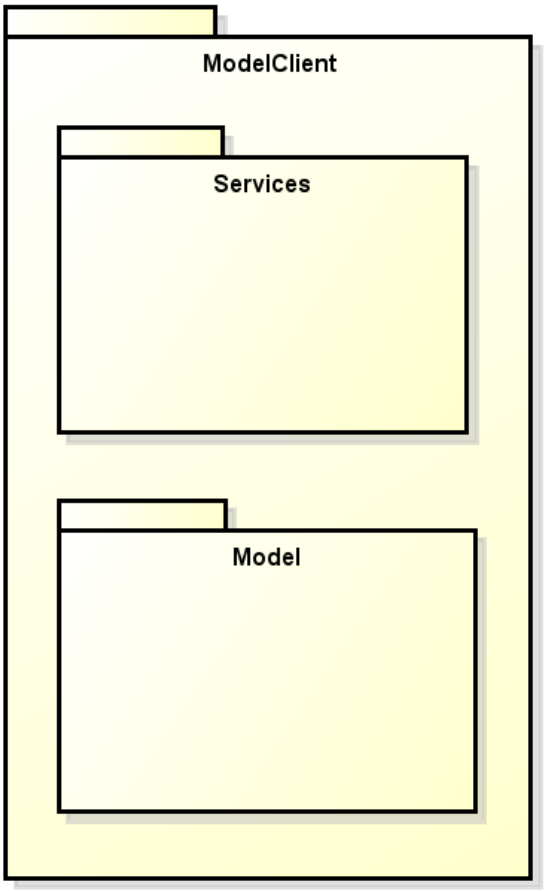

Figura 22: Componente MaaP::Client::ModelClient

4.4.4.2 Descrizione

Componente Model del design pattern MVVM.

4.4.4.3 Sotto-componenti

- MaaP::Client::ModelClient::Services;
- MaaP::Client::ModelClient::Model.

4.4.4.4 MaaP::Client::ModelClient::Services

4.4.4.4.1 Informazioni sul package

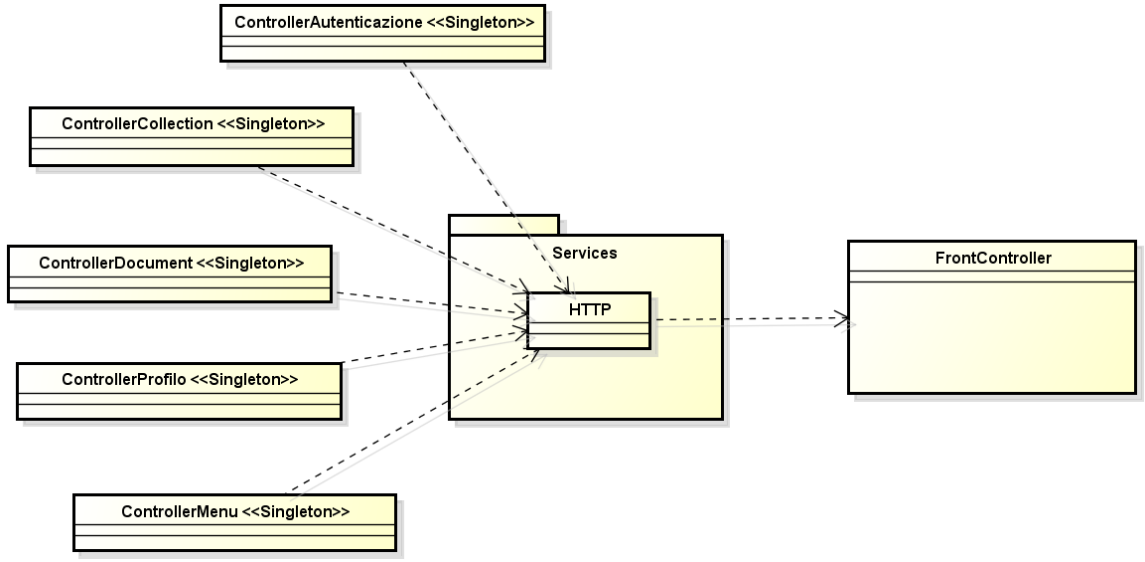

Figura 23: Componente MaaP::Client::ModelClient::Services

4.4.4.4.2 Descrizione

Componente parte del ModelClient contenente i servizi per la comunicazione con il server.

4.4.4.4.3 Classi

4.4.4.4.3.1 HTTP

Nome
MaaP::Client::ModelClient::Services::HTTP
Descrizione
Classe che rappresenta il servizio di comunicazione HTTP con il server.
Utilizzo
Viene utilizzata per inviare richieste HTTP al server.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerAutenticazione**
  Relazione entrante debole, interazione con il controller dell'Autenticazione;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerCollection**
  Relazione entrante debole, interazione con il controller della Collection;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerDocument**
  Relazione entrante debole, interazione con il controller del Document;

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerProfilo**
  Relazione entrante debole, interazione con il controller del profilo;
- **MaaP::Client::ControllerModelView::ControllerClient::ControllerMenu**
  Relazione entrante debole, interazione con il controller del menù;
- **MaaP::Controller::FrontController**
  Relazione uscente debole, contiene un riferimento ad un oggetto di tipo FrontController per inviare richieste HTTP al server.

4.4.4.5 MaaP::Client::ModelClient::Model

4.4.4.5.1 Informazioni sul package

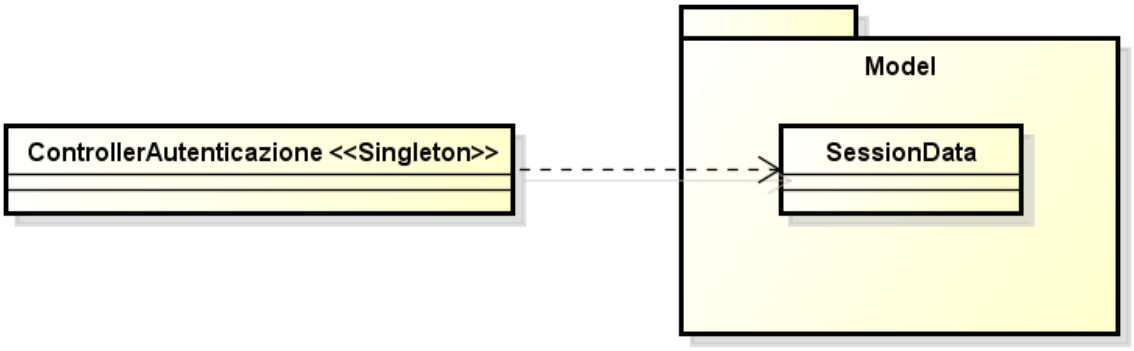

Figura 24: Componente MaaP::Client::ModelClient::Model

4.4.4.5.2 Descrizione

Componente parte del ModelClient contenente i dati di sessione.

4.4.4.5.3 Classi

4.4.4.5.3.1 SessionData

Nome
MaaP::Client::ModelClient::Model::SessionData
Descrizione
Classe che rappresenta i dati di sessione utente.
Utilizzo
Viene utilizzata memorizzare i dati di sessione del client.
Relazioni con altre classi

- **MaaP::Client::ControllerModelView::ControllerClient::ControllerAutenticazione**
  Relazione entrante debole, interazione con il controller dell'Autenticazione.

5 Diagrammi di attività

6 Design Pattern

I Design Pattern sono soluzioni a problemi ricorrenti. Adottare i Design Pattern semplifica l'attività di progettazione, rende l'architettura più manutenibile e favorisce il riutilizzo del codice.
I design pattern possono essere suddivisi in:

- **Design pattern architetturali:** definiscono l'architettura dell'applicazione ad un livello più elevato;
- **Design pattern creazionali:** consentono di nascondere i costruttori delle classi, permettendo di creare oggetti senza conoscere la loro implementazione;
- **Design pattern strutturali:** consentono di riutilizzare classi pre-esistenti, fornendo un'interfaccia più adatta;
- **Design pattern comportamentali:** definiscono soluzioni per le interazioni tra oggetti.

Per una descrizione generale ed approfondita dei Design Pattern utilizzati si veda l'Appendice A.
Nella realizzazione del progetto MaaP si è deciso di implementare i seguenti Design Pattern:

6.1 Design Pattern architetturali

6.1.1 MVVM

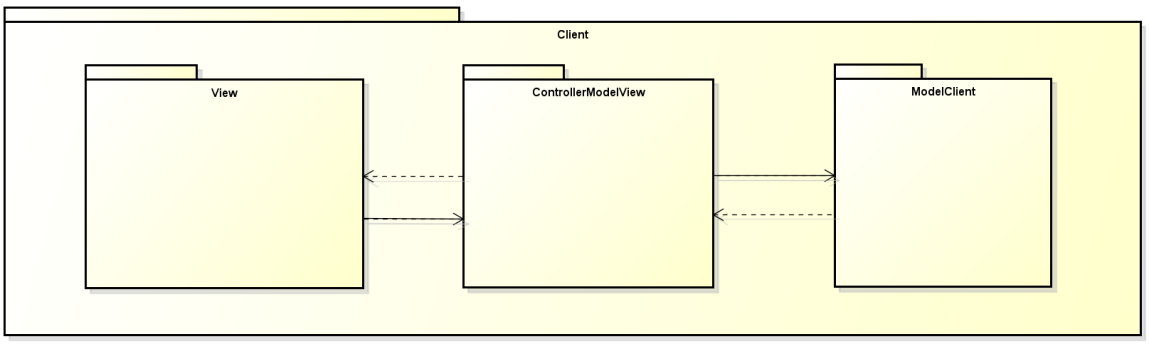

Figura 25: Applicazione di MVVM in MaaP

- **Scopo:** Il pattern MVVM è stato scelto per separare la logica dell'applicazione lato client dalla rappresentazione grafica;
- **Utilizzo:** Nel progetto MaaP la scelta di utilizzare AngularJS come base di partenza per l'applicazione lato client ha implicitamente comportato l'utilizzo del design pattern MVVM perché proprio di AngularJS.

6.1.2 Three-tier

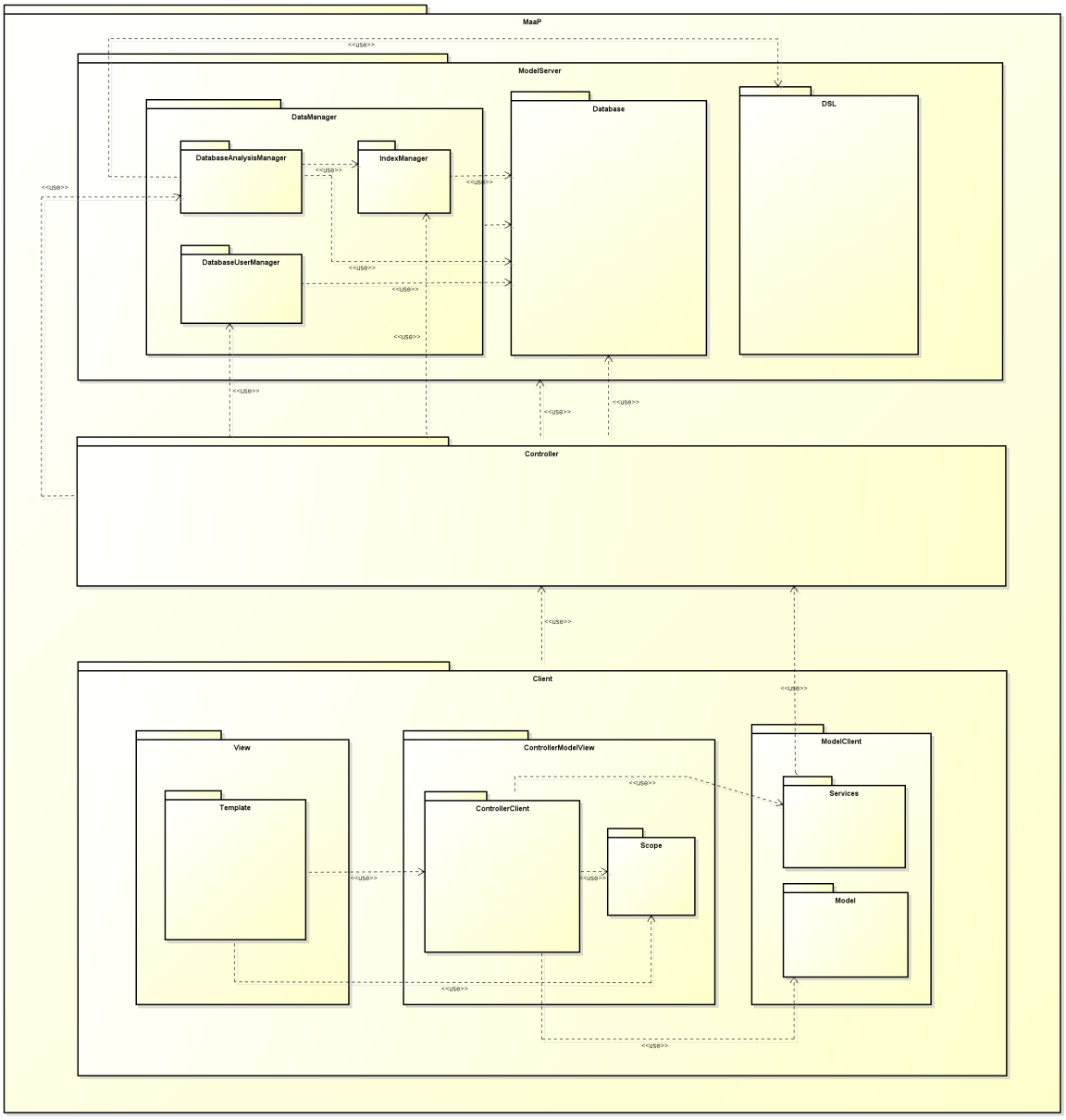

Figura 26: Applicazione di Three-tier in MaaP

- **Scopo:** Il pattern Three-tier è stato scelto per suddividere... ;
- **Utilizzo:** Nel progetto MaaP ...

6.2 Design Pattern creazionali

6.2.1 Singleton

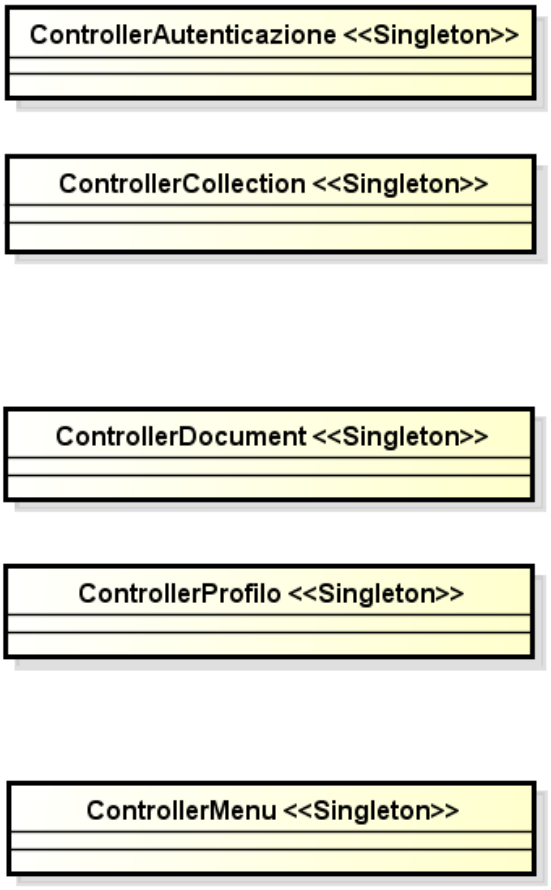

Figura 27: Applicazione di Singleton in MaaP

- **Scopo:** Viene usato il pattern Singleton per le classi che devono avere un'unica istanza durante l'esecuzione dell'applicazione;
- **Utilizzo:** Le classi che devono avere un'unica istanza sono i controller lato client.

6.3 Design Pattern comportamentali

6.3.1 Strategy

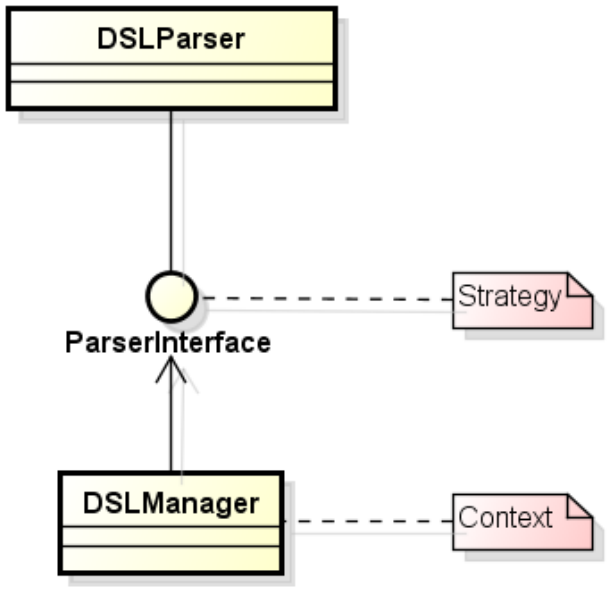

Figura 28: Applicazione di Strategy in MaaP

- **Scopo:** Il pattern Strategy viene usato per isolare più algoritmi che svolgono la stessa funzione dal codice che esegue la funzione;
- **Utilizzo:** In MaaP è stato usato gestire inizialmente un singolo algoritmo di parsing del file di descrizione, ma permetterà in futuro di aggiungere nuovi algoritmi di parsing differenziati senza modificare le classi che ne fanno uso.
  La concrete strategy attualmente presente è: DSLParser.

6.4 Design Pattern strutturali

6.4.1 Adapter

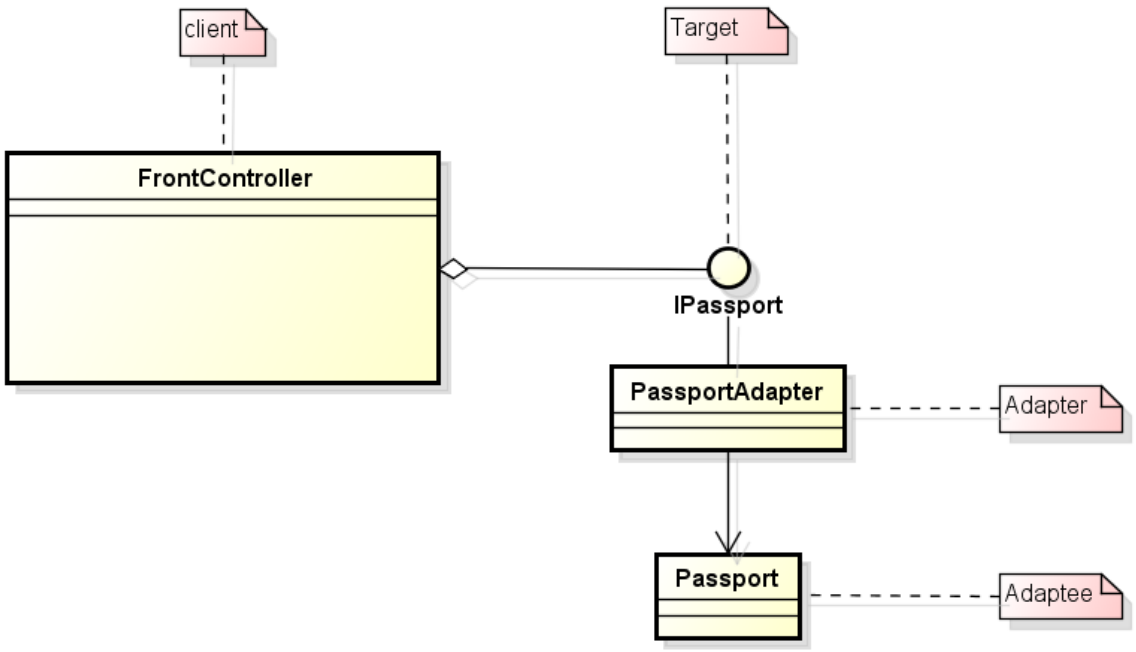

Figura 29: Applicazione di Adapter in MaaP

- **Scopo:** Il pattern Adapter viene utilizzato per adattare una classe riutilizzando un oggetto già esistente. Questo semplifica l'eventuale processo di sostituzione dell'oggetto esistente, creando un'interfaccia stabile per il resto dell'applicazione;
- **Utilizzo:** In MaaP è stato usato per adattare la classe Passport nel Controller. PassportAdapter adatta Passport.

6.4.2 Facade

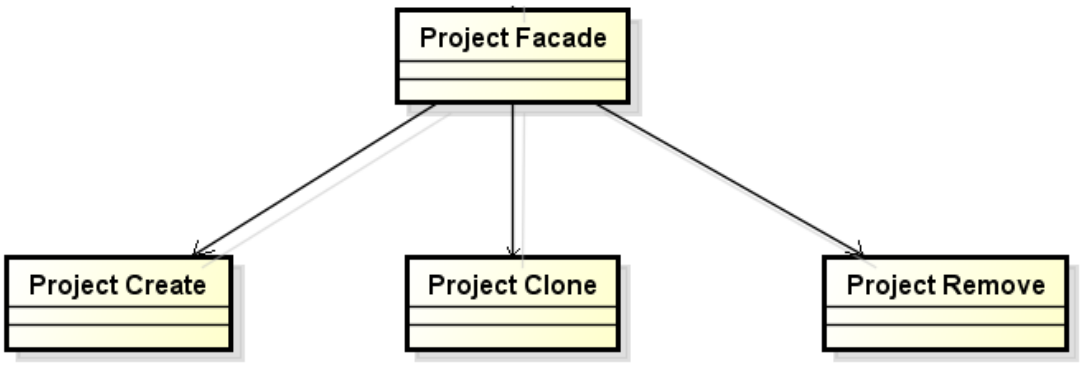

Figura 30: Applicazione di Facade in MaaP

- **Scopo:** Il pattern Facade viene usato per fornire un'interfaccia unica a più classi;
- **Utilizzo:** In MaaP, ProjectFacade è una Facade che presenta un'interfaccia per tutti gli oggetti gestiscono la creazione e/o modifica di un progetto:
  - ProjectCreate;
  - ProjectClone;
  - ProjectRemove.

7 Stime di fattibilità e di bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente per fornire una stima sulla fattibilità e di bisogno delle risorse.
L'analisi dell'architettura progettata ha permesso di constatare che le tecnologie che si è scelto di adottare risultano sufficientemente adeguate per la realizzazione del prodotto e riescono a ricoprire le esigenze progettuali.
Gli strumenti scelti sono conosciuti dalla maggioranza dei componenti del gruppo che si impegneranno comunque ad approfondire le loro conoscenze inerenti all'utilizzo degli stessi.
Gli strumenti utilizzati sono:

- **NodeJS:** per la realizzazione dell'applicazione web lato server;
- **AngularJS:** per la realizzazione dell'applicazione web lato client;
- **Mongoose:** per la comunicazione con il database MongoDB;
- **Express:** framework per NodeJS;
- **Passport:** modulo per la gestione dell'autenticazione utente;
- **PegJS:** generatore di parser javascript per il file di descrizione.

A Descrizione Design Pattern

A.1 Design Pattern architetturali

A.1.1 MVVM

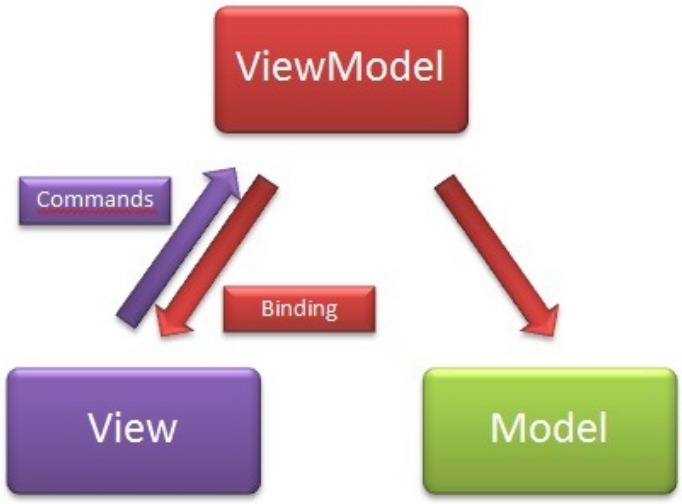

Figura 31: Diagramma del design pattern MVVM

- **Scopo:** Disaccoppiare le tre componenti seguenti:
  - Model: dati di business e regole di accesso;
  - View: rappresentazione grafica;
  - ViewModel: punto d'incontro tra View e Model. I dati ricevuti da quest'ultimo sono elaborati per essere presentati e passati alla View.
- **Motivazione:** Lo scopo di molte applicazioni è quello di recuperare dati e visualizzarli in maniera opportuna a seconda delle esigenze degli utenti. Poiché il flusso chiave di informazione avviene tra il dispositivo su cui sono memorizzati i dati e l'interfaccia utente, si è portati a legare insieme queste due parti per ridurre la quantità di codice e migliorare le performance dell'applicazione. Questo approccio, apparentemente naturale, presente alcuni problemi significativi; uno di questi è che l'interfaccia utente tende a cambiare più in fretta rispetto al sistema di memorizzazione dei dati. Un altro problema, che si ha nel mettere insieme i dati e l'interfaccia utente, è che le applicazioni aziendali tendono ad incorporare logica di business che va al di là della semplice trasmissione dei dati. C'è la necessità, quindi, di rendere modulari le funzionalità dell'interfaccia utente in maniera tale da poter facilmente modificare le singole parti. La soluzione a tutto ciò è costituita dal pattern Model-View-ViewModel (MVVM) che separa la modellazione del dominio, la presentazione e le azioni basate sugli input degli utenti all'interno di tre classi separate;
- **Applicabilità:** Il pattern MVVM può essere utilizzo nei seguenti casi:

- Quando si vuole trattare un gruppo di oggetti come un oggetto singolo;
- Quando si vuole disaccoppiare View e Model instaurando un *protocollo di sottoscrizione*$_G$ e notifica tra loro;
- Quando si vogliono agganciare più View ad un Model per fornire più rappresentazioni del Model stesso.

A.1.2 Three-Tier

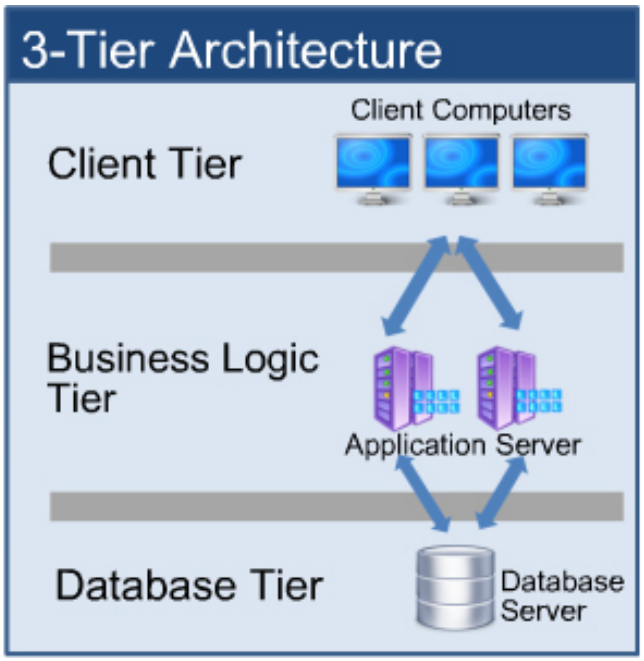

Figura 32: Diagramma del design pattern Three-tier

- **Scopo:** Disaccoppiare le tre componenti seguenti:
  - Client tier: strato che definisce l'interfaccia utente;
  - Business logic tier: strato che definisce la logica funzionale;
  - Database tier:strato che definisce la gestione dei dati persistenti.
- **Motivazione:** Il pattern Three-tier presenta diverse analogie con il pattern MVVM presentato precedentemente. Esso svolge un ruolo importante nella progettazione di applicazioni web specialmente basati su Web-service. Inoltre la suddivisione in moduli separati che seguono il paradigma client-server permette la sostituzione e/o modifica di un singolo modulo o strato indipendentemente dagli altri conferendo scalabilità e manutenibilità all'applicazione. I moduli possono essere inoltre distribuiti su diversi nodi di una rete anche eterogenea.
- **Applicabilità:** Il pattern Three-tier può essere utilizzo nei seguenti casi:
  - bla bla.

A.2 Design Pattern creazionali

A.2.1 Singleton

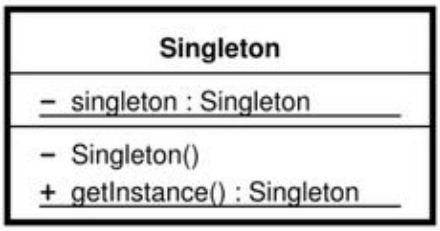

Figura 33: Diagramma del design pattern Singleton

- **Scopo:** Assicurare che una classe abbia solo un'istanza e fornire un punto d'accesso globale a tale istanza;
- **Motivazione:** L'uso di questo design pattern è importante poter assicurare che per alcune classi esista una sola istanza. Per far ciò la classe stessa ha la responsabilità di creare le proprie istanze, assicurare che nessun'altra istanza possa essere creata e fornire un modo semplice per accedere all'istanza;
- **Applicabilità:** Il pattern Singleton può essere utilizzato nei seguenti casi:
  - Quando deve esistere esattamente un'istanza di una classe e tale istanza deve essere resa accessibile ai client attraverso un punto di accesso noto a tutti gli utilizzatori;
  - Quando l'unica istanza deve poter essere estesa attraverso la definizione di sottoclassi e i client devono essere in grado di utilizzare le istanze estese senza dover modificare il proprio codice.

A.3 Design Pattern strutturali

A.3.1 Adapter

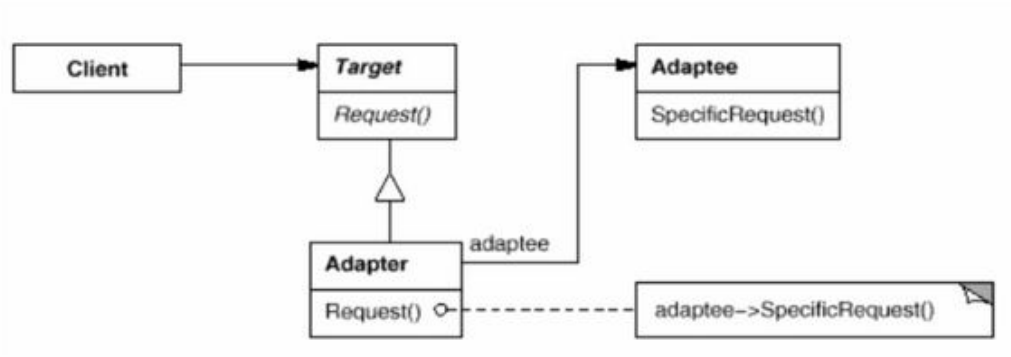

Figura 34: Diagramma del design pattern Adapter

- **Scopo:** Convertire l'interfaccia di una classe in un'altra interfaccia richiesta dal client. Consente a classi diverse di operare insieme quando ciò non sarebbe altrimenti possibile a causa di interfacce incompatibili;
- **Motivazione:** A volte una classe di supporto, che è stata progettata con obbiettivi di riuso, non può essere riusata semplicemente perché la sua interfaccia non è compatibile con l'interfaccia richiesta da un'applicazione;
- **Applicabilità:** Il pattern Adapter può essere utilizzato nei seguenti casi:
  - Quando si vuole usare una classe esistente, ma la sua interfaccia non è compatibile con quella desiderata;
  - Quando si vuole creare una classe riusabile in grado di cooperare con classi non correlate o impreviste, cioè con classi che non necessariamente hanno interfacce compatibili;
  - Per gli oggetti adapter quando si devono utilizzare diverse sottoclassi esistenti, ma non è pratico adattare la loro interfaccia creando una sottoclasse per ciascuna di esse.

A.3.2 Facade

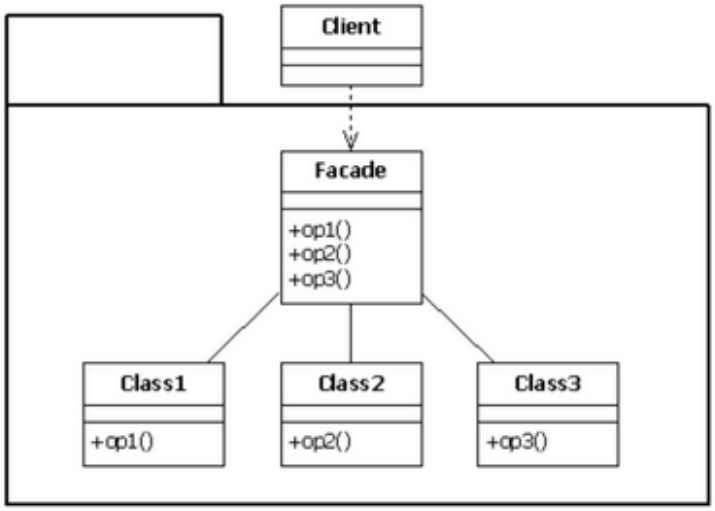

Figura 35: Diagramma del design pattern Facade

- **Scopo:** Fornire un'interfaccia unificata per un insieme di interfacce presenti in un sottosistema. Definisce un'interfaccia di livello più alto che rende il sottosistema più semplice da utilizzare;
- **Motivazione:** Suddividere un sistema in sottosistemi aiuta a ridurre la complessità. Un obbiettivo comune di progettazione è la minimizzazione delle comunicazioni e delle dipendenze fra i diversi sottosistemi. Un modo per raggiungere questo obbiettivo è introdurre un oggetto facade, che fornisce un'interfaccia unica e semplificata per accedere alle funzionalità offerte da un sottosistema;
- **Applicabilità:** Il pattern Facade può essere utilizzato nei seguenti casi:
  - Quando si vuole fornire un'interfaccia semplice a un sottosistema complesso poiché fornisce una vista semplice di base su un sottosistema che si rivela essere sufficiente per la maggior parte dei client;
  - Nei casi in cui si cono molte dipendenze fra i client e le classi che implementano un'astrazione in quanto si disaccoppia il sottosistema dai client e dagli altri sistemi, promuovendo portabilità ed indipendenza dei sottosistemi;
  - Quando si vogliono organizzare i sottosistemi in una struttura a livelli.

A.4 Design Pattern comportamentali

A.4.1 Strategy

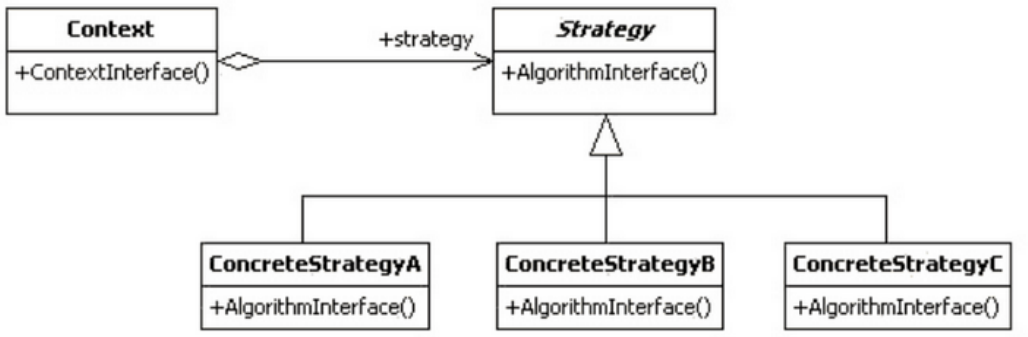

Figura 36: Diagramma del design pattern Strategy

- **Scopo:** Definire una famiglia di algoritmi, incapsularli e renderli intercambiabili. Permette agli algoritmi di variare indipendentemente dai client che ne fanno uso;
- **Motivazione:** Esistono molti algoritmi per risolvere un problema. Codificare statisticamente ognuno di questi algoritmi nelle classi che ne fanno richiesta non è auspicabile per svariati motivi. Si possono evitare questi problemi definendo delle classi che incapsulano svariati algoritmi chiamati Strategy;
- **Applicabilità:** Il pattern Strategy può essere utilizzato nei seguenti casi:
  - Molte classi correlate differiscono fra loro solo per il comportamento;
  - Sono necessarie più varianti di un algoritmo;
  - Un algoritmo usa una struttura dati che non dovrebbe essere resa nota ai client;
  - Una classe definisce molti comportamenti che compaiono all'interno delle scelte condizionali multiple.