Tesina di Applicazioni Telematiche

Piscopo Davide - Mat. M63/000472 Primativo Andrea - Mat. M63/000440
Silvestro Rolando - Mat. M63/000482

12 maggio 2017

Indice

Introduzione

Il sistema progettato ha lo scopo di permettere ad un utente di poter controllare da remoto, tramite un' applicazione mobile, i device intelligenti presenti in casa sua.

A questo scopo si è sviluppata una web application convergente, che da un lato è capace di comunicare con i device installati nell'abitazione, e dall'altro espone metodi REST che permettono l'interazione web con un'applicazione, anch'essa appositamente sviluppata.

Per quanto riguarda le tecnologie utilizzate abbiamo:

- JavaEE per lo sviluppo lato server. In particolare si è utilizzato **Jetty** come web server, e la libreria **Leshan** per la parte di comunicazione con gli smart object conformi allo standard **IPSO**.
- Ionic per lo sviluppo dell'applicazione mobile. Si noti che il deployment dell'app è stato effettuato solo per piattaforme android, sebbene ionic permetta di deployare facilmente anche su piattaorme iOS.

Capitolo 1

Requisiti

1.1 Requisiti Funzionali

La progettazione è partita dall'analisi dei requisiti, riportati di seguito:

Requisiti funzionali utente

1. L'utente deve poter monitorare lo stato di tutti i device installati nell'abtazione.
2. L'utente deve poter essere in grado di comandare i singoli device.
3. L'utente deve poter impostare delle regole che permettano di automatizzare il comportamento di un sottoinsieme dei device installati.

Requisiti funzionali di sistema

1. Il sistema deve essere in grado di accettare la registrazione di nuovi device conformi allo standard IPSO.
2. Il sistema deve essere in grado di caricare regole precedentemente salvate su un supporto persistente.
3. Il sistema deve poter essere in grado di ricevere notifiche dai device ad ogni cambio di un valore di interesse, in seguito ad opportuna richiesta.
4. Il sistema deve notificare l'utente in caso di esecuzione di una regola impostata.

Capitolo 2

Analisi del sistema

2.1 Diagramma dei casi d'uso

Di seguito il diagramma dei casi d'uso. Si noti che nelle fasi successive ci si è concentrati sui casi d'uso **Imposta regola** e **Notifica nuovo valore** in quanto sono i casi più significativi e su cui si è concentrato maggiormente lo sviluppo.

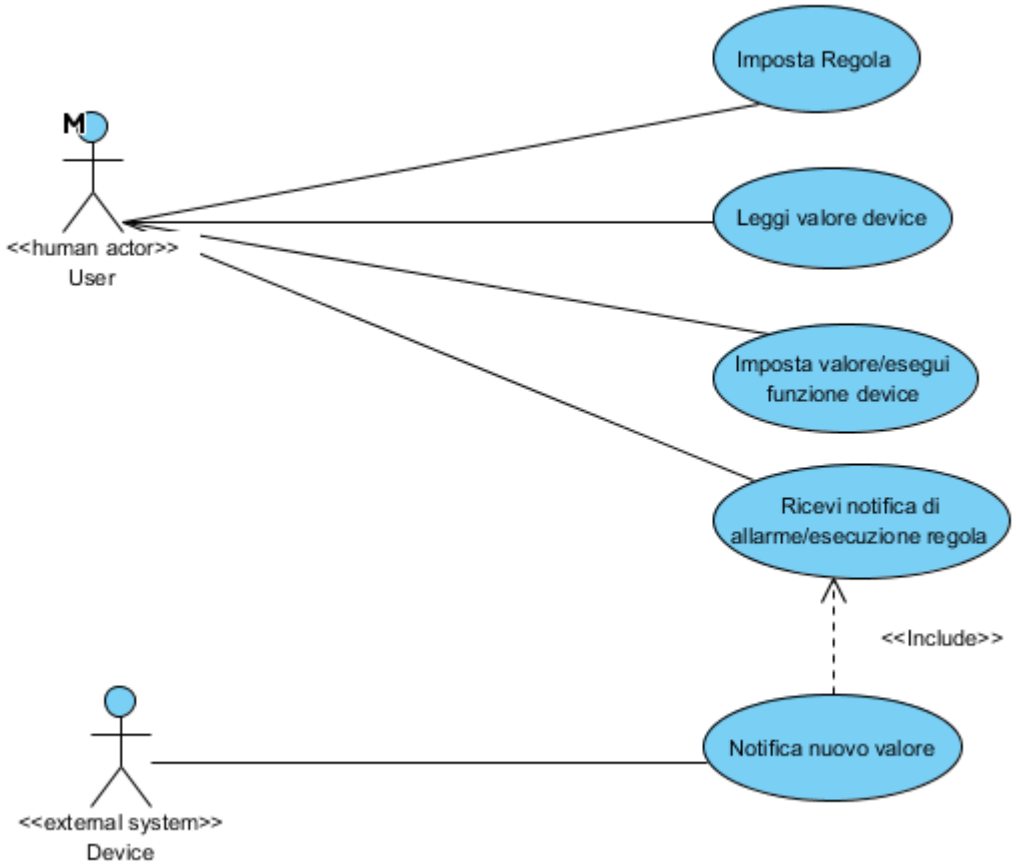

Figura 2.1: Diagramma dei Casi d'uso

2.1.1 Descrizione del caso d'uso UC1 [Imposta regola]

Questo caso d'uso descrive le azioni da svolgere al fine di impostare e attivare una nuova regola di automazione o di allarme.

Attori coinvolti: Utente, Device

Scenario Principale:

1. L'utente, tramite l'interfaccia grafica, imposta i parametri della regola (risorse interessate, operatori logici, valore target, eventuale azione da eseguire) e li invia al server
2. Il server valida e salva tali parametri
3. Per ogni device interessato dalla regola appena salvata, il server provvede a sottoscriversi agli aggiornamenti dei valori di interesse per il device in oggetto.

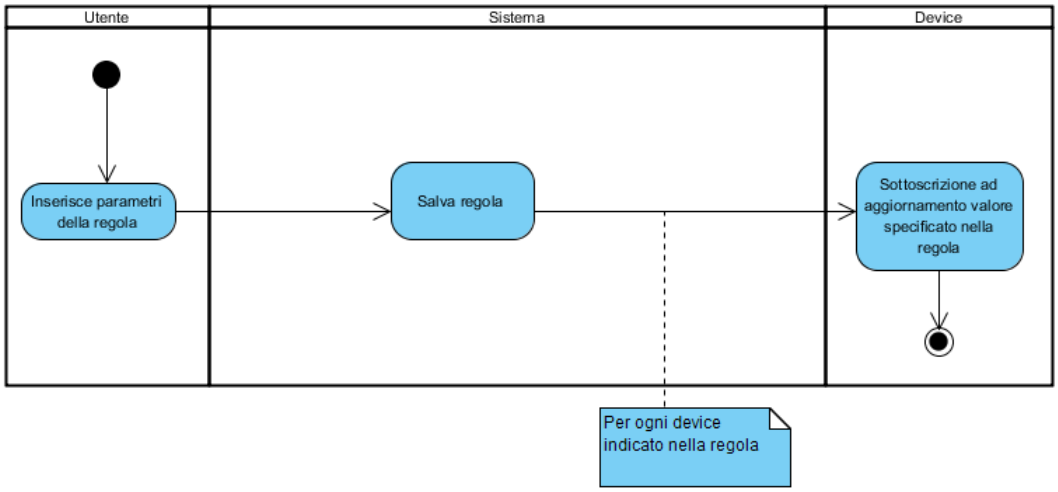

Figura 2.2: Activity Diagram UC1

2.1.2 Descrizione del caso d'uso UC2 [Notifica nuovo valore]

UC2 descrive il comportamento del sistema alla ricezione della notifica, da parte di un device, di un nuovo valore.

Attori coinvolti: Device, Utente

Precondizioni: Deve precedentemente essere stata inviata una richiesta di sottoscrizione al device (o in seguito all'impostazione di una regola che interessi tale device, o in seguito alla richiesta di monitoring dell'utente).

Scenario Principale:

1. Il device invia la notifica, contenente il nuovo valore registrato
2. Il server provvede innanzitutto ad aggiornare la vista dell'interfaccia grafica (nel caso in cui l'utente abbia fatto richiesta di monitoring per quel valore di quel device)

3. **Se** esistono regole che interessano quel device allora:
4. Le regole influenzate dal device vengono rivalutate sulla base del nuovo valore ricevuto.
5. **Se** una o più regole sono triggerate dal nuovo valore allora:
6. Viene eseguita l'eventuale azione prevista dalla regola
7. L'utente viene notificato.

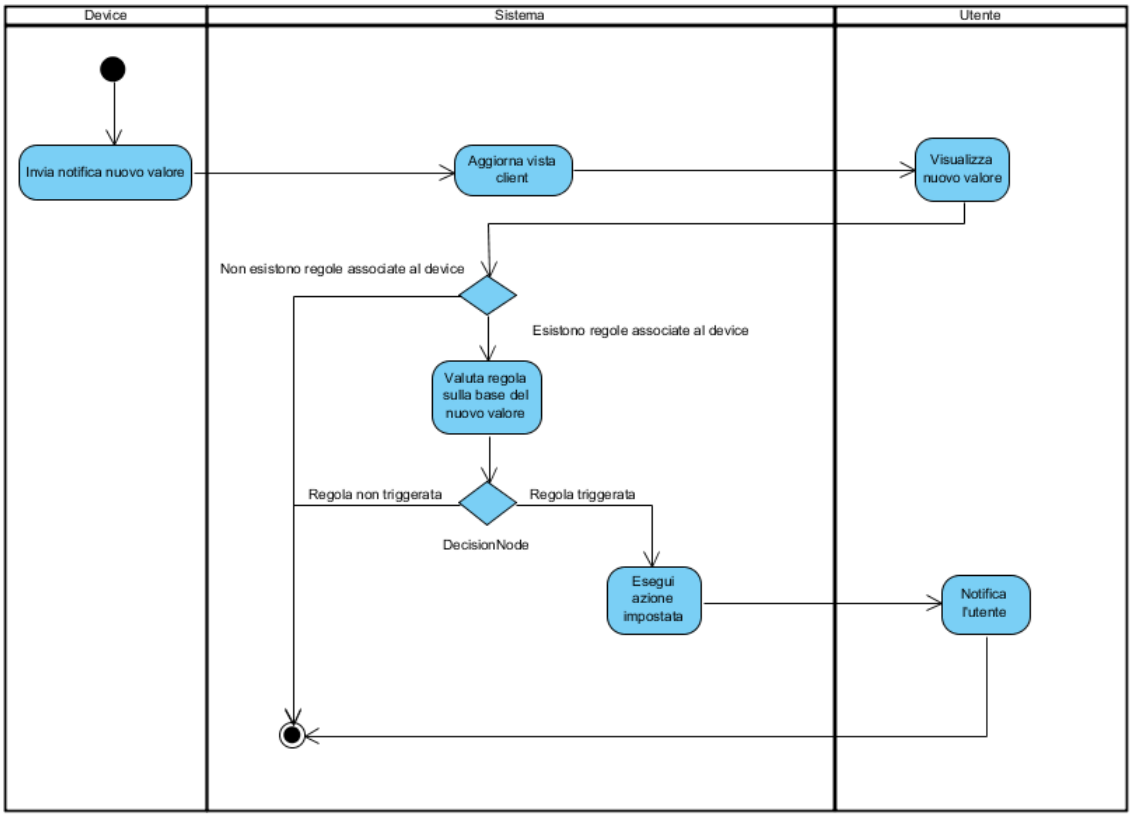

Figura 2.3: Activity Diagram UC2

2.2 Modello del dominio

Di seguito è mostrato il modello di dominio che descrive le varie entità individuate come facenti parte del sistema, nonchè le loro relazioni.

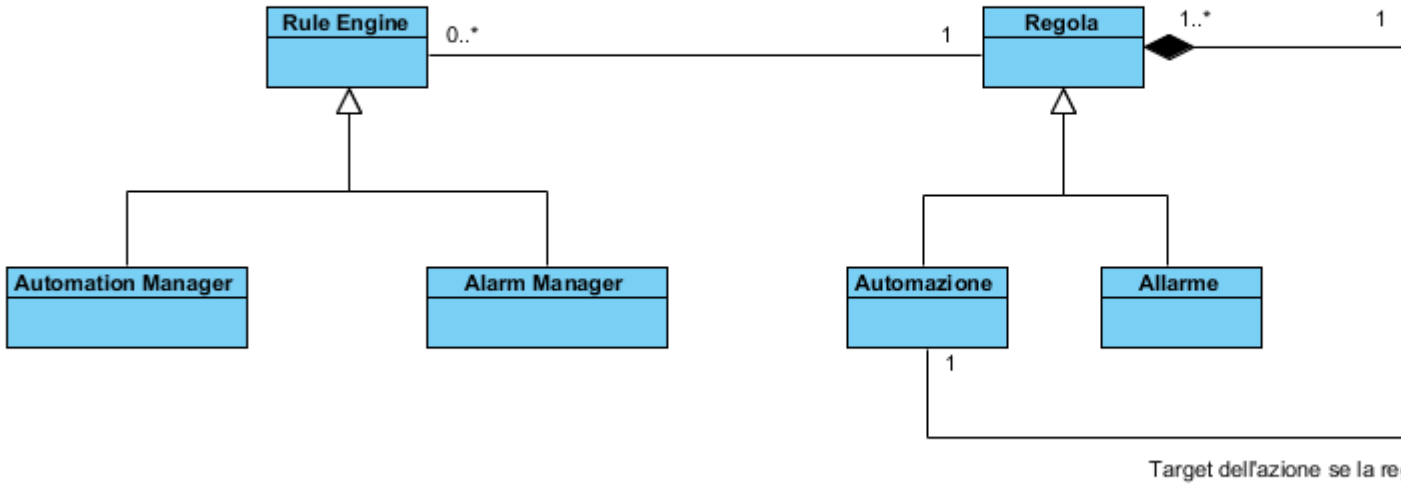

Figura 2.4: Modello di dominio

2.2.1 Rationale

Per quanto riguarda la parte delle **Regole** si ha appunto l'entità regola che si specializza nelle classi **Automazione** e **Allarme**. Queste classi servono a modellare il concetto di regole di automazione e di allarme che l'utente può impostare.

La regola è composta da **Condizioni.** Una condizione specifica la sorgente, in termini di **Risorsa** (valore associato ad un **Device**), un valore target ed un operatore logico usato per il confonto tra valore sorgente e valore target.

Inoltre l'entità **Regola** specifica un azione, ancora una volta espressa in termini di **Risorsa** (questa volta intesa come azione associata ad un **Device**).

Infine si è individuata l'entità **Rule Engine** che rappresenta colui che gestisce le **Regole**. Esso si occuperà di tenere traccia delle regole impostate, nonchè di valutarle. Le specializzazioni di questa classe, **Automation Manager** e **Alarm Manager**, implementano l'azione da intraprendere in seguito al trigger della regola.

2.3 Diagrammi dinamici

In questa sezione si presentano dei diagrammi dinamici, relativi ai casi d'uso indicati, che mostrano l'interazine tre le componenti del sistema.

2.3.1 Communication Diagram [Imposta regola]

Il seguente communication diagram mostra le interazioni necessarie ad impostare una regola.

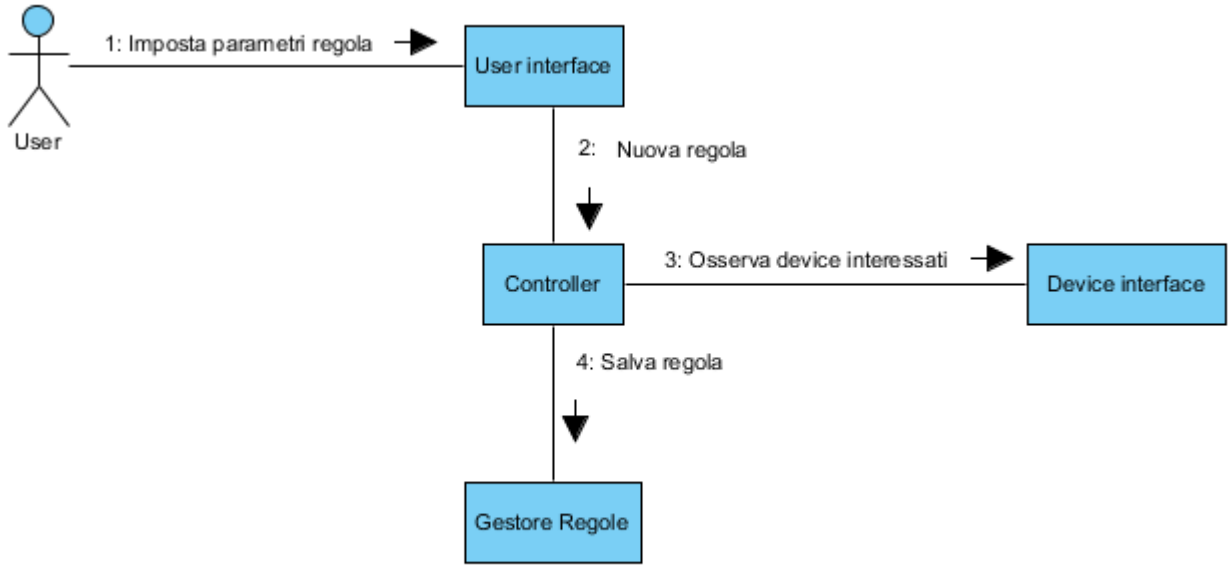

Figura 2.5: Communication Diagram UC1

2.3.2 Communication Diagram [Notifica nuovo valore]

Il seguente communication diagram mostra le interazioni che sono poste in atto al momento della ricezione di una notifica da parte di un device.

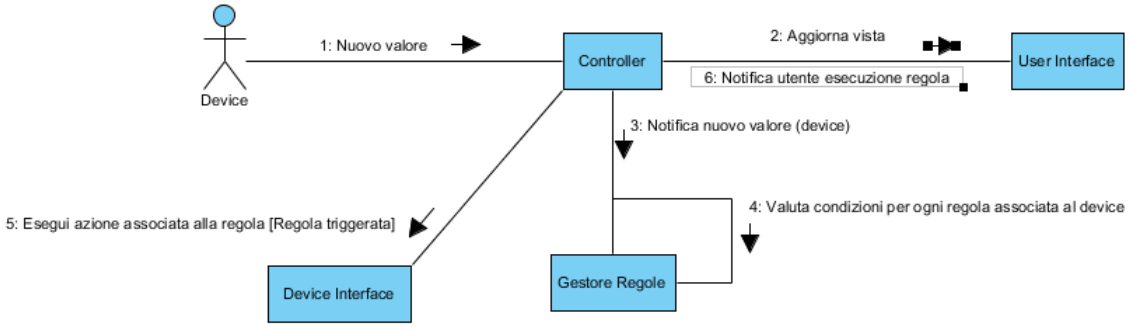

Figura 2.6: Communication Diagram UC2

Capitolo 3

Diagrammi architetturali

3.1 Overview del Sistema

Di seguito è rappresentata informalmente l'architettura del sistema.

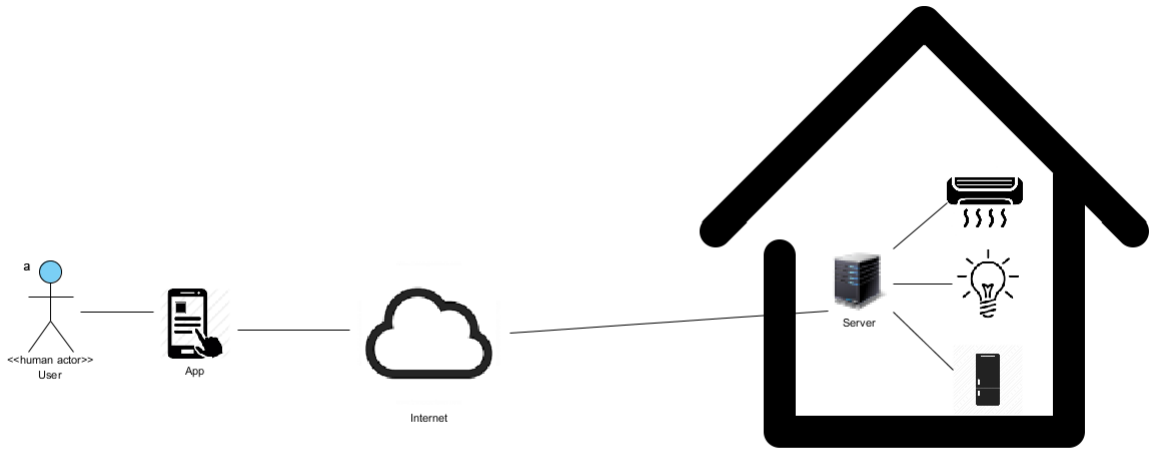

Figura 3.1: Vista Architetturale: Panoramica

3.1.1 Descrizione

Le componenti del sistema implementato sono due.

- **Il server**, che sarà installato all'interno dell'abitazione e si occuperà della comunicazione con i device presenti (COAP) e della comunicazione con l'app mobile.
- **L'app mobile**, che fornisce un'interfaccia per l'utente e permette di interagire da remoto (HTTP) con il server.

3.2 Vista architetturale - alto livello

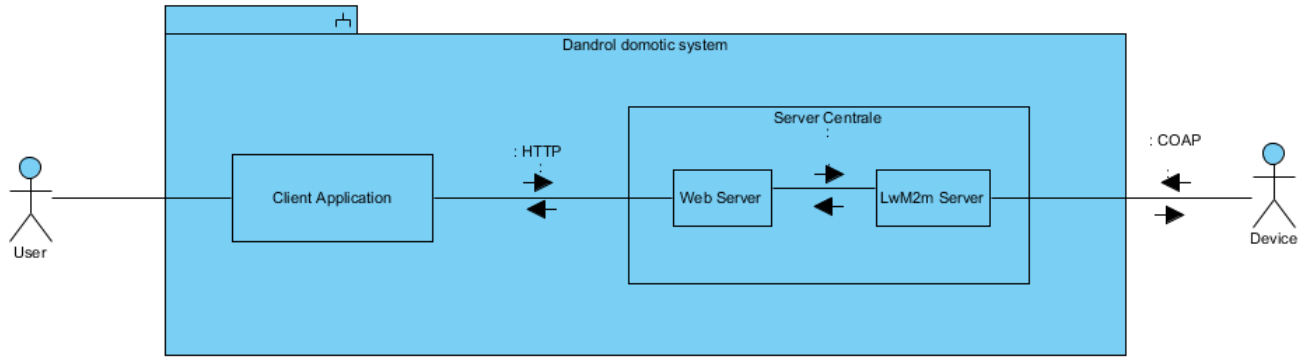

Figura 3.2: Vista Architetturale: Alto Livello

3.2.1 Descrizione

Piu nel dettaglio, i componenti fondamentali del sistema sono i seguenti:

- **Client application**, ossia l'app mobile.
- **Server Centrale**, composto fondamentalmente da:
  - **Web Server**: è il server che comunica con l'applicazione tramite API RESTful ed implementa la logica utente tramite delle apposite servlet, che si occupano di gestire le richieste dell'utente.
  - **LightWeightMachine2Machine Server**: è il server che implementa lo stack **OMA LWM2M** e **COAP** per la comunicazione coi device.

3.3 Module View - Server Centrale

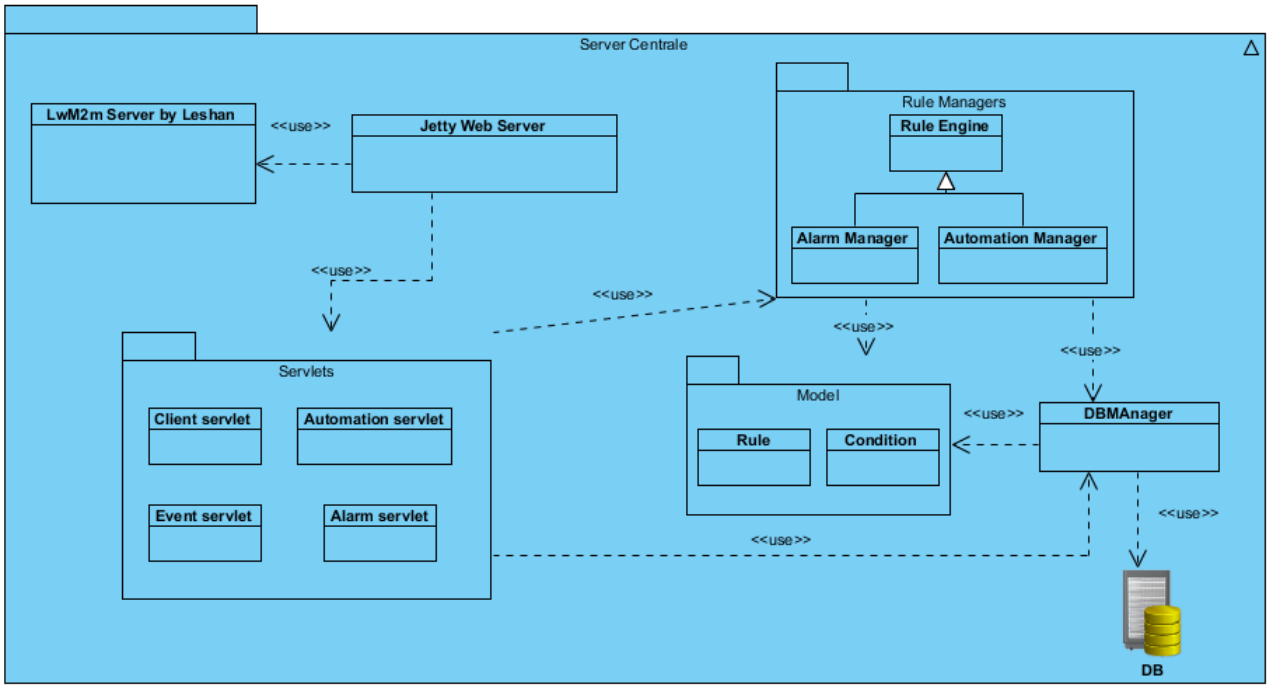

Figura 3.3: Diagramma dei Moduli

3.3.1 Descrizione

Scendendo nel dettaglio, nel server centrale sono stati identificati i seguenti moduli software:

- **Jetty web server**: implementazione di web server.
- **LwM2m Server by Leshan**: è un server Lwm2m implementato utilizzando la libreria **Leshan** (vedi appendice A)
- **Servlets**: in questo modulo sono contenute le servlet che si occupano della gestione delle richieste da parte del client, nonchè dell'invio di push notification da parte del web server verso il client.
- **Model**: modulo che contiene il modello dei dati, ossia Regole e Condizioni.
- **Rule Managers**: contiene la classe astratta **Rule Engine** e le sue specializzazioni che si occupano di gestire le regole salvate persistentemente e quelle impostate a runtime dall'utente
- **DBManager**: classe che astrae l'accesso al database
- **DB:** Mysql database per la conservazione di regole.

3.4 Module View - MobileApp

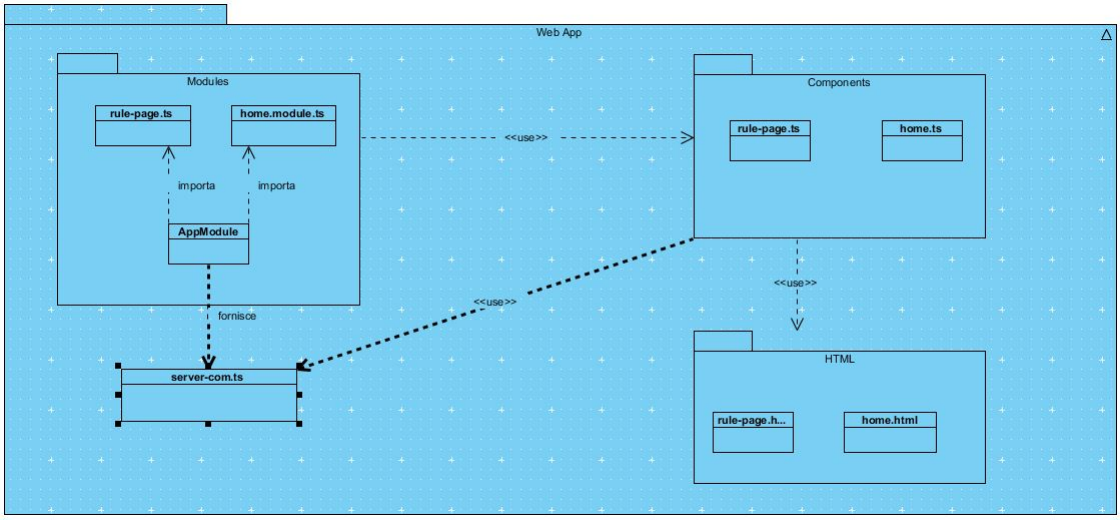

Figura 3.4: Diagramma dei Moduli

3.4.1 Descrizione

Le pagine navigabili dell'applicazione sono:

- **home.html**: utilizza il servizio **ServerCom** per richiedere tramite Http Get la lista degli endpoint connessi
- **device-detail.html:** visualizza i dettagli dell'endpoint selezionato in **home.html**
- **resource-detail.html:** visualizza le risorse associate ad un oggetto IPSO associato ad un dispositivo selezionato tramite **device-detail.html** e la cui descrizione viene richiesta al servizio **ServerCom**
- **rule-page.html:** utilizza il servizio **ServerCom** per richiedere tramite HttpGet la lista delle regole attive sul server, permette tramite un bottone di inviare regole nuove al server tramite **ServerCom** con il metodo Http Post
- **rule-detail.html:** visualizza i dettagli relativi ad una singola regola, selezionata da **rule-page.html** e permette di navigare verso **device-detail.html** per visualizzare i dettagli delle risorse coinvolte

Ad ognuna di queste pagine html sono associati un componente ed un modulo angular 2.

Il componente radice app.component.ts si preoccupa di realizzare il menu da cui è possibile navigare verso le due pagine **home.html** e **rule-page.html**

3.5 Vista Architetturale - Connettori

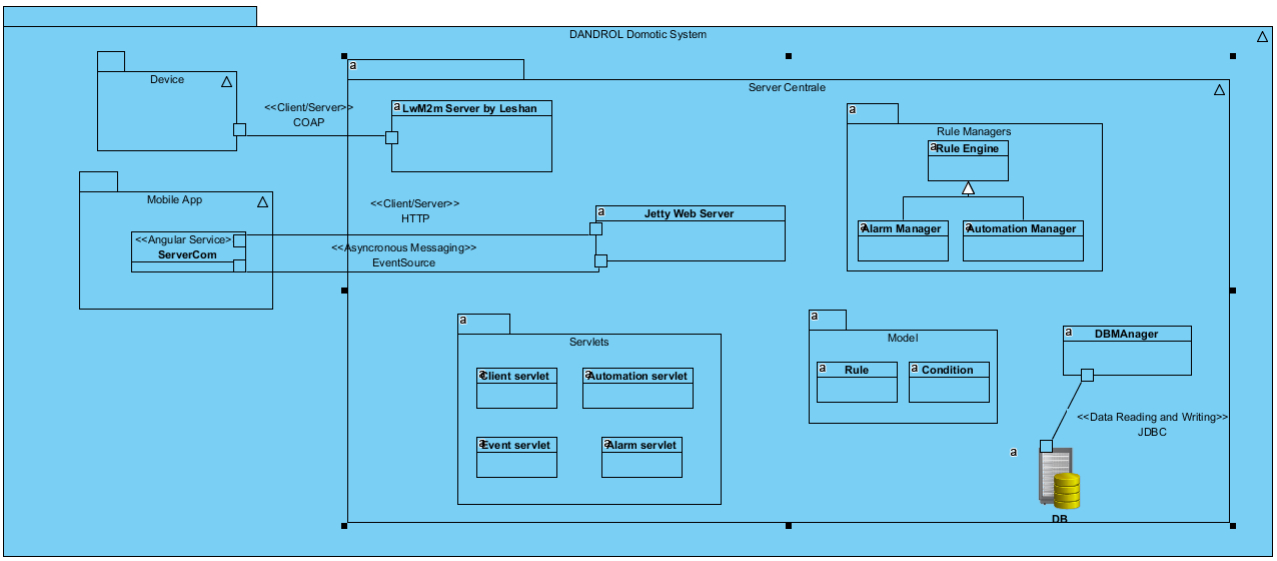

Figura 3.5: Diagramma dei Moduli

3.5.1 Descrizione

Questa vista a componenti e connettori indica come i diversi componenti dell'architettura comunicano tra di loro e quali tipi di connettori utilizzano per tale comunicazione.

In particolare sono stati individuati tre pattern di comunicazione:

- **Call/Return**: tale tipo di comunicazione (sostanzialmente chiamata a funzione, quindi invocazione di metodi) è utilizzato dai componenti intra-server centrale per comunicare tra di loro.
- **Client/Server**: utilizzato per la comunicazione tra :
  - l'app mobile e il web server. Implementato tramite protocollo **HTTP.**
  - i device e il lwm2m server. Implementato tramite protocollo **COAP.**
- **Asyncronous Messaging**: Il web server utilizza questo pattern per inviare notifiche al client in seguito a qualche evento particolare. Imlementato tramite la specifica **Event Source. NB: è prevista la migrazione a WebSocket dati i problemi riscontrati con EventSource.**
- **Data Reading and Writing**: Connettore utilizzato da un componente Data Accessor per richiedere i dati forniti da un componente di tipo Repository . Il nome del connettore specifica la tecnologia utilizzata. Nel caso specifico, la comunicazione col DB avviene tramite JDBC.