Software Design Description

MapYou

Giuseppe Fusco
matricola: 399000091
email: giuseppe.fusco.666@gmail.com

Giovanni Di Blasio
matricola: 399000090
email: diblasiogiovanni48@gmail.com

Indice

1. Introduzione

1.1. Obiettivo

L'obiettivo del documento è fornire una descrizione dettagliata dei requisiti per il sistema software "MapYou".

1.2. Scopo

Mapyou è un'applicazione/sistema mobile su piattaforma Android che consente ad un insieme di utenti, appartenenti ad una community (**Mapme**), di raggiungere una location, tramite un percorso, e "guidarli" al raggiungimento della stessa.
L'utilizzo del sistema è consentito solo tramite registrazione.
Mapyou fornisce agli utenti iscritti ad una Mapme un servizio di visione comune degli spostamenti effettuati dagli altri utenti per facilitarne l'indirizzamento verso la location di arrivo.
Tale servizio include:

- una sezione interattiva (**RTI**: real-time interaction):
    - visualizzazione real-time delle posizioni degli utenti iscritti alla Mapme;
    - visualizzazione del percorso della Mapme;
- lo scambio di messaggi tra utenti.

Per fornire tale servizio il sistema necessita di una connessione sia Internet che GPS per ottenere le informazioni geografiche degli utenti.
Lo scambio di messaggi tra utenti è realizzato mediante un sistema di chat interno all'applicazione mobile.
Tutte le informazioni del sistema sono mantenute in un database locato su un server.
L'applicazione potrà essere scaricata da qualsiasi application store o servizi simili.

1.3. Definizioni, acronimi e abbreviazioni

Termine	Definizione
Mapyou	Nome del sistema
Utente	Qualcuno che interagisce con l'applicazione mobile
Location	Locazione fisica di un punto di arrivo/destinazione
Mapper	Utente che decide/sceglie/promuove una location di destinazione
Mapme	Comunità/gruppo/insieme di utenti
Mapped	Utente che fa parte di una mapme
RTI	Modalità di interazione real-time con il sistema
GPS	Global Positioning System
GPS-Navigator	Software installato sul dispositivo mobile che fornisce connessioni GPS
Notification-System	Sistema software di terze parti che permette l'invio di notifiche verso dispositivi mobile
Application Store	Applicazione installata sul dispositivo mobile che aiuta gli utenti a cercare nuove applicazioni compatibili con il sistema operativo del dispositivo ed effettuarne il download da Internet

2. Architettura software attuale

Il sistema è stato implementato dal nulla in quanto nessun sistema avente le stesse caratteristiche esiste in precedenza, per cui questa è la fase di "Greenfield Engineering" e la raccolta dei requisiti è stata effettuata colloquiando direttamente con il cliente, cercando di estrarre i requisiti fondamentali dalle sue richieste.

3. Architettura software proposta

3.1. Overview

Il sistema è costituito da un nucleo centrale contenente l'archivio di tutte le informazioni sulle MapMe ed sugli utenti. Gli utenti possono interagire con il sistema centrale da qualsiasi postazione. Le richieste devo essere sempre fatte al sistema centrale. L'architettura più idonea per il sistema è di tipo Client/Server.

Il sistema permette la creazione, la gestione delle partecipazione alle MapMe, lo scambio di messaggi privati tra i vari utenti. Quando un utente richiederà un determinato servizio, la richiesta sarà inoltrata al Server, il quale tramite un'interazione con il DBMS, preleva l'informazione, la quale sarà inoltrata all'utente che ne ha fatto richiesta.

3.2. Decomposizione del sistema in sottosistemi

Il sistema software è stato decomposto in sottosistemi indipendenti. I moduli dei sottosistemi sono stati progettati evitando un forte accoppiamento tra di essi ed incrementandone la coesione. La progettazione dei vari sottosistemi e delle comunicazioni tra moduli è stata effettuata adottando una serie di pattern comportamentali, strutturali e creazionali, per agevolare lo sviluppo di nuovi moduli, la portabilità del sistema, i vari interventi di manutenzione e aggiornamento.

Di seguito è illustrato il diagramma di deployment con i componenti suddivisi per ogni nodo.

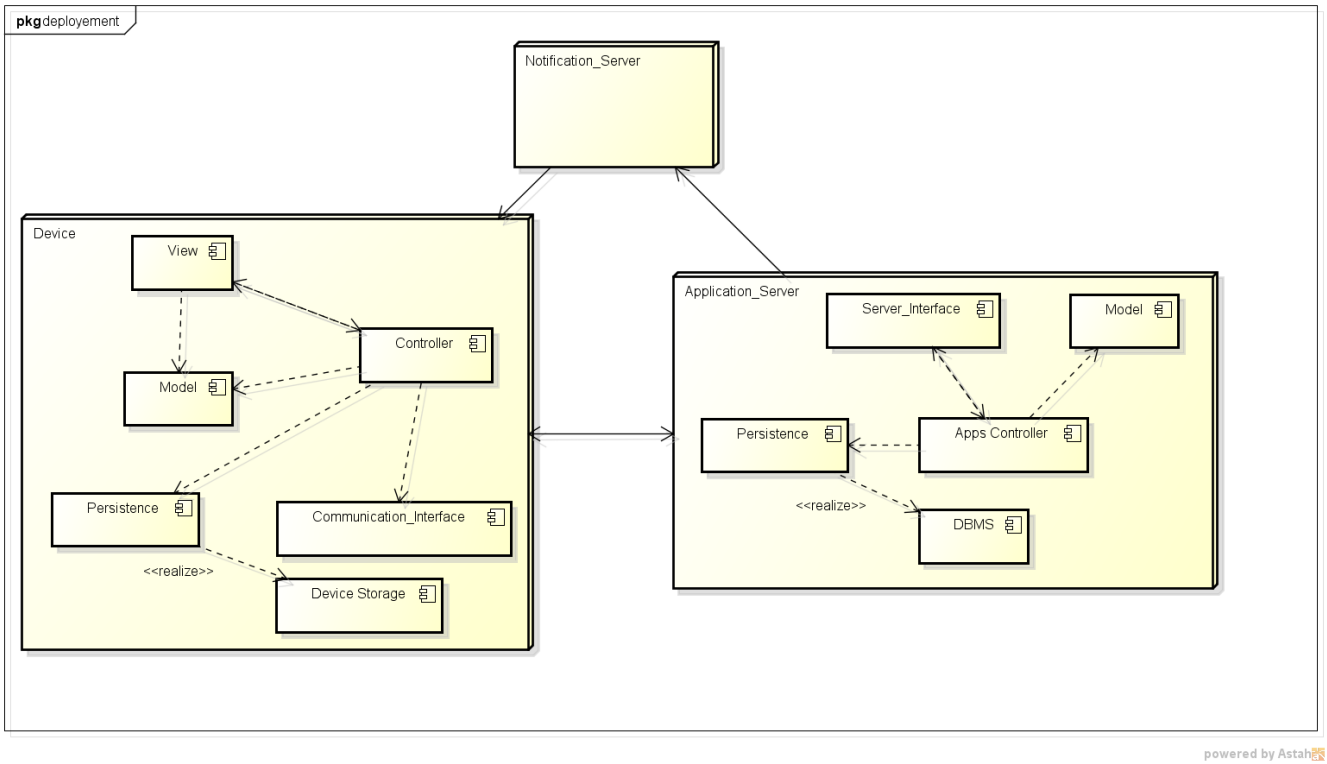

Fig. 1 Deployment diagram.

I sottosistemi fondamentali possono essere divisi su due macro livelli, il primo riguarda i sottosistemi lato Client ed il secondo riguarda i sottosistemi lato Server, i quali comunicano mediante un sottosistema di comunicazione. Un altro componente chiave per la comunicazione tra i dispositivi è il Notification System: è un sistema software di terze parti che permette l'invio di notifiche verso dispositivi mobile.

3.2.1. Sottosistema Client

Il sottosistema Client è stato progettato utilizzando il pattern architetturale MVC (Model-View-Controller).

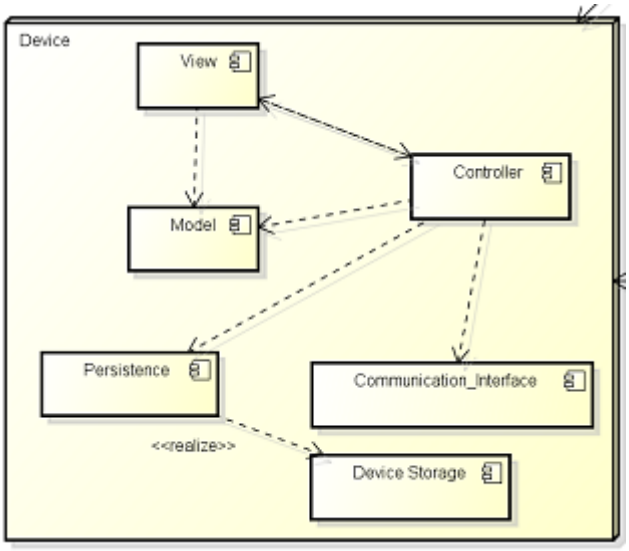

Fig. 2 Sottosistema client.

- **Model:** componente che gestisce gli oggetti di dominio del sistema.

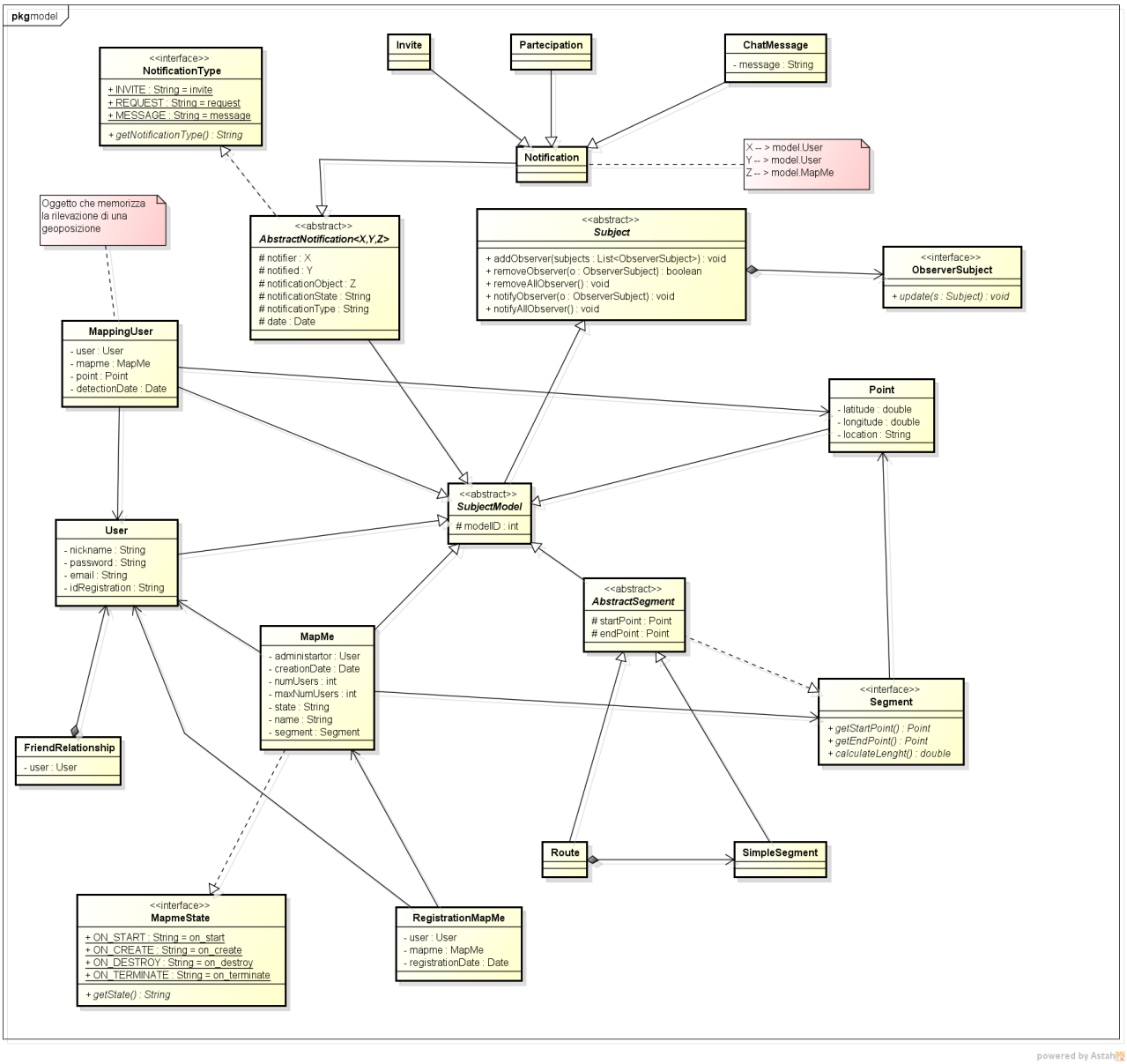

Fig. 3 Class diagram di dominio.

I pattern utilizzati nella progettazione sono i seguenti:

- **Composite:** è stato utilizzato per la gestione dei segmenti di un percorso, in quanto un generico percorso è composto da un solo segmento, il quale può, o non, essere a sua volta decomposto in molteplici segmenti.

- **Abstract Factory:** è stato utilizzato per la gestione dei messaggi di notifica (Invito – Richiesta di partecipazione – Messaggi tra utenti).

- **Observer:** è stato utilizzato per gli aggiornamenti degli oggetti del componente View, che visualizzano le informazioni inerenti al componente Model.

- **Persistence:** componente che gestisce la memorizzazione delle informazioni degli utenti e del sistema.

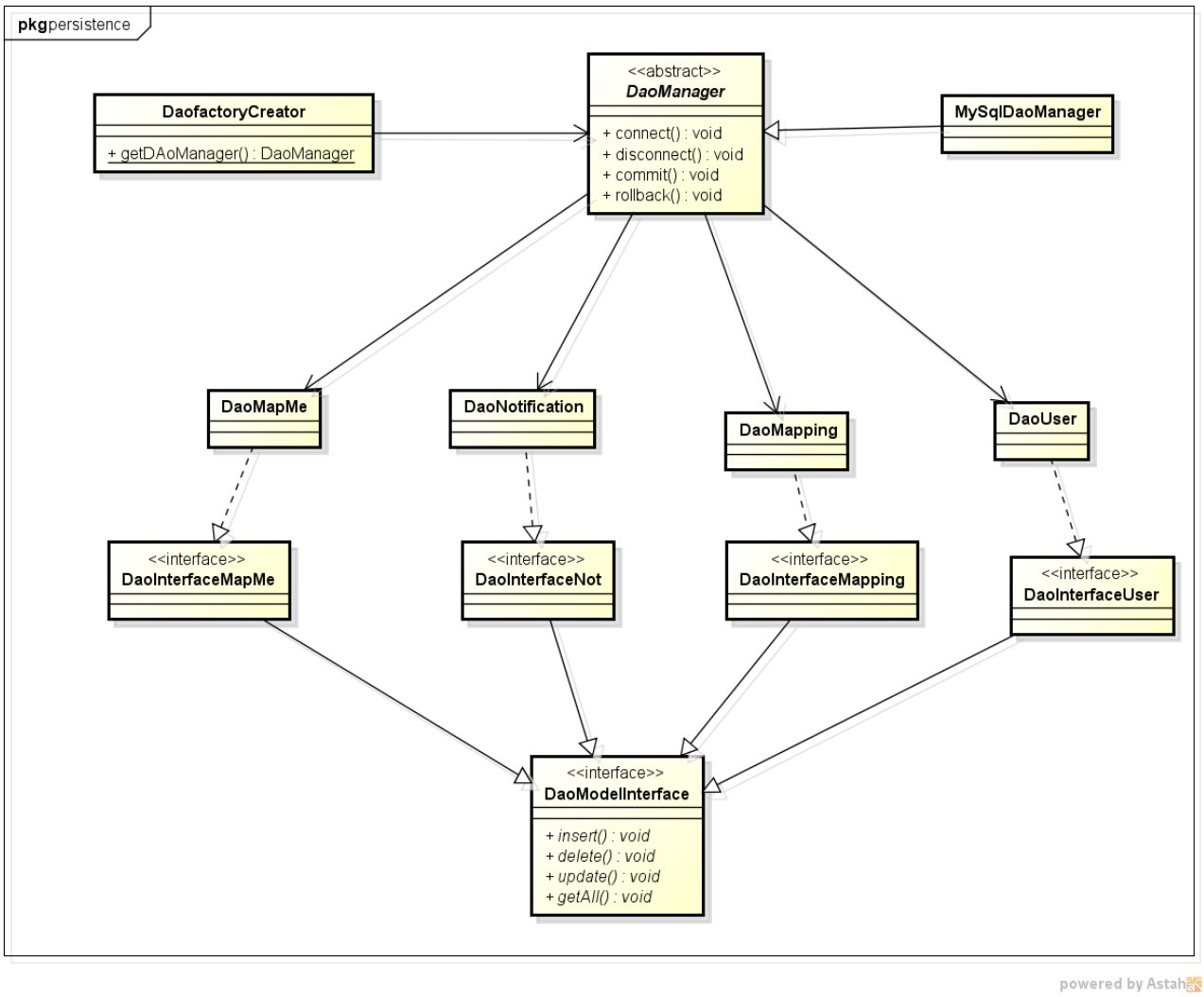

Fig. 4 Class diagram della persistenza.

Tale componente è stato progettato utilizzando il pattern strutturale DAO (Data Access Object), per agevolare lo sviluppatore nell'integrazione di diverse sorgenti.

- **Controller:** fornisce un'interfaccia per nascondere la logica del sistema al componente View.

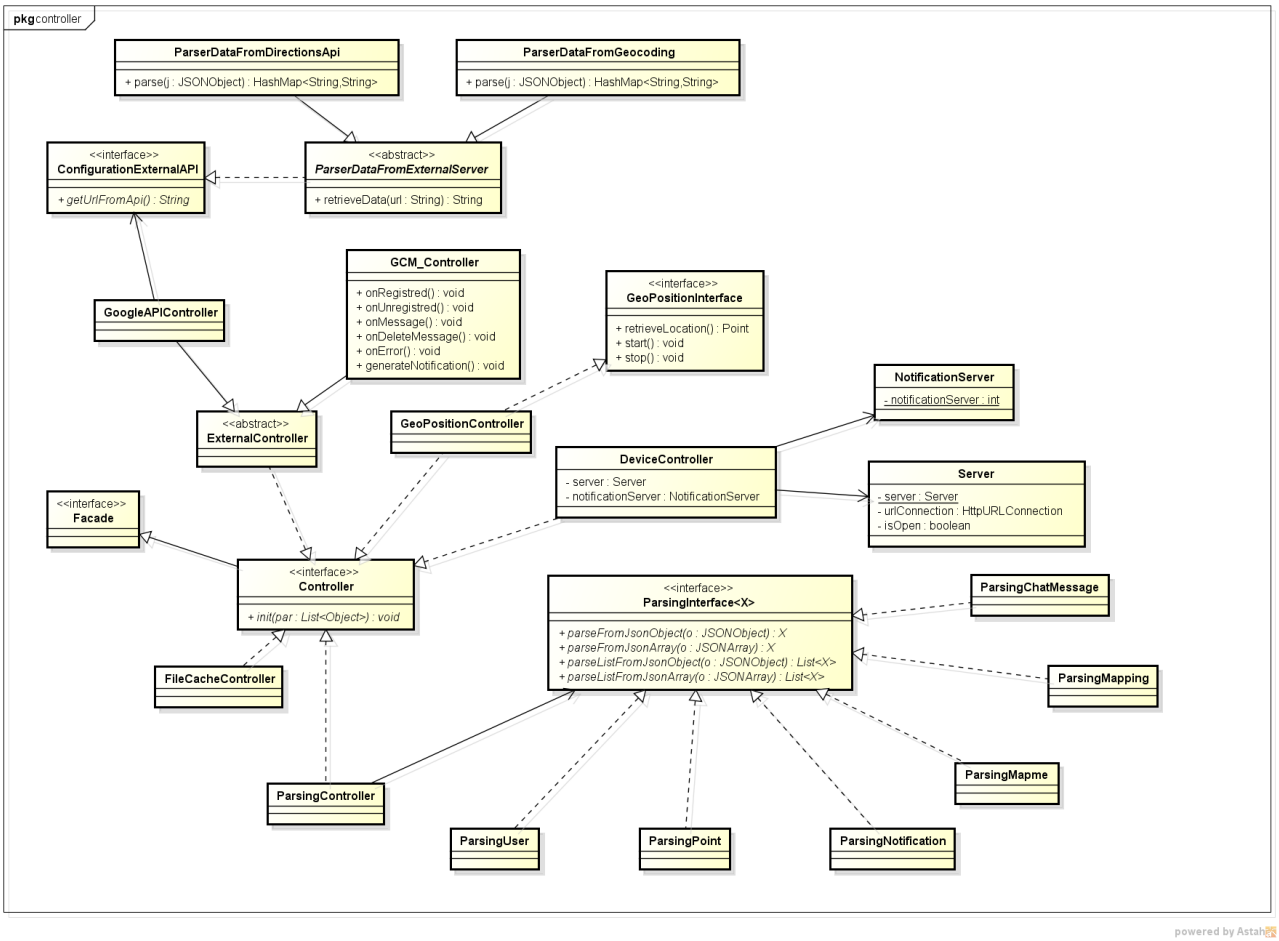

Fig. 5 Class diagram del componente Controller.

Il componente Controller è composto da classi di controllo che hanno la responsabilità di gestire funzionalità specifiche. Ogni classe di controllo è stata progettata utilizzando il pattern Facade, in modo tale da esporre esternamente soltanto l'interfaccia di accesso, nascondendone la logica. Le classi di controllo sono:

Nome della classe	Interazione con	Funzionalità gestita
GCM_Controller	Sistema di notifiche	Messaggi di notifica.
GoogleAPIController	Servizi di Google	Gestione delle location e dei percorsi.
Parsing_Controller	Web server	Gestione e manipolazione delle informazioni scambiate con il web server.
FileCacheController	Device	Gestione della cache.
DeviceController	Device	Gestione del device.
GeoPositionController	GPS Application	Gestione e rilevamento delle informazioni geografiche degli utenti.

Tale progettazione consente l'estensione di nuove funzionalità mantenendo basso l'accoppiamento tra sottomoduli, senza inficiare la struttura preesistente.

- **View:** componente che gestisce l'interazione con l'utente.

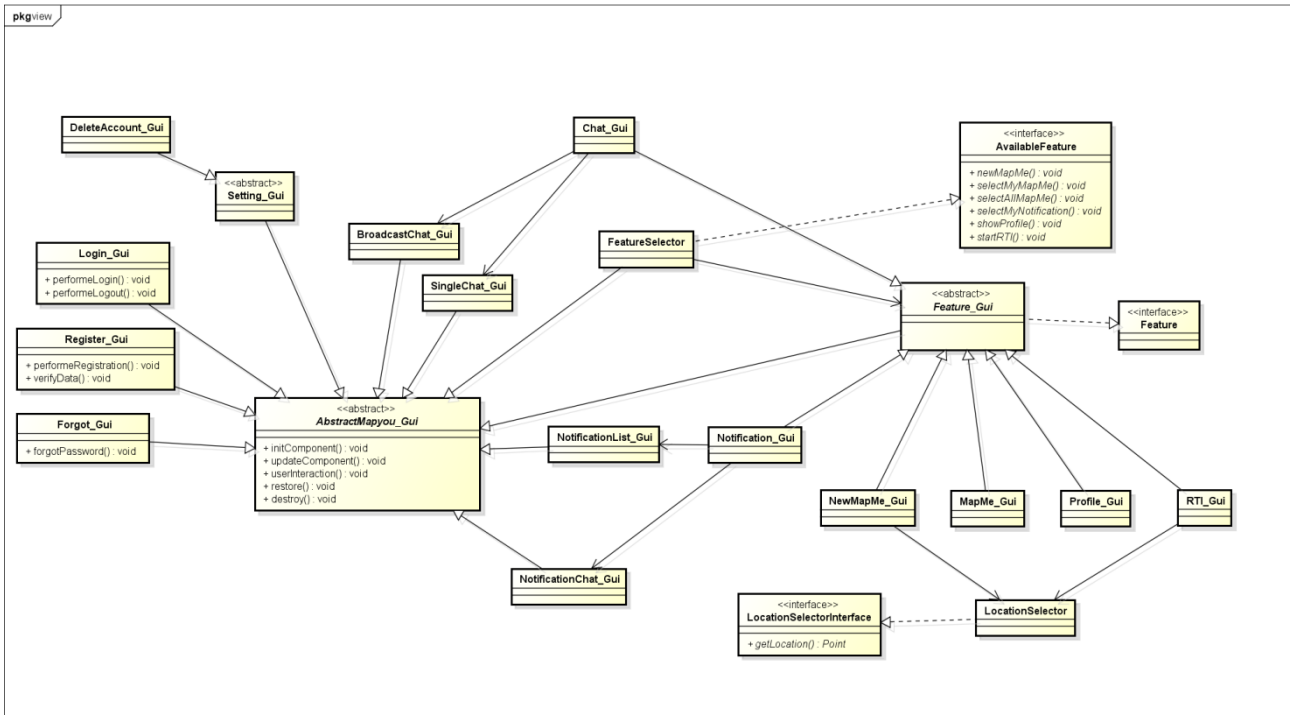

Fig. 7 Class diagram del component View.

L'idea base per la progettazione del componente View è che ogni classe e/o gruppo di classi gestisca una specifica funzionalità del sistema, in maniera tale da essere facilmente estendibile a nuove implementazioni, a prescindere dalla piattaforma sottostante (nel caso specifico: Android).

- **Communication Interface:** tale componente verrà trattata nel paragrafo 3.2.3.

3.2.2. Sottosistema Server

Il sistema server riceve richiesta effettuate dai client per la gestione e memorizzazione dei dati persistenti. La gestione della persistenza è stata progettata, come per il client, con il pattern DAO. Di conseguenza, il sistema server utilizza lo stesso modello dati del client. Per questi motivi, tali componenti non verranno descritti in quanto presentati nel paragrafo 3.2.1.

Oltre al componente della persistenza e del modello dati, il sistema server presenta i seguenti componenti:

- **Server_Interface:** interfaccia di accesso alle funzionalità fornite dal Notification-System.

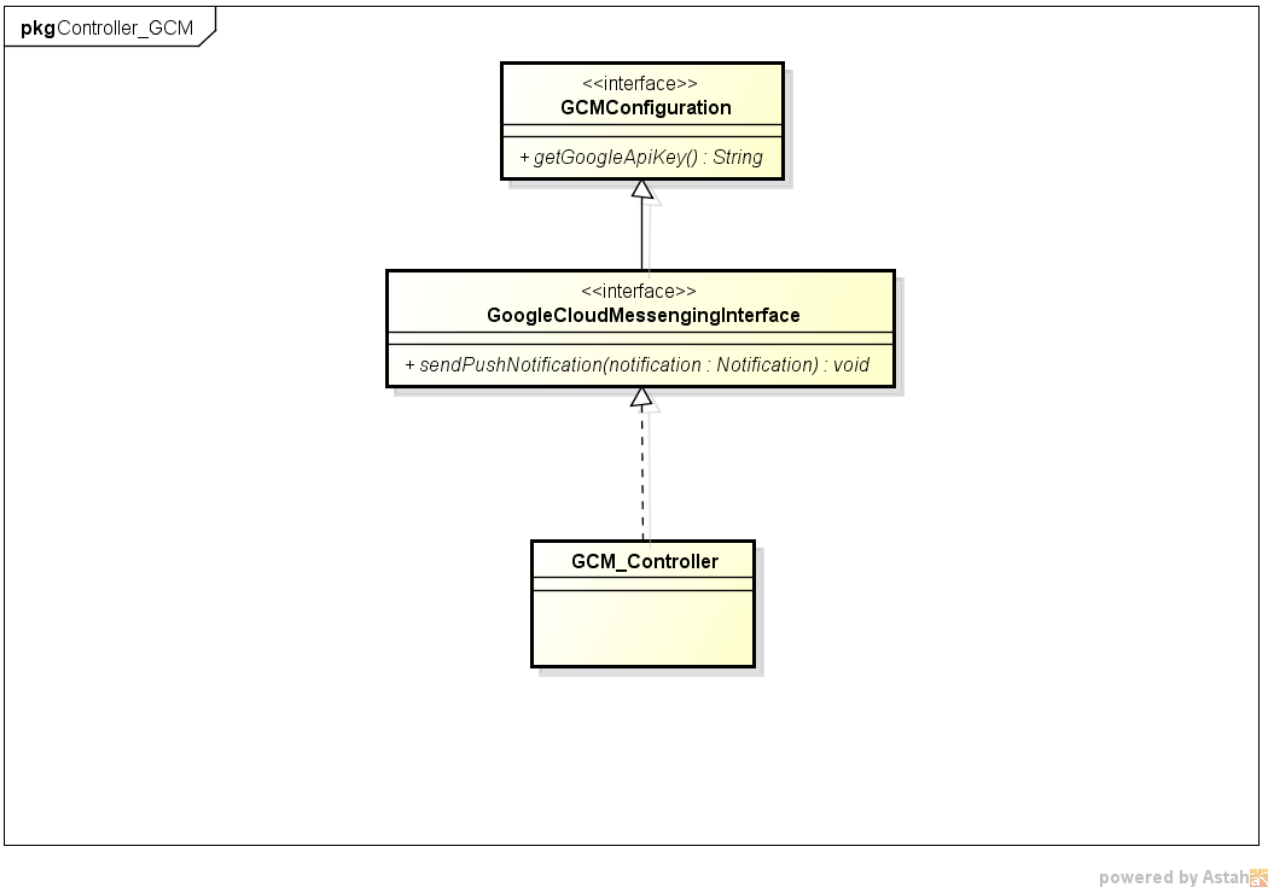

Fig. 8 Interfaccia verso il Notification-System.

- **AppsController:** interfaccia di accesso alle funzionalità fornite dal web server. Tale componente è stato progettato secondo i seguenti pattern:
    - Facade: per consentire ai client di interfacciarsi con il web server, senza conoscerne l'effettiva logica di controllo;
    - Proxy: gestisce l'abilitazione o meno di determinate funzionalità;
    - Factory Creator: gestisce l'accesso, l'inizializzazione e la chiusura della specifica sorgente di persistenza tramite interfacce comuni, in maniera tale da non modificare la logica di controllo del server in caso di cambiamento della sorgente.

Chiaramente, nell'implementazione attuale, le classi Facade e Proxy avranno le stesse responsabilità in quanto sono abilitate tutte le funzionalità descritte nel Software Requirements Specification. La scelta di effettuare una distinzione di ruoli tra le classi Facade e Proxy è dettata dal requisito di riusabilità e manutenibilità del codice.

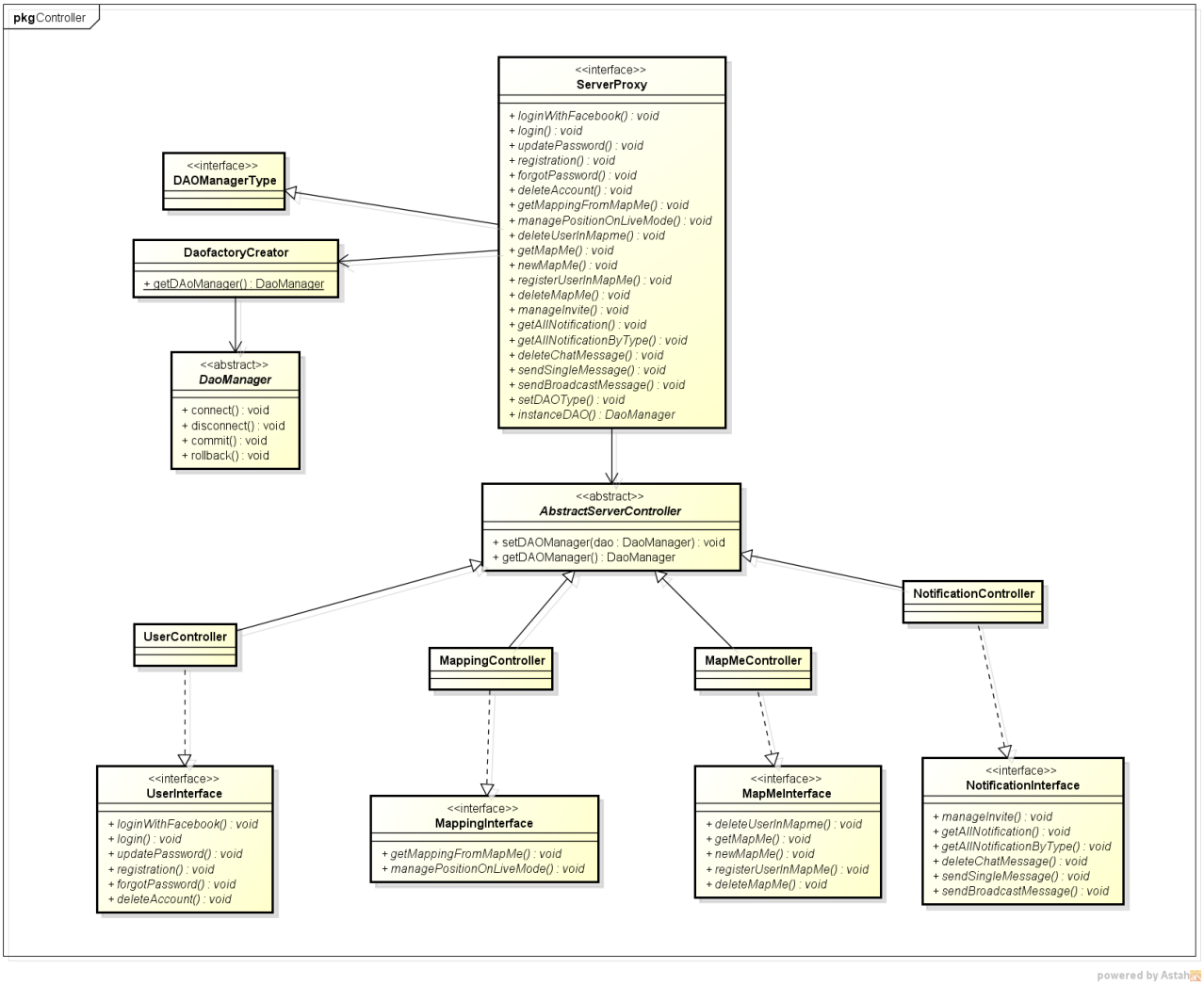

Fig. 9 Controller sul web server.

3.2.3. Sottosistema di comunicazione

Tutte le comunicazioni avvengono mediante scambi di messaggi utilizzando il protocollo HTTP.

Il sottosistema di comunicazione sul client è stato modellato attraverso due classi principali, utilizzando il pattern Singleton, in modo da non avere istanze multiple degli oggetti in memoria:

- Server: gestisce la comunicazione con il web server;
- NotificationServer: gestisce la comunicazione con il Notification-System.

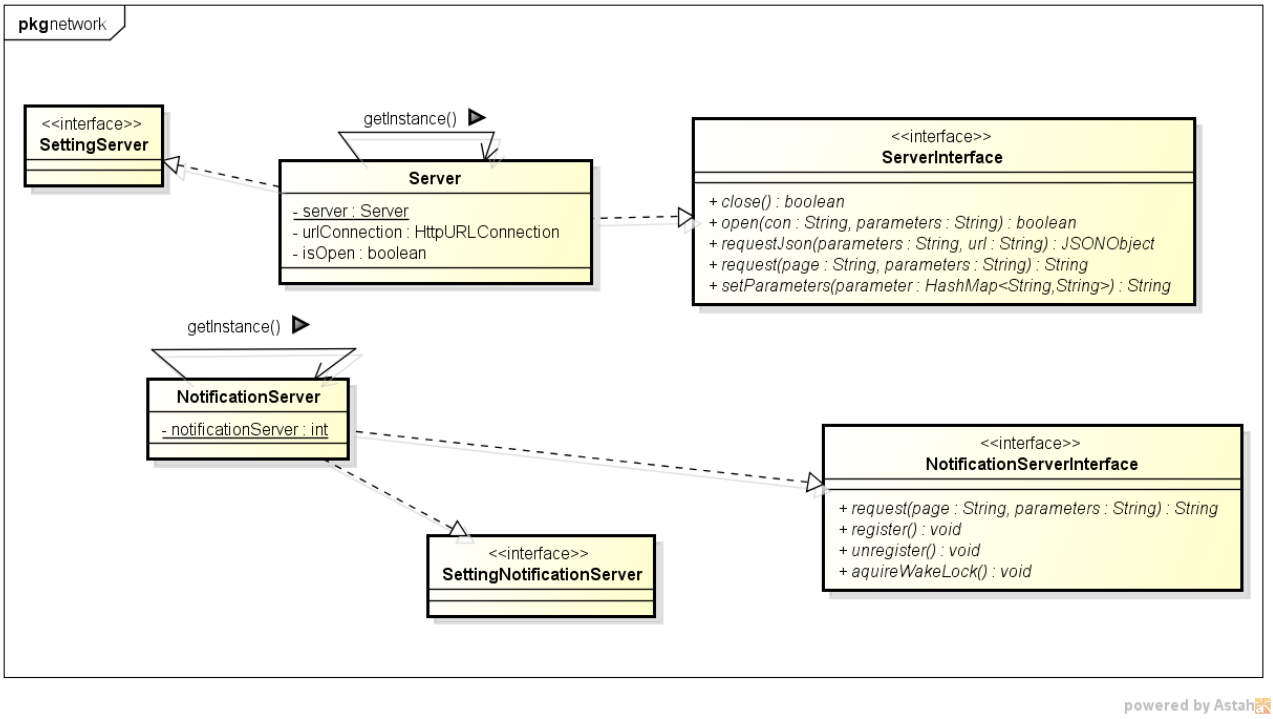

Fig. 10 Sottosistema di comunicazione.

Uno schema di comunicazione ad alto livello è fornito in Fig. 11.

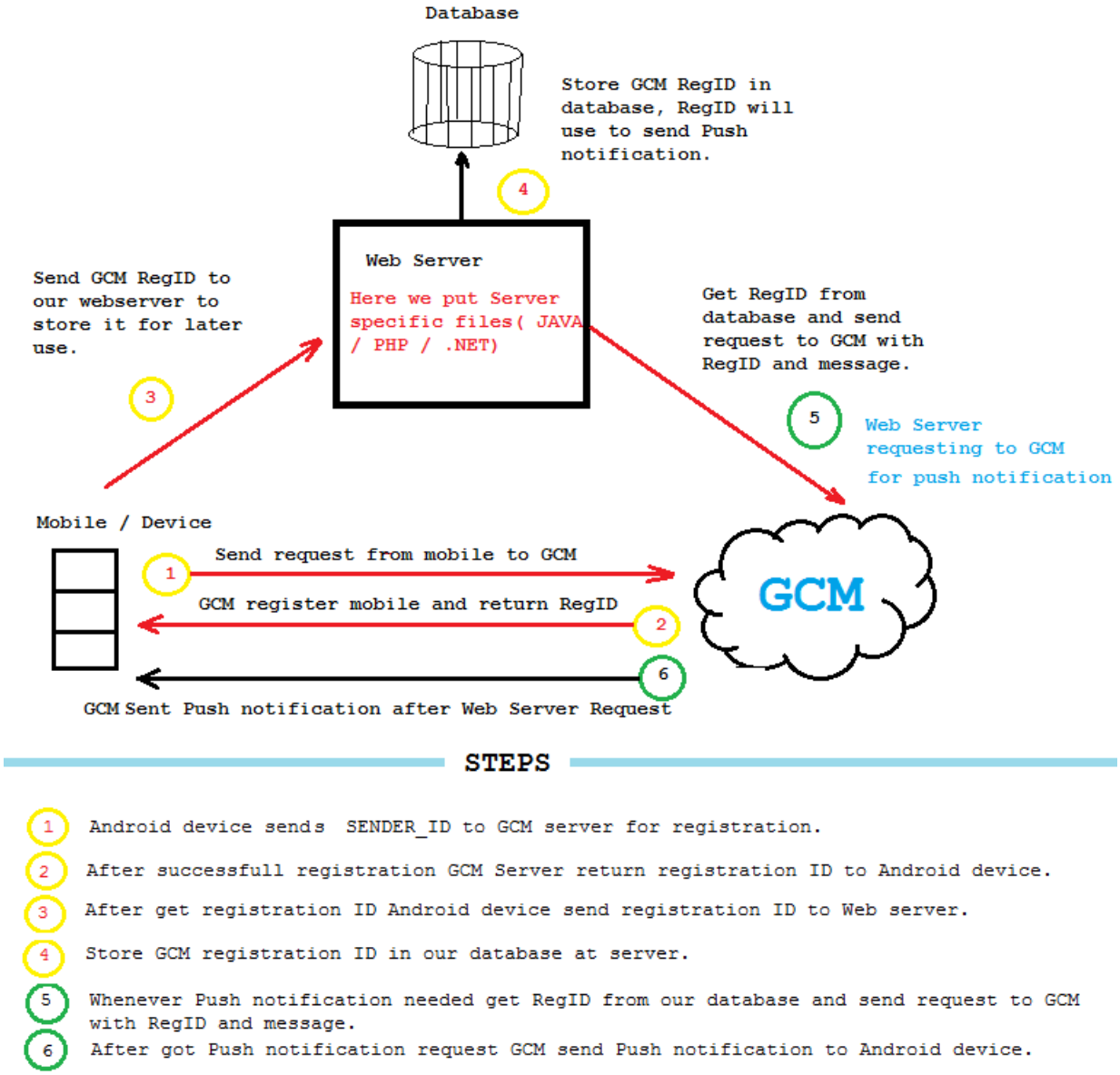

Fig. 11 Esempio di comunicazioni tra dispositivo, web server e Notifiation-Sysem.

3.3. Diagrammi di interazione

In parallelo allo sviluppo del sistema sono stati progettati alcuni diagrammi di interazione per determinati casi d'uso specificati nel Software Requirements Specifications.

- ID: FR-1

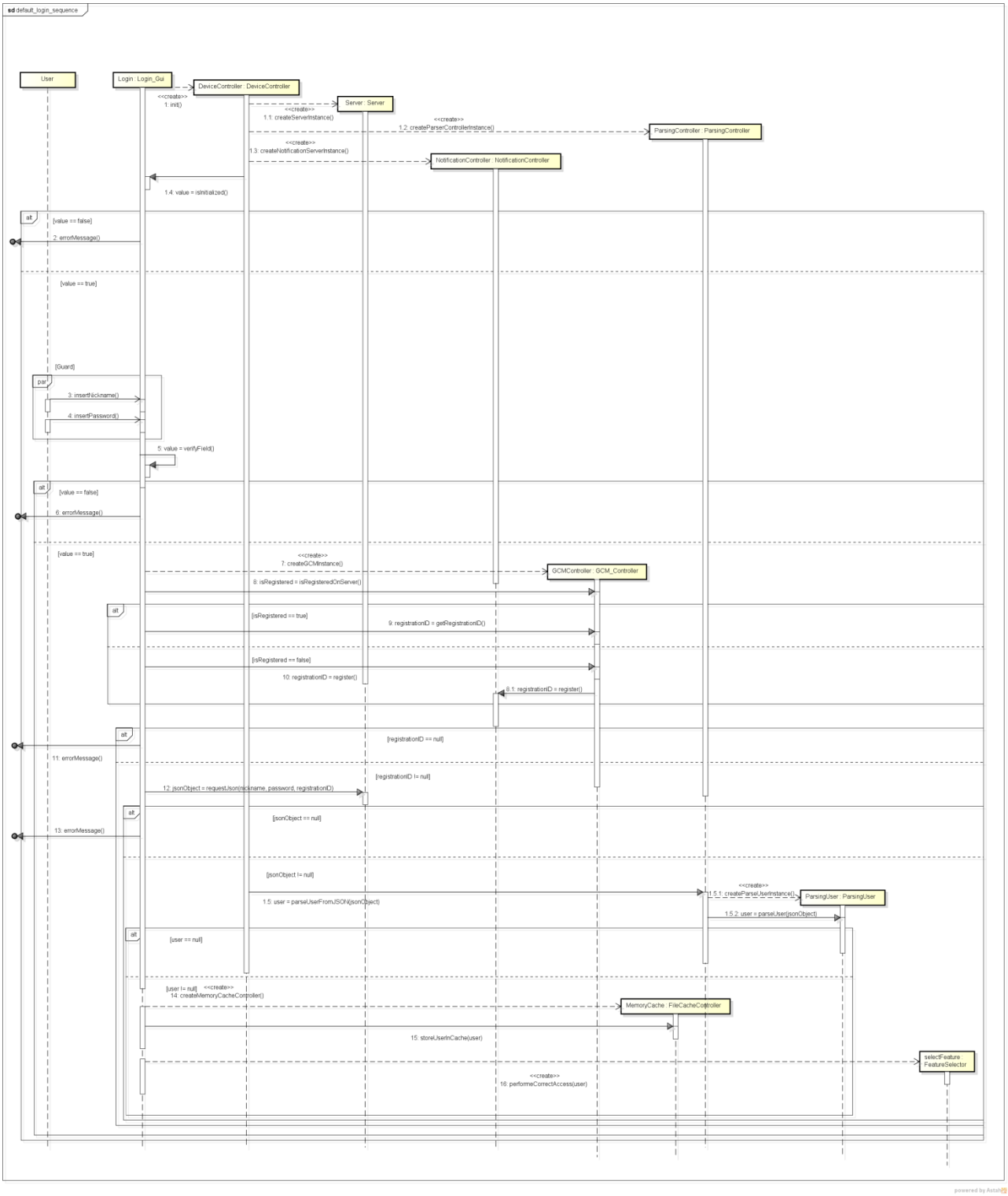

Fig. 12 Sequence diagram "Accesso al sistema".

- ID: FR-9

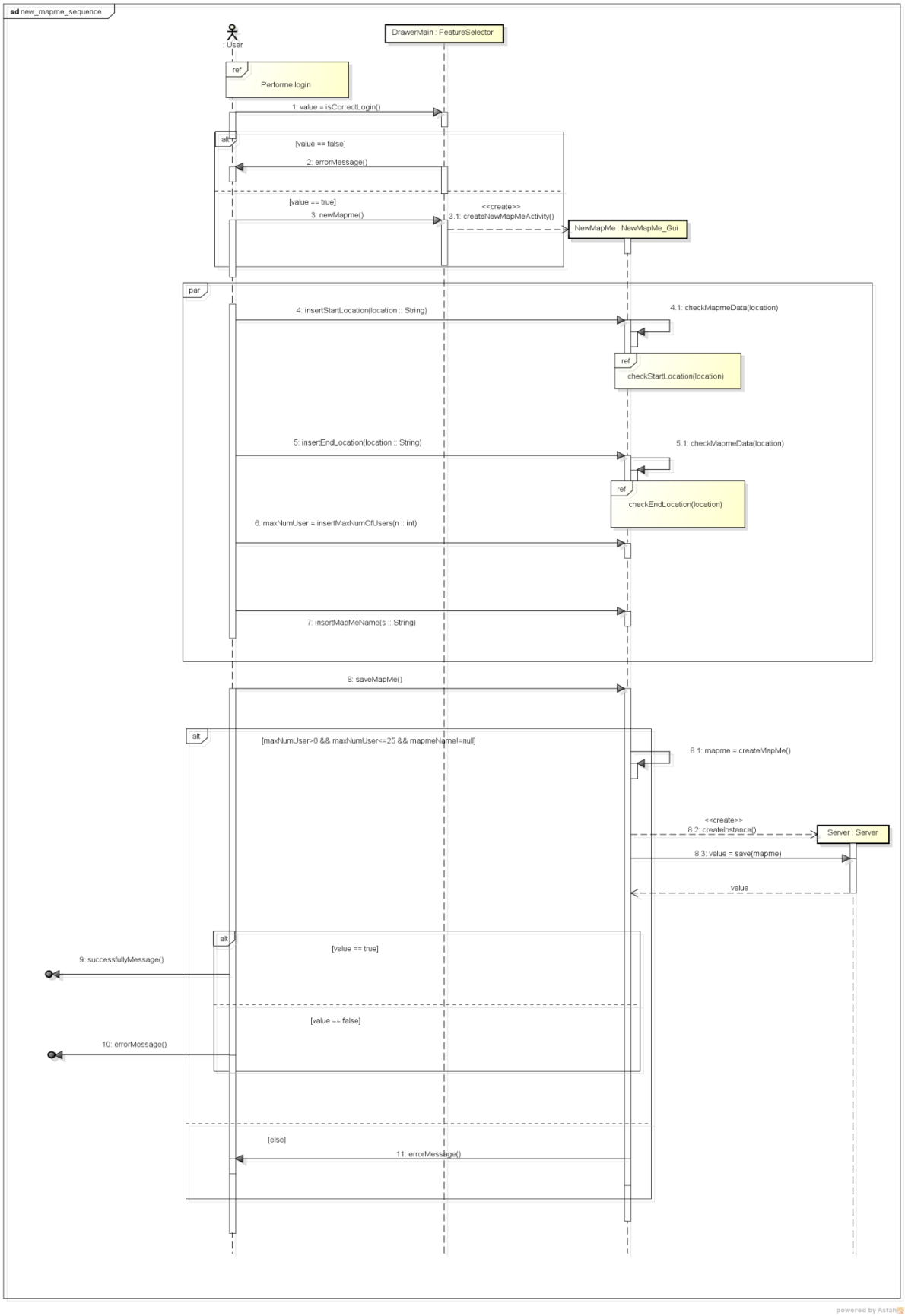

Fig. 13 Sequence diagram “Creazione di una MapMe”.

- ID: FR-11

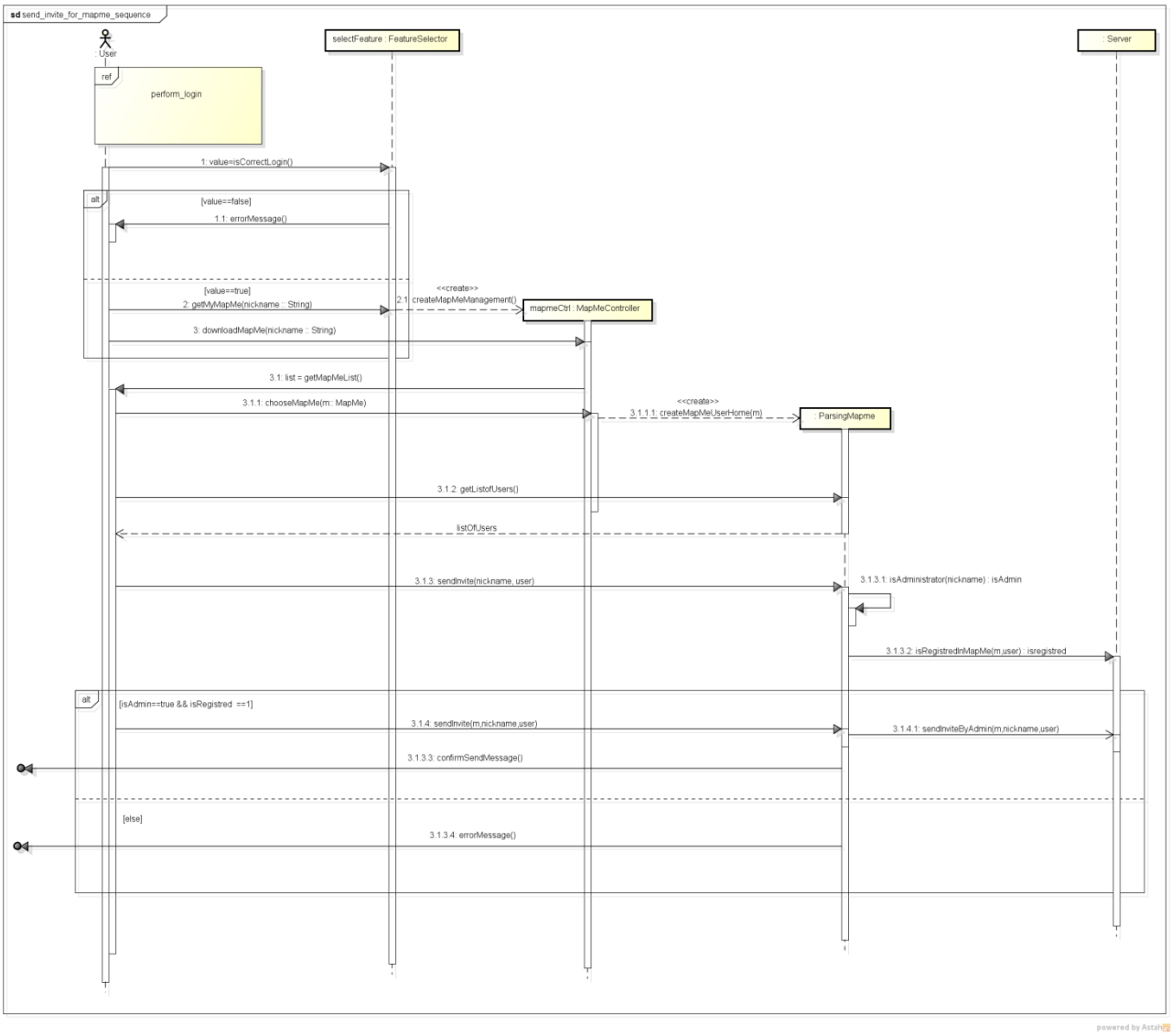

Fig. 14 Sequence diagram “Invito di partecipazione ad una MapMe”.

- ID: FR-12

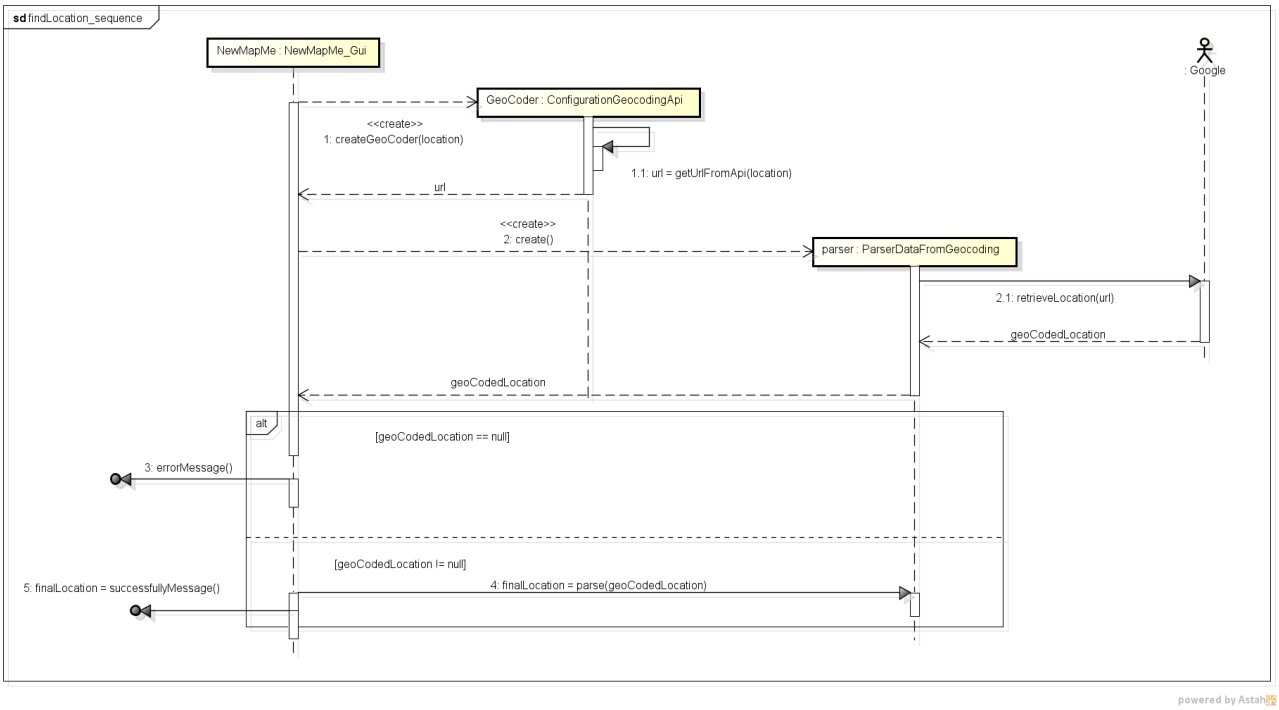

Fig. 15 Sequence diagram “Selezione di una location”.

3.4. Mapping Hardware/Software

Per il sistema, basato su un'architettura distribuita client/server, è stata scelta la seguente configurazione:

- Piattaforma Client: dispositivo Android la cui versione del sistema operativo sia superiore a 2.3;
- Connettività: Tasso di trasmissione standard (minimo 56 kbps);
- Protocollo di rete: HTTP;
- Piattaforma Server: Web Server Apache, fornito dal servizio di hosting di Altervista (http://www.altervista.org/), con estensione php;
- Gestione della persistenza sul server: DBMS MySql fornito da Altervista. L'interfaccia software di comunicazione con il DBMS utilizzata è PhpMyAdmin;
- Gestione della persistenza sul client: filesystem presente sul dispositivo Android;
- Notification System: servizio di push notification fornito da Google (GCM, Google Cloud Messaging);
- GPS Application: applicazione di gestione del dispositivo GPS integrata nel dispositivo Android.
- Tecnologia per lo scambio di informazioni tra client e server: JSON.

Il sistema è vincolato dai seguenti limiti:

- Dimensione massima del database: 500 MB;
- Potenza del database: 20000 operazioni orarie;
- Notification System: ogni messaggio può avere una dimensione massima di 4 Kb. I messaggi che eccedono tale limite saranno rifiutati.

3.5. Gestione dei dati persistenti

Sia sul server che sul client la gestione dei dati persistenti è stata progettata utilizzando il pattern DAO (Data Access Object).

Sul client, però, poiché le informazioni principali da memorizzare sono i percorsi delle Mapme quando si accede alla sezione RTI, è risultato più efficiente, in termini di risparmio energetico e query a grana grossa, utilizzare il filesystem nativo anziché un DBMS. Ciò nonostante, tale pattern è stato comunque implementato, nello specifico utilizzando il database SQLite, per possibili estensioni future ad altre sorgenti di persistenza.

3.6. Flusso di controllo globale

Il server sarà sempre funzionante, in attesa di eventuali richieste di servizi da parte dei client: nel caso di più richieste contemporanee, il server utilizzerà la tecnica FIFO (First In First Out).

La logica di controllo, sia sul client che sul web server, è decentralizzata, in maniera tale da ridurre le probabilità di colli di bottiglia dovuti ad accessi numerosi e contemporanei.