

2021/2022



CovidSafe

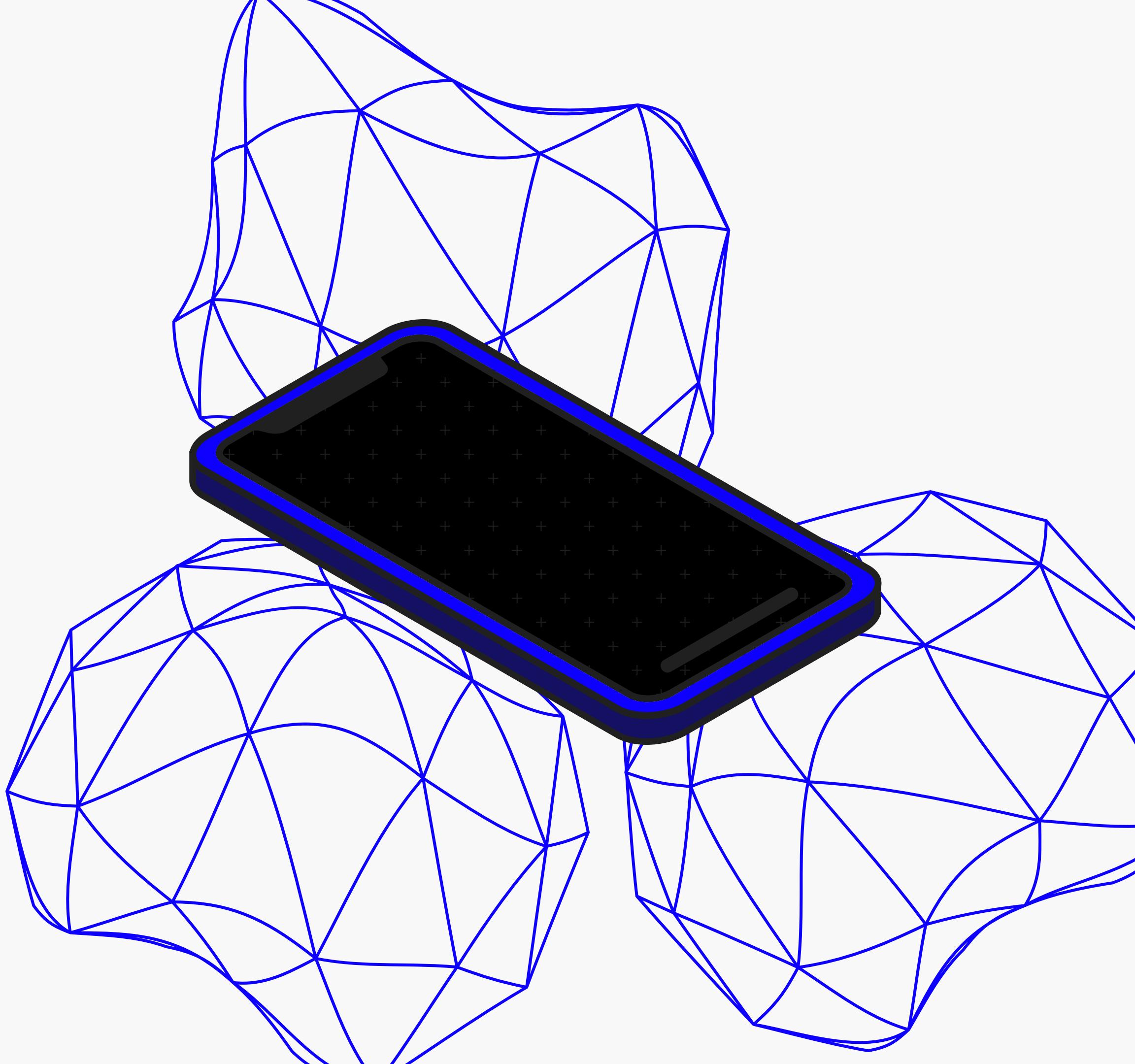
Università degli Studi di Napoli Federico II

Introduzione

CovidSafe

E' **un'applicazione** per dispositivi mobili che offre dei servizivolti al monitoraggio ed alla segnalazione di casi Covid-19 alle autorità competenti ed alle persone che hanno intrattenuto contatti con individui contagiati.

Inoltre permette di **monitorare** i luoghi chiusi per rilevare il numero di individui presenti in tempo reale e funzioni a fini commerciali (informazioni generiche, possibilità di prenotazione, etc.).



Metodología



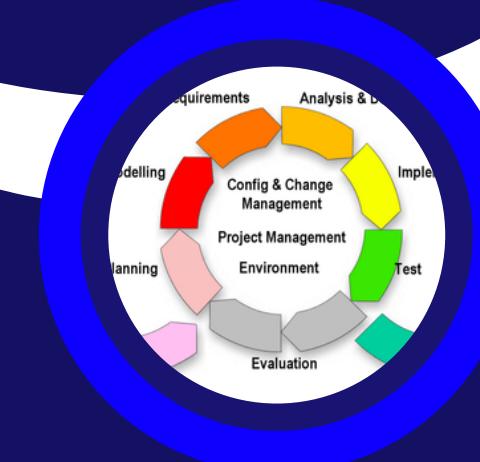
TEAMS



SCRUM

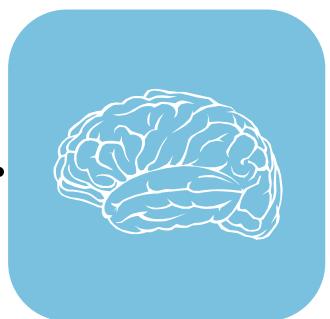


TRELLO



UP

Metodologia



Workshop dei requisiti:
2 giorni:
- Brainstorming:
- Specifica dei requisiti



**Studio di fattibilità
e tecnologie disponibili.**
3 giorni



**Prototipazione "usa
e getta"**
3 giorni



**Product
Backlog**

Product Backlog



Primo sprint

Realizzazione del modulo di trasmissione e della ricezione del segnale Bluetooth attraverso la libreria AltBeacon.



Secondo sprint

Costruzione del sistema per la persistenza dei dati di interesse sul dispositivo.



Terzo sprint

Realizzazione del modulo di comunicazione con il lato server del sistema software



Quarto sprint

Costruzione della componente server del sistema software.

Sprint Backlog

All'inizio della settimana

- Progettazione del sottosistema da sviluppare, delineandone il design e l'architettura (2 giorni).
- Implementazione, spesso modificando i documenti di progettazione in corso d'opera.

I 3 giorni centrali

Erano dedicati pienamente all'implementazione. Ove possibile, sono stati condotti dei test sulle singole classi.

Ogni Mattina

Meeting riepilogativo

Il sabato mattina

Sprint Review:
-Riunione riepilogativa ed organizzativa.
-Valutazione del lavoro svolto durante la settimana.



Tecnologia Bluetooth

- Tracciamento assoluto e relativo.
- E' applicabile sia in ambienti chiusi che aperti.
- Setup e configurazione veloce
- Basso consumo di batteria
- Possibilità di determinare la **distanza** relativa tra dispositivi interagenti.
- Costi
- Ampiamente diffuso
- Esauriva documentazione e framework disponibili

BLE

- Banda a 2.4 GHz, restando costantemente in modalità riposo fino a quando non viene avviata una connessione
- Tempi di connessione molto ridotti
- Basso consumo di batteria

Protocol Data Unit

L'informazione standard dei beacon consiste in un [UUID](#) (Universally Unique IDentifier) ed in due interi chiamati [minor](#) e [major](#), valori assegnati per identificare i dispositivi con una accuratezza maggiore rispetto all'utilizzo del solo UUID.

Un [beacon](#) è un dispositivo BLE che trasmette piccole quantità di dati ad intervalli di tempo regolari.



AltBeacon Library

E' la principale libreria Open Source per il rilevamento dei beacons in [Android](#).

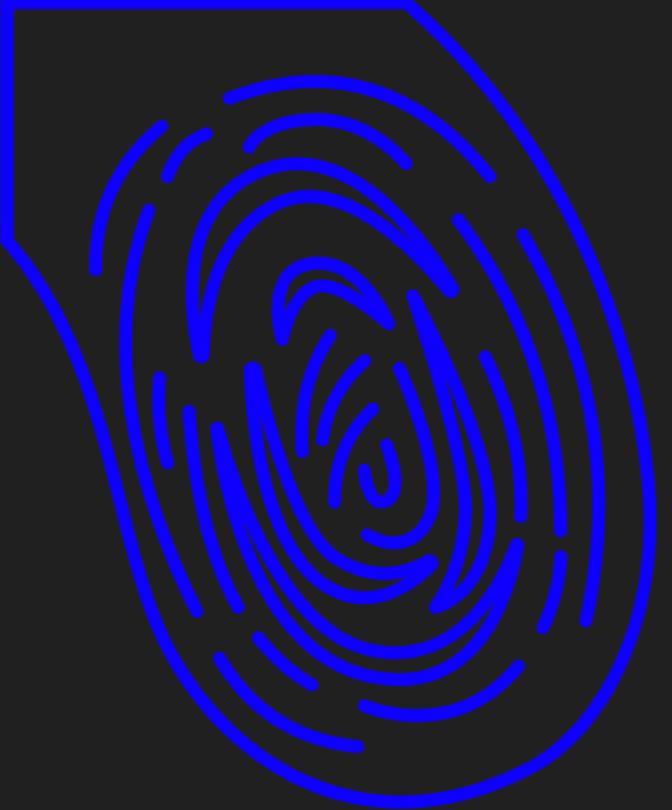
La libreria rileva i beacon che rispettano lo standard AltBeacon, ma può essere opportunamente configurata per rilevare altri formati (IBeacon di Apple, etc.).

Per trasmettere come beacon, è necessario Android 5+ e un firmware che supporti la [BLE Peripheral Mode](#). La libreria supporta sia la rilevazione che la notifica dello standard beacon Exposure Notification Service fornendo il massimo supporto per Eddystone

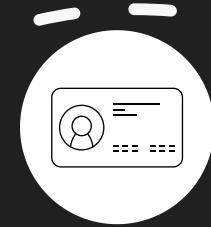


AltBeacon

The Open and Interoperable Proximity Beacon Specification

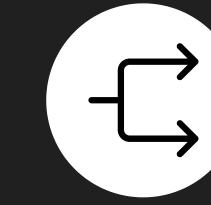


MECCANISMO DI IDENTIFICAZIONE



UUID

Si è provveduto a generare un UUID customizzato al fine di identificare la suddetta applicazione ed il servizio da essa offerto.



MINOR+MAJOR

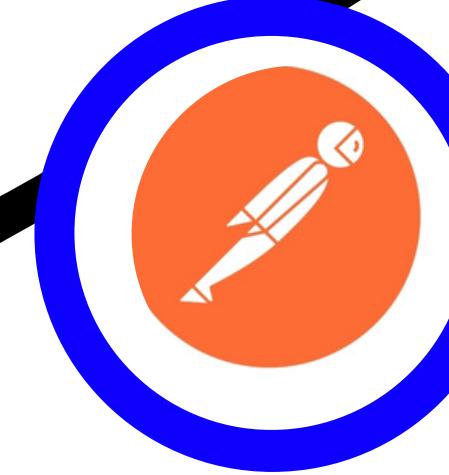
A ciascun utente, durante la fase preventiva di registrazione, viene associata una coppia di interi univoca (valori minor, major) al fine di identificare ciascun utente che usufruisce dell'applicazione



Strumenti Utilizzati



RETROFIT



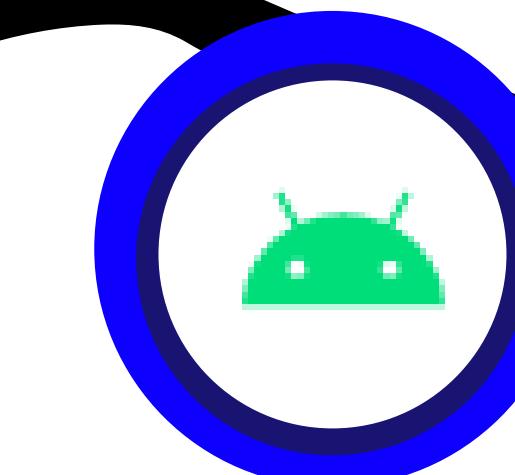
POSTMAN



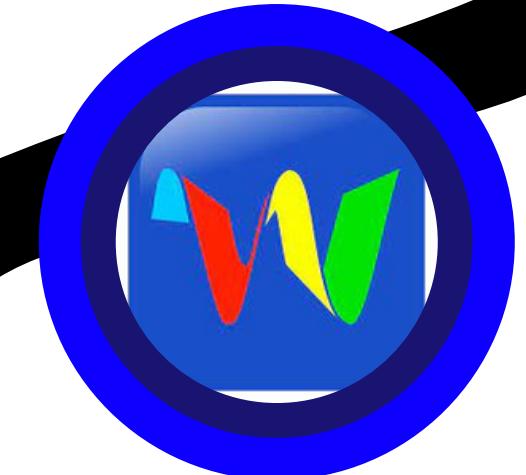
SPRING



XAMPP



ROOM

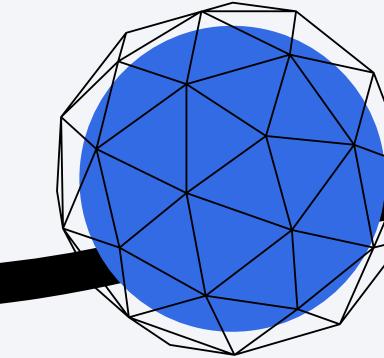
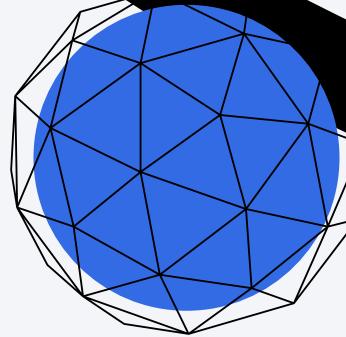


APACHE MAVEN



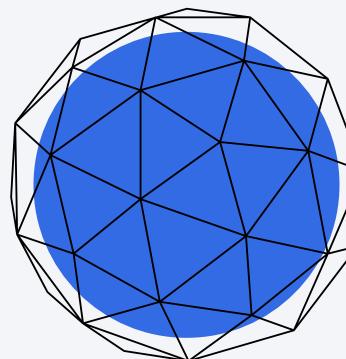
BEACON SCANNER
& SIMULATOR

ANALISI DEI REQUISITI



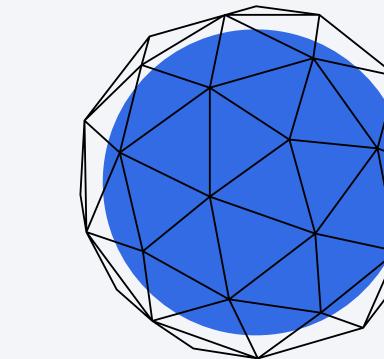
Obiettivi

Monitoraggio di luoghi aperti al pubblico (spiagge, bar, ecc.).
Tracciabilità della catena di contatti intrattenuti dall'utente.



Stakeholders

Utenti e Proprietari



Requisiti minimi

Hardware: Bluetooth 4.0

Software: sistema operativo Android 8+

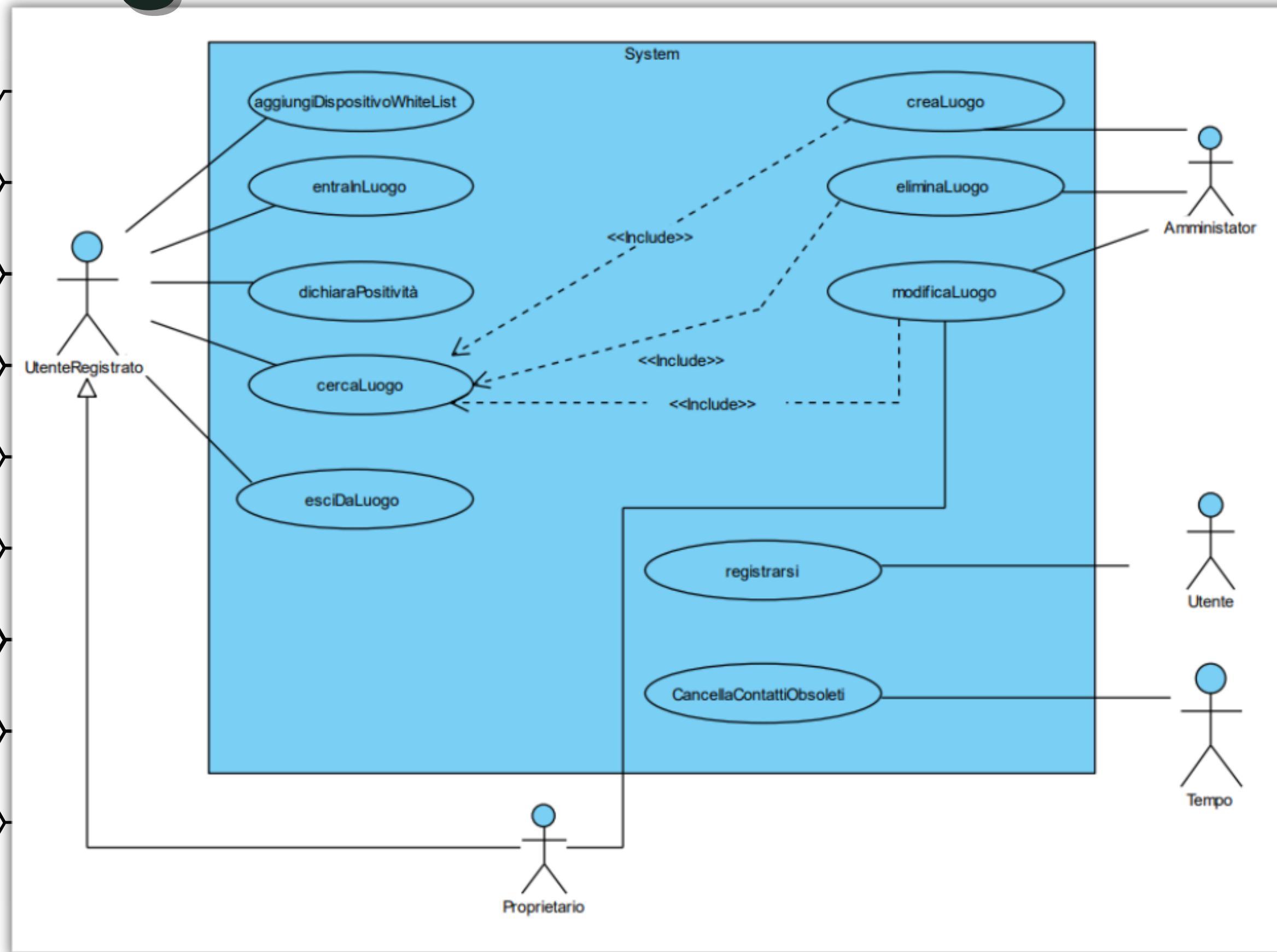
Vincoli

- Il sistema dovrà rispettare le norme vigenti sulla privacy.
- Il sistema dovrà essere portatile.
- Il sistema dovrà essere facilmente adoperabile dagli utenti.
- Il sistema dovrà essere scalabile.
- Il sistema dovrà essere evolvibile.
- Il sistema dovrà provvedere a fornire un meccanismo di protezione dei dati

REQUISITI LIVELLO UTENTE

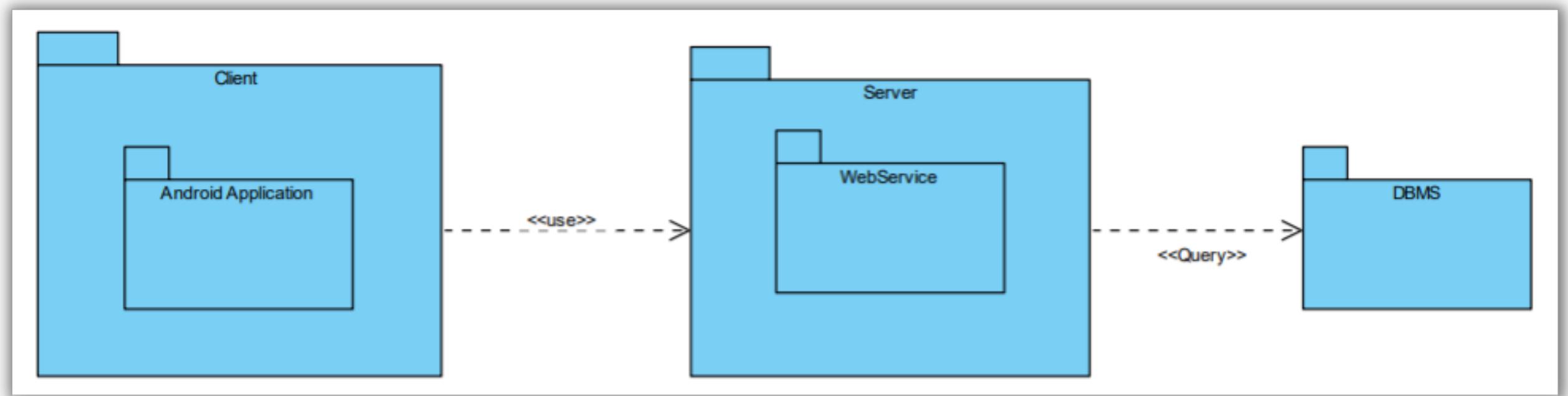
1. Fornire un meccanismo di **registrazione** al sistema.
2. Autenticare l'utente, consentendogli di accedere solo alle funzionalità di sua competenza.
3. Tenere traccia dei **contatti** intrattenuti dall'utente per un tempo definito di 15 giorni, oltre cui il rischio di contagio è da considerarsi nullo.
4. Fornire a qualsiasi utente la possibilità di **segnalare** la propria positività.
5. Notificare la positività di un utente alla sua **catena** di contatto.
6. Fornire la possibilità di creare una white list.
7. Elaborare **statistiche** con i dati raccolti.
8. Fornire un meccanismo di ricerca di luoghi pubblici e relative statistiche.
9. Monitorare il numero di **accessi** ai luoghi pubblici.
10. Fornire ai gestori la possibilità di modificare i parametri di un luogo pubblico di sua competenza.
11. Fornire ad un amministratore la possibilità di aggiungere ed eliminare luoghi pubblici e/o proprietari.

Diagramma dei casi d'uso

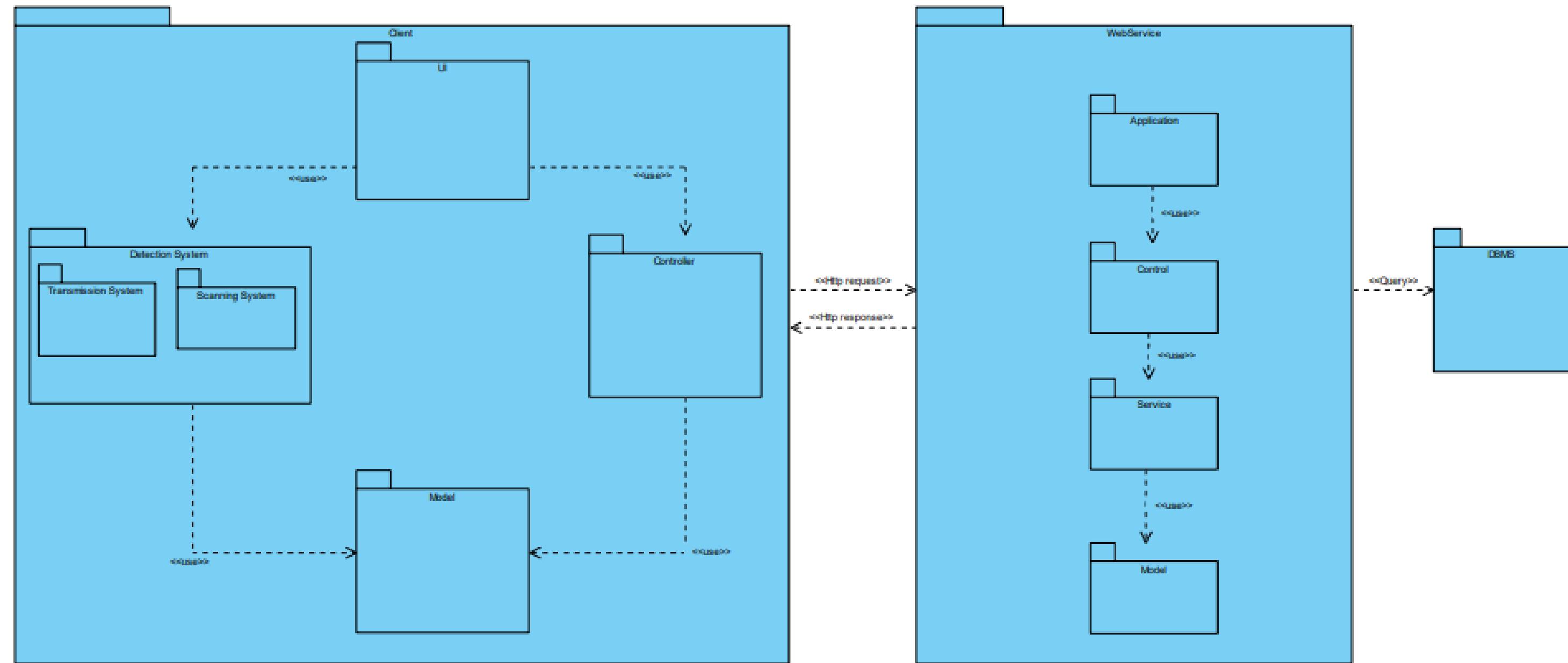


Architettura Generale

del Sistema Software.

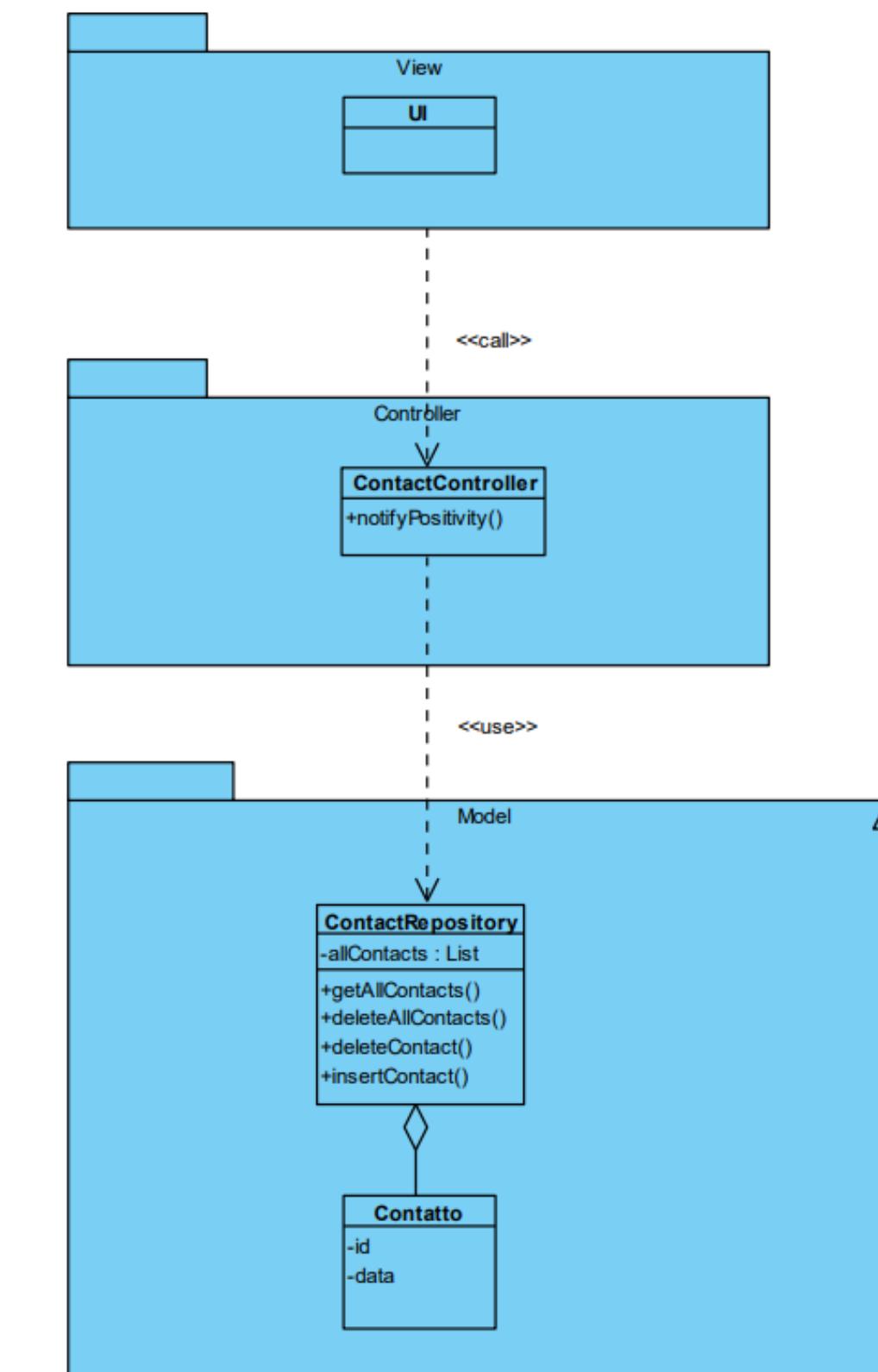


Organizzazione in Package



Pattern MVC

Con Repository Pattern



il model non aggiorna la view perchè è tassativo che i contatti non siano mostrati a video

Rest

POST REQUEST

FORMATO DATI

JSON

PARAMETRI DI INPUT

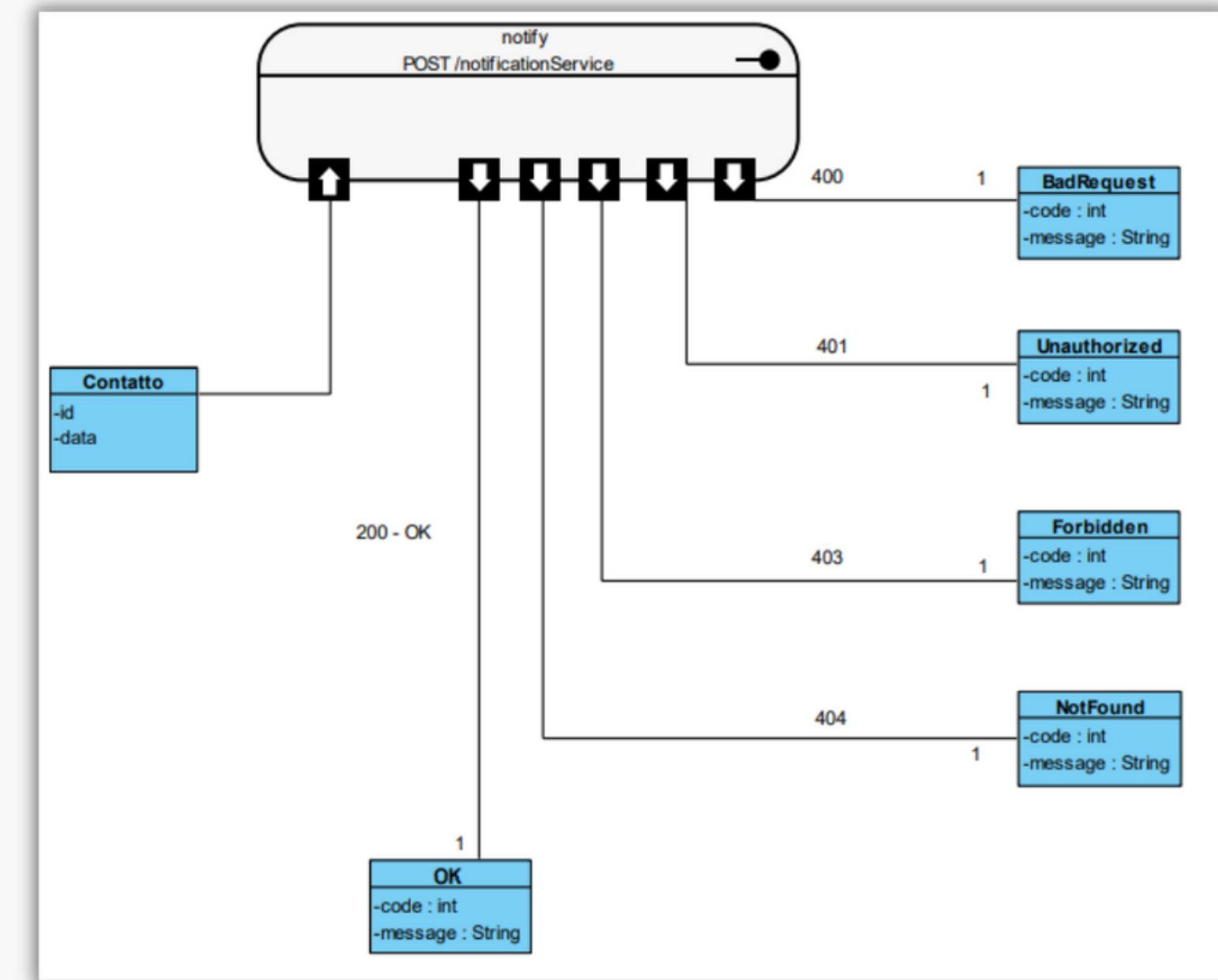
Tramite la classe POJO "contatto", i dati di interesse (data, identificativo) sono inviati, serializzando opportunamente l'oggetto, al servizio che provvede a notificare l'avvenuta positività agli utenti.

OUTPUT

Messaggio di risposta

IDENTIFICAZIONE DELLA RISORSA

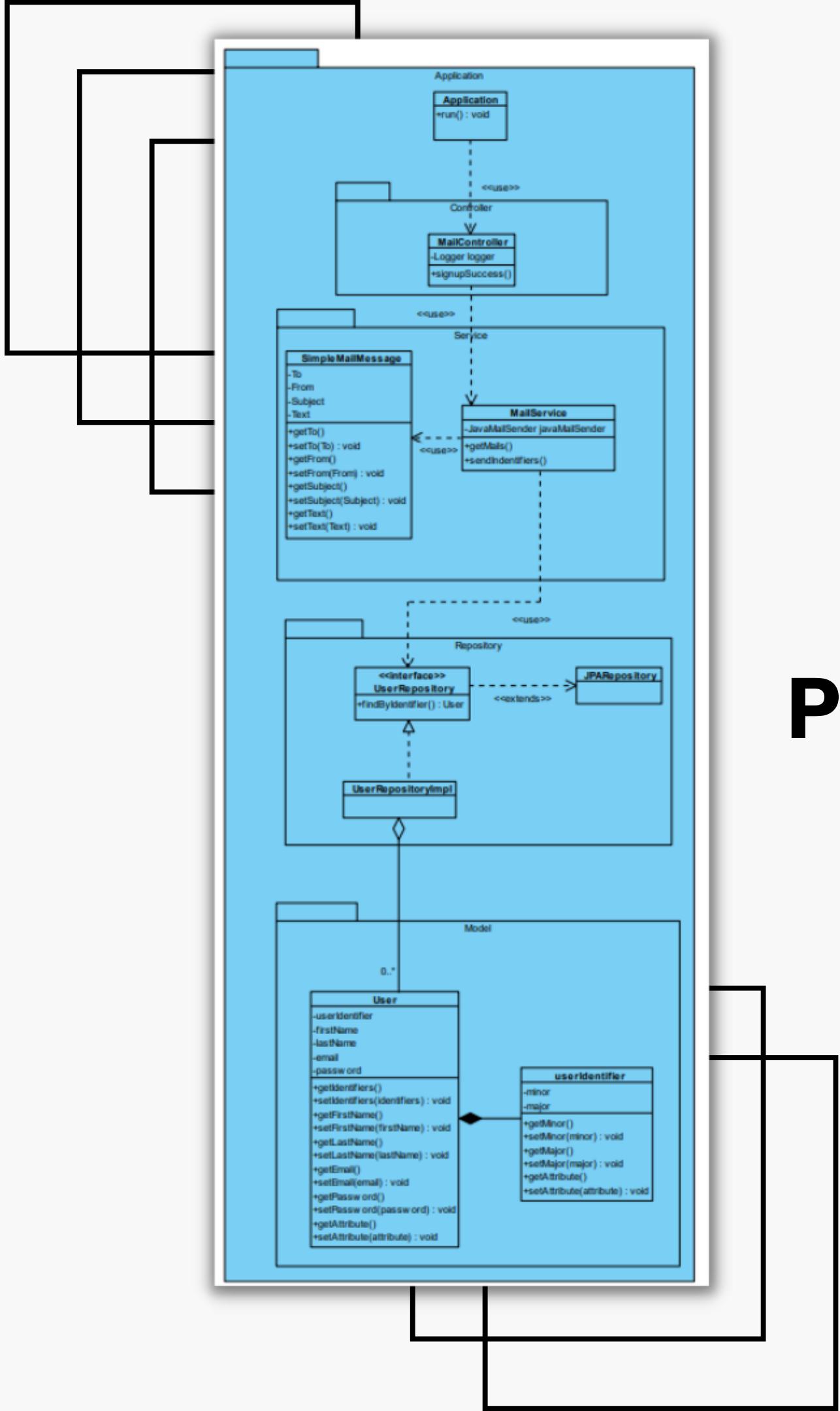
viene specificato un URL specifico ("notificationService") oltre che ad un base URL univoco



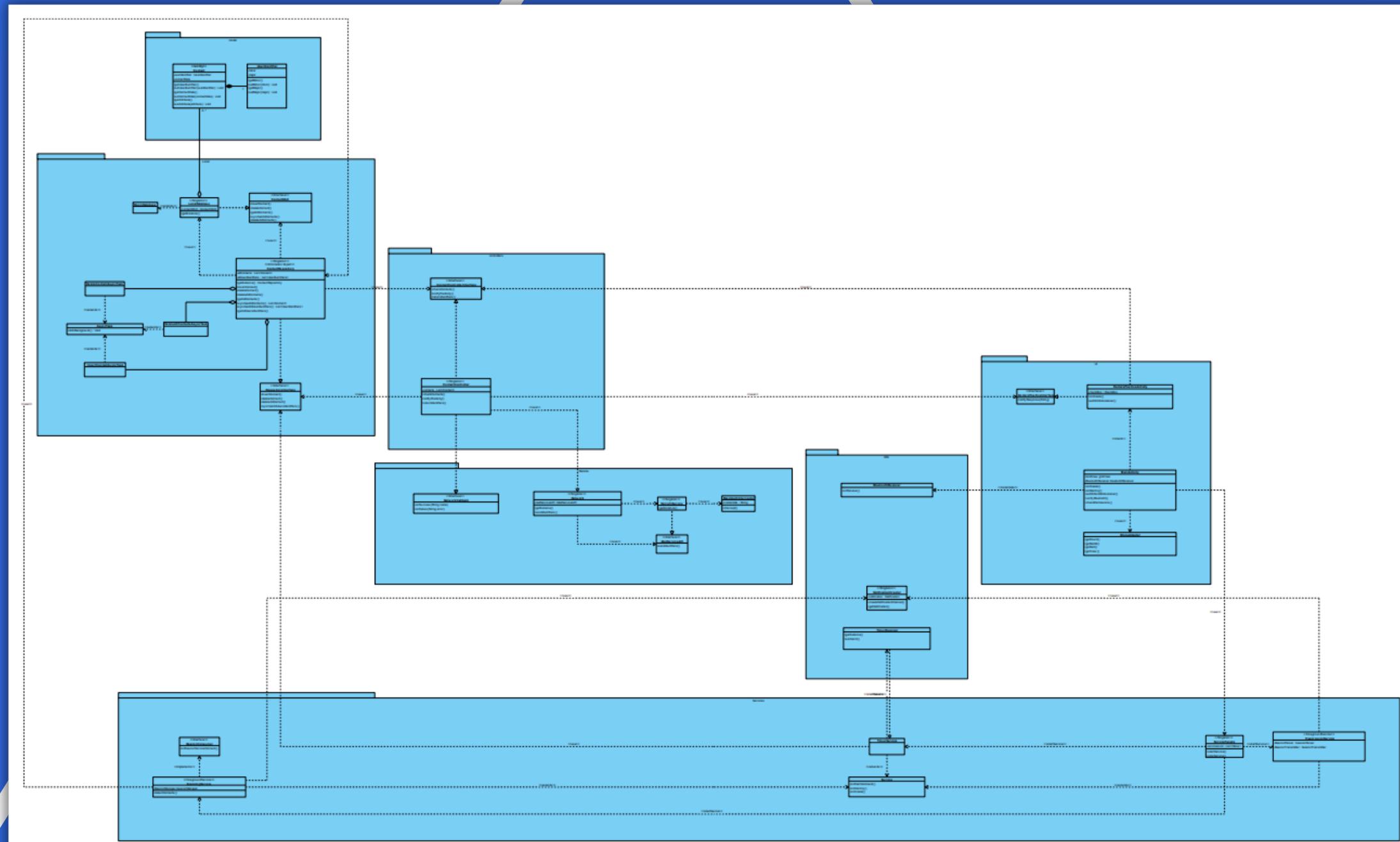
REST, COME STILE ARCHITETTURALE, È STATO DEFINITO SULLA BASE DEL PROTOCOLLO HTTP PER LA COMUNICAZIONE TRA **CLIENT** E **WEBSERVICE** (RESTFUL).

I sistemi REST sono **Stateless**: non prevedono il concetto di sessione. La comunicazione tra utente del servizio (client) e servizio (server) deve essere senza stato tra le richieste, inoltre la **risorsa** (un elemento di dati) esiste indipendentemente dalla sua rappresentazione

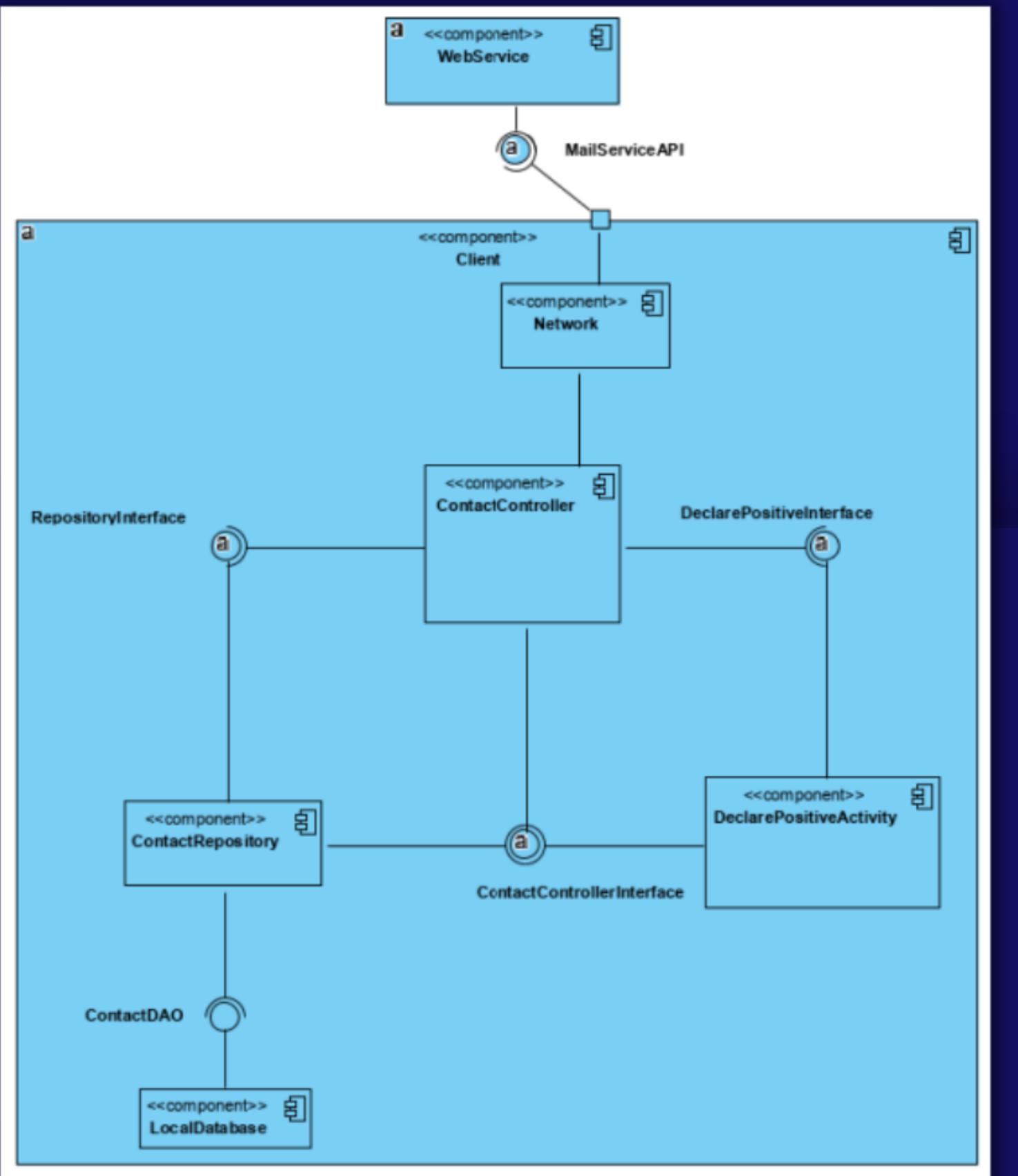
Package Diagram



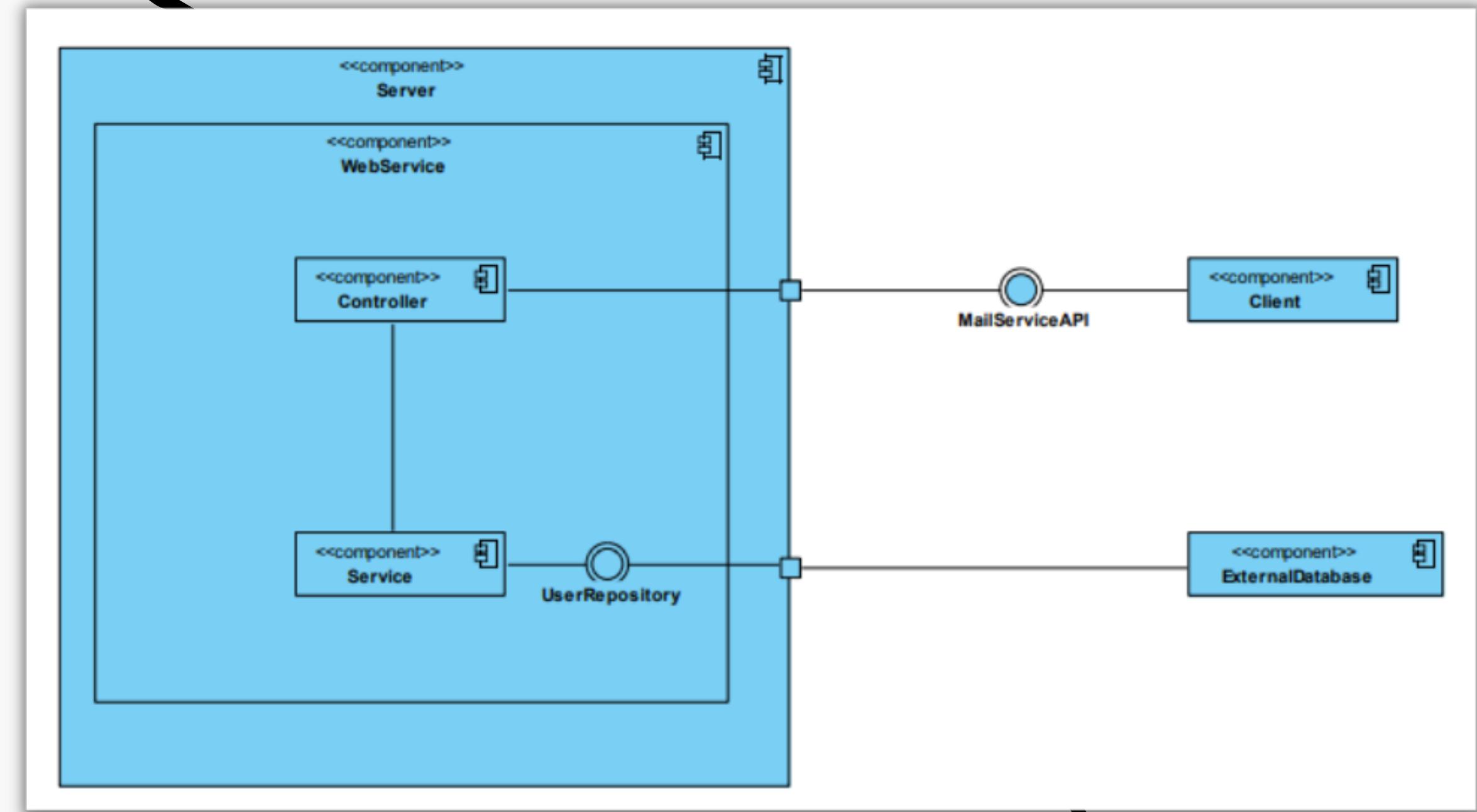
Package Diagram



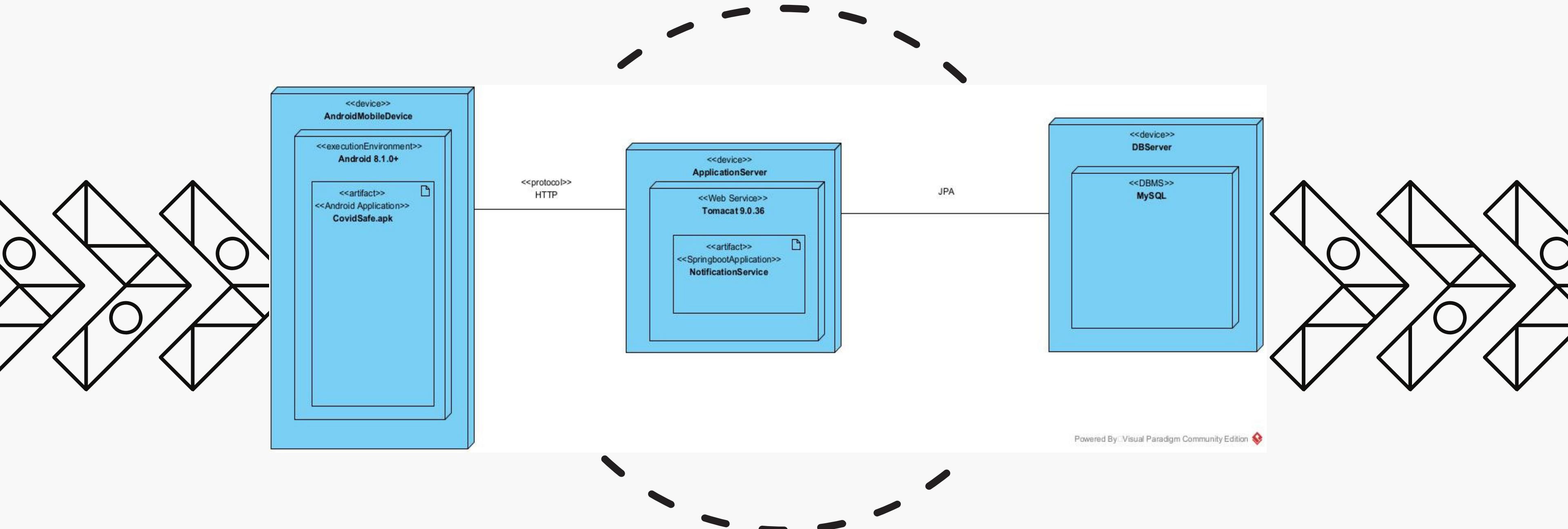
Component Diagram



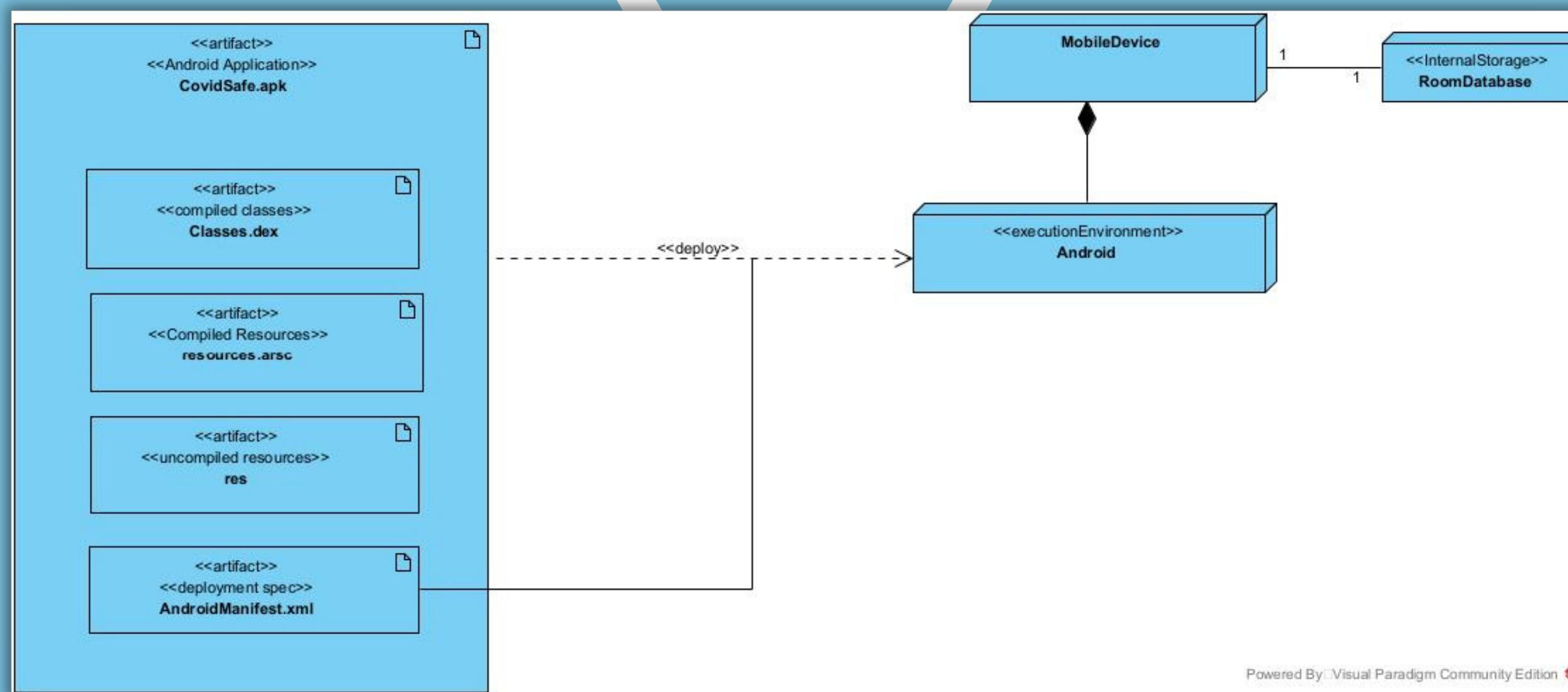
Component Diagram



Deployment Diagram



Deployment Diagram



System Domain Model

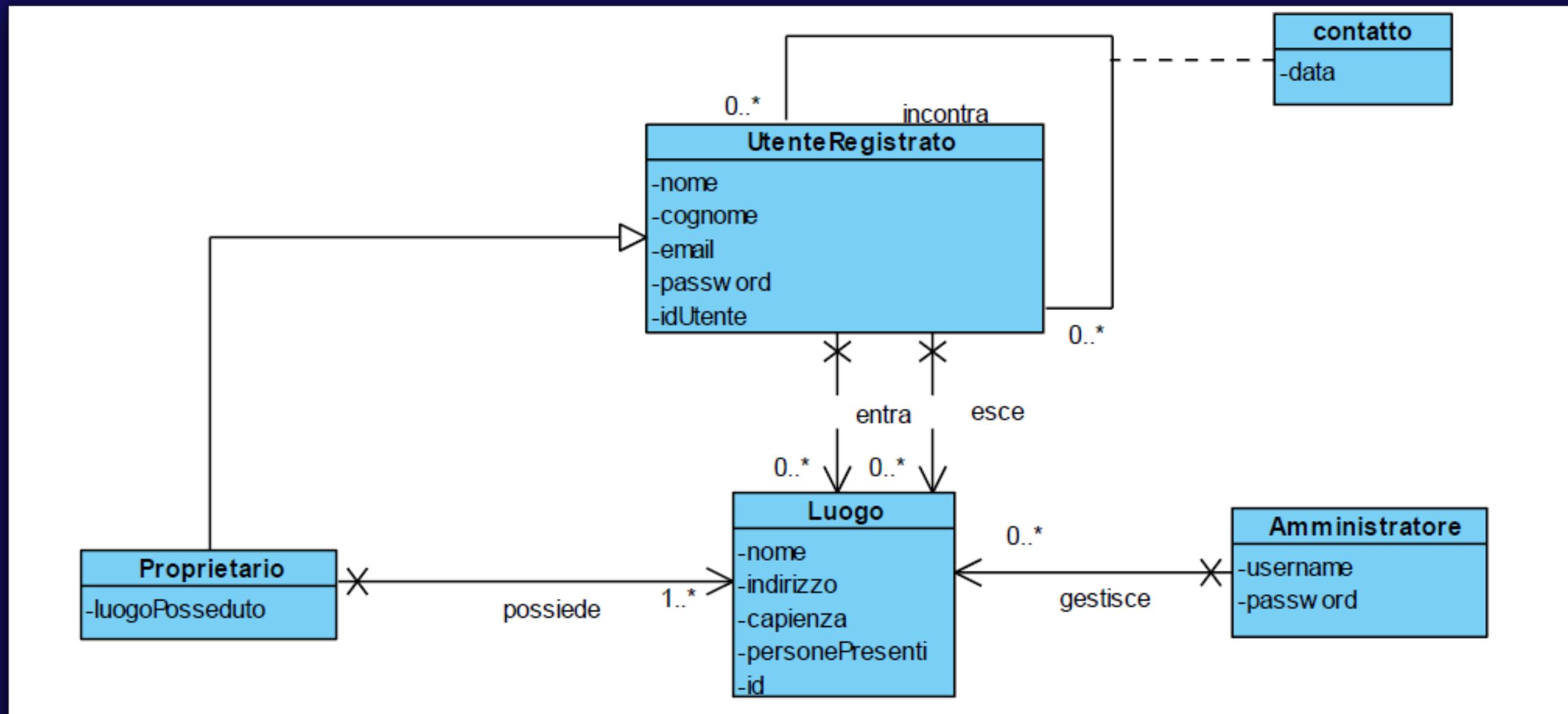
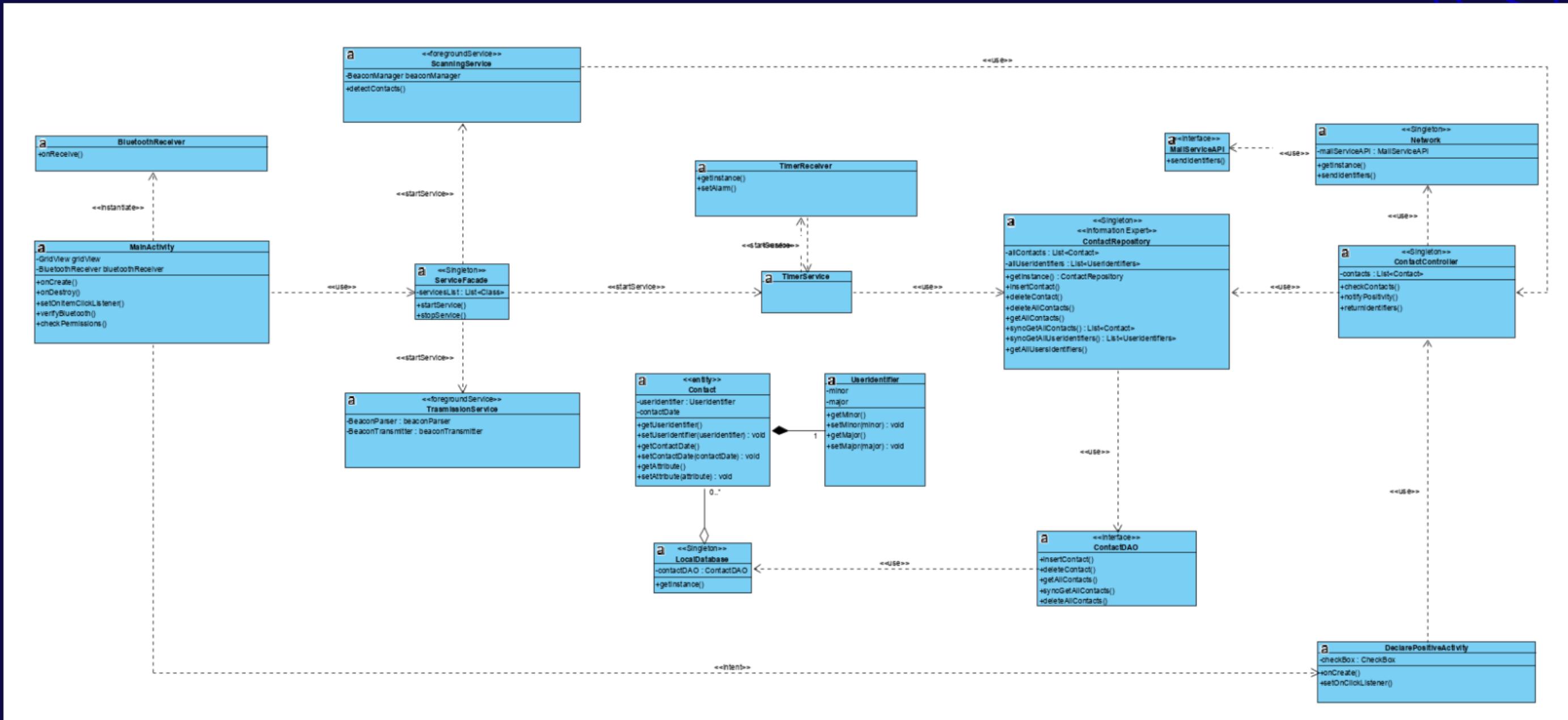
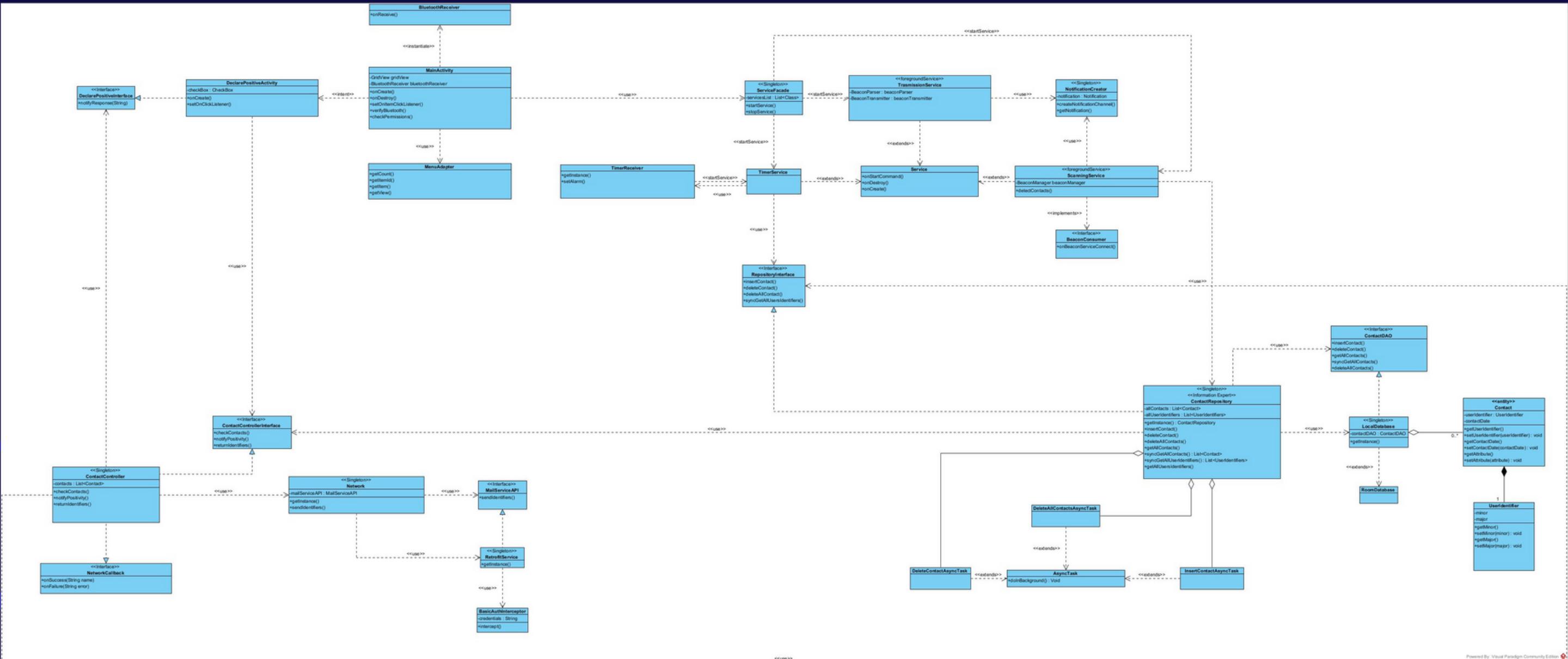


Diagramma delle classi

Raffinato

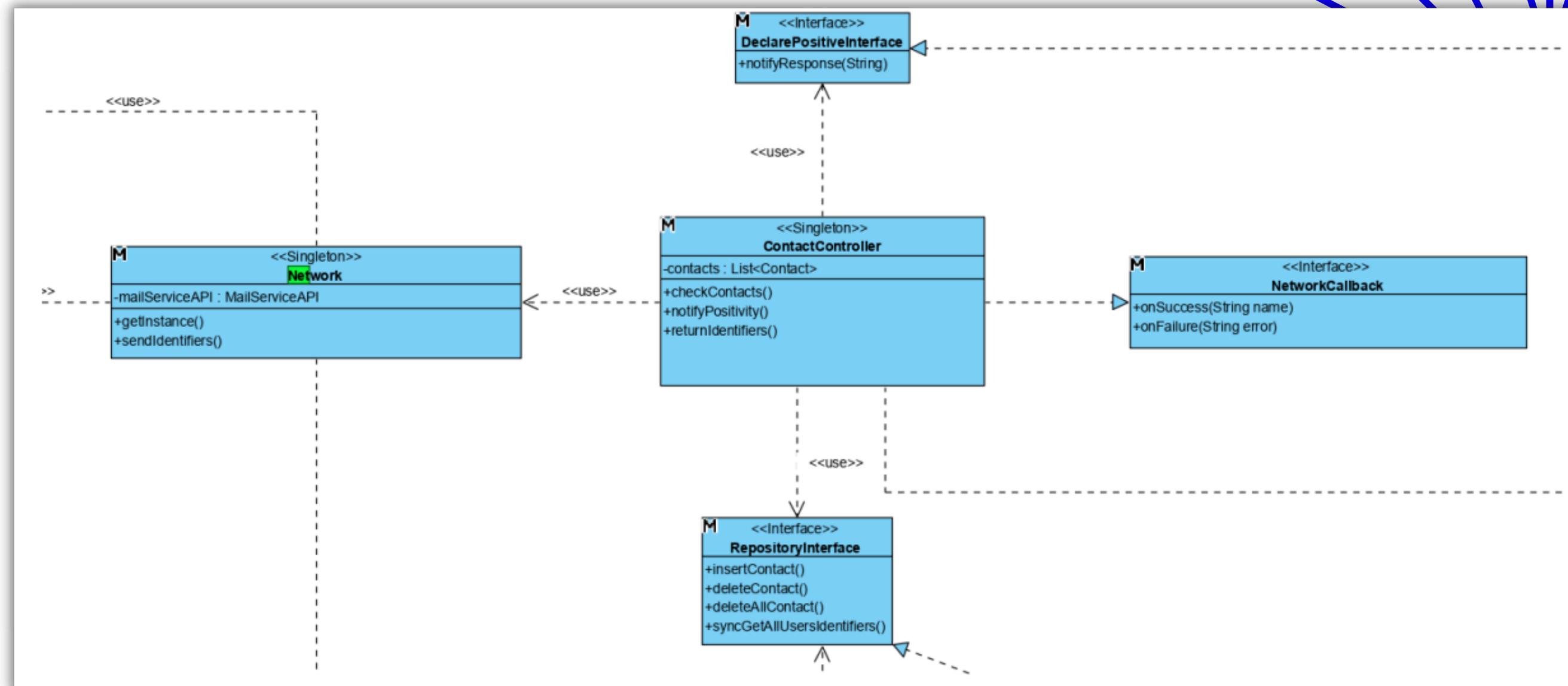


Class Diagram



Class Diagrams

E' stato utilizzato un **Session Controller**, che ha come unico ruolo quello di coordinare l'esecuzione di una sequenza di operazioni per implementare un caso d'uso o uno scenario. Si è cercato di tenere la **View** lontana da qualunque responsabilità che non sia quella di mostrare dati all'utente

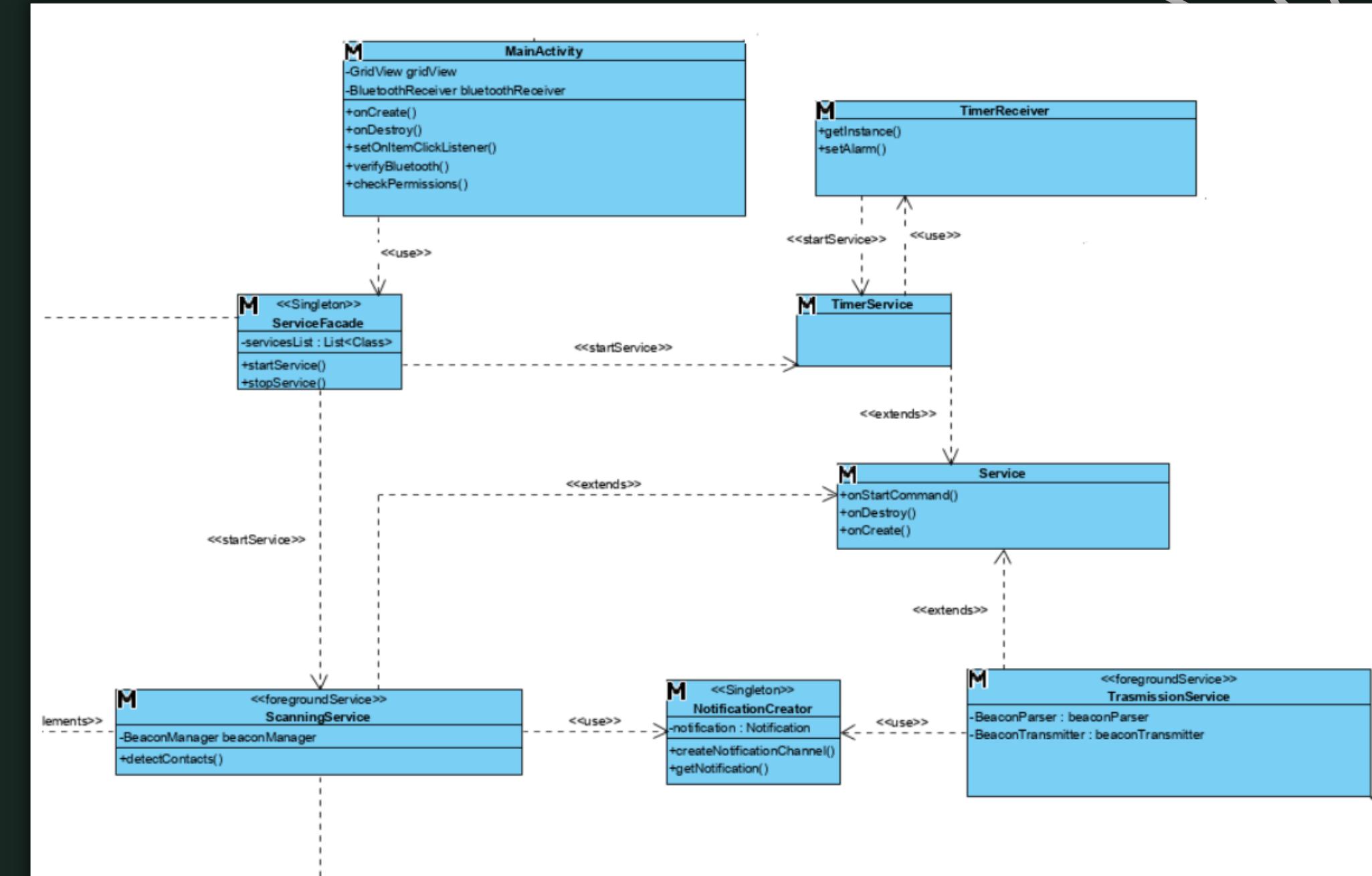


Class Diagrams

Si è scelto di utilizzare implementare un **Facade Pattern** per encapsulare i servizi in maniera tale da:

- semplificarne l'utilizzo da parte del sistema
- disaccoppiare l'activity dai servizi

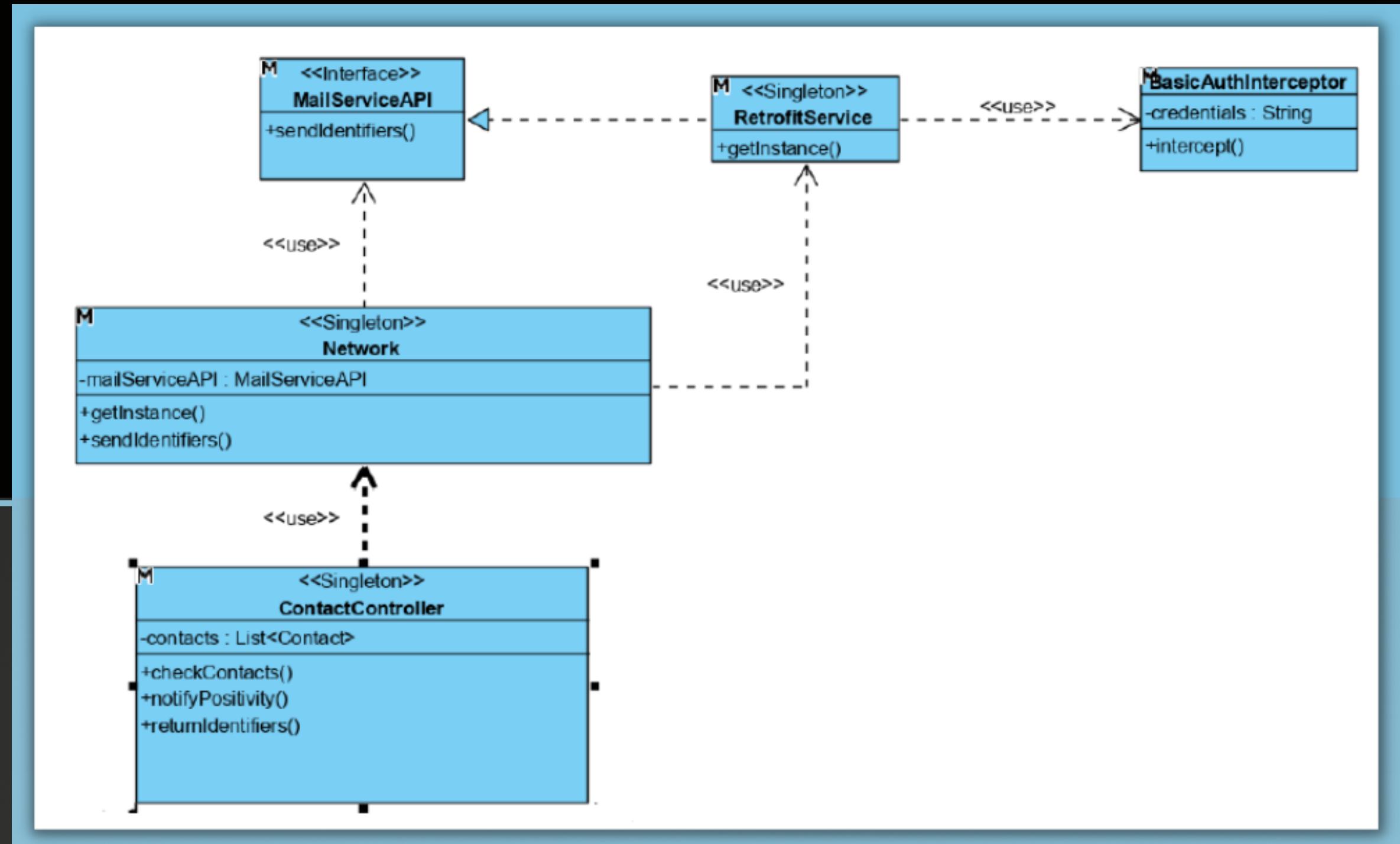
Le funzioni di scansione, trasmissione ed eliminazione di contatti obsoleti sono state implementate come **Foreground Service**



Class Diagrams

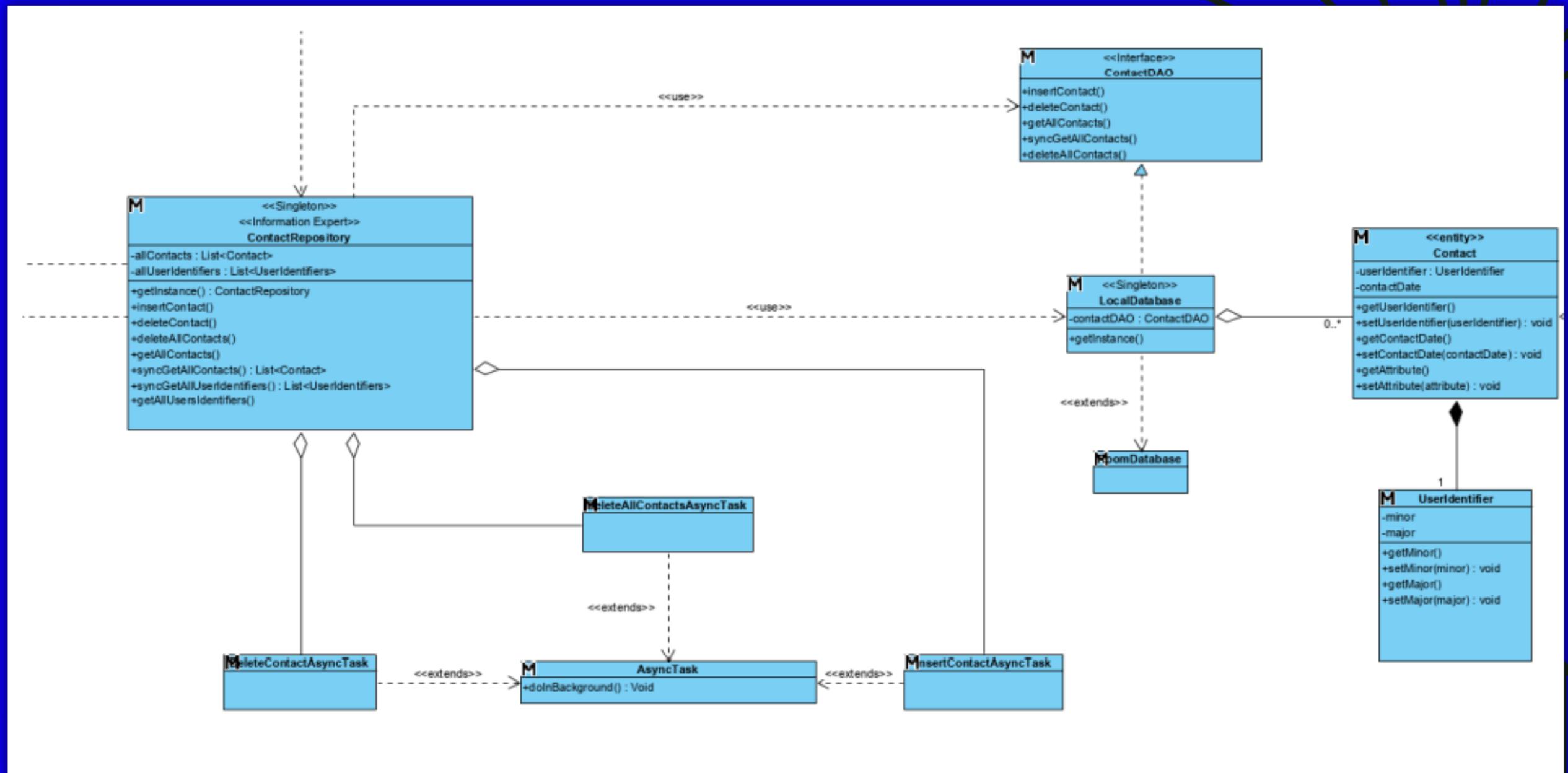
La classe Network accoppare in un unico oggetto le responsabilità relative all'interfacciamento con la rete evitando di assegnare responsabilità al controller fuori dalla sua competenza

È stato utilizzato il pattern Observer affinché venisse notificato alla Repository il cambiamento dei dati all'interno del database, che in tal modo è costantemente aggiornata.



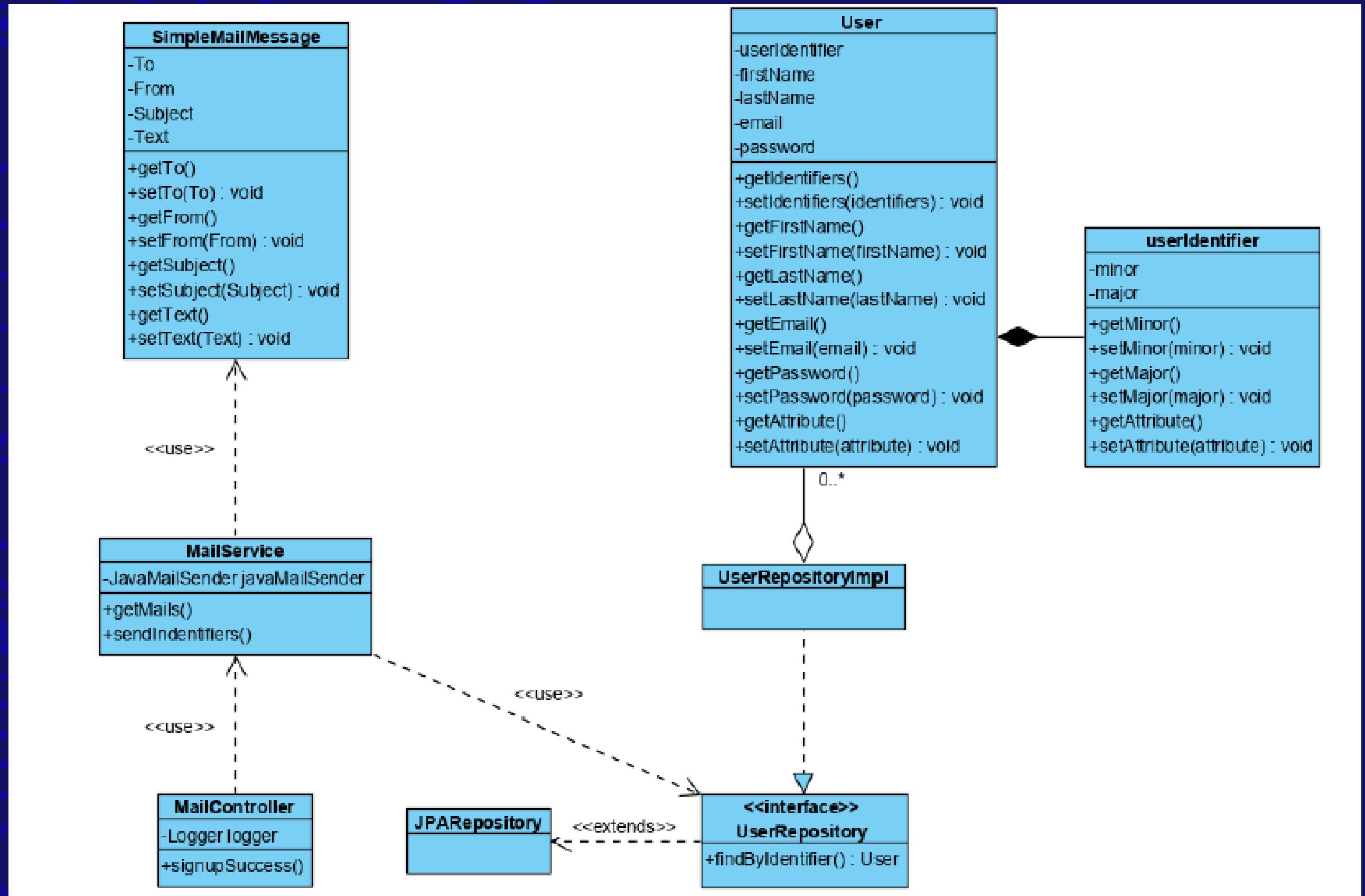
Class Diagrams

Il framework utilizzato per gestire la persistenza dei contatti sul dispositivo (Room) ha consentito di definire un'interfaccia (ContactDAO) attraverso cui interagire con il Database interno, evitando di implementare la classe concreta.



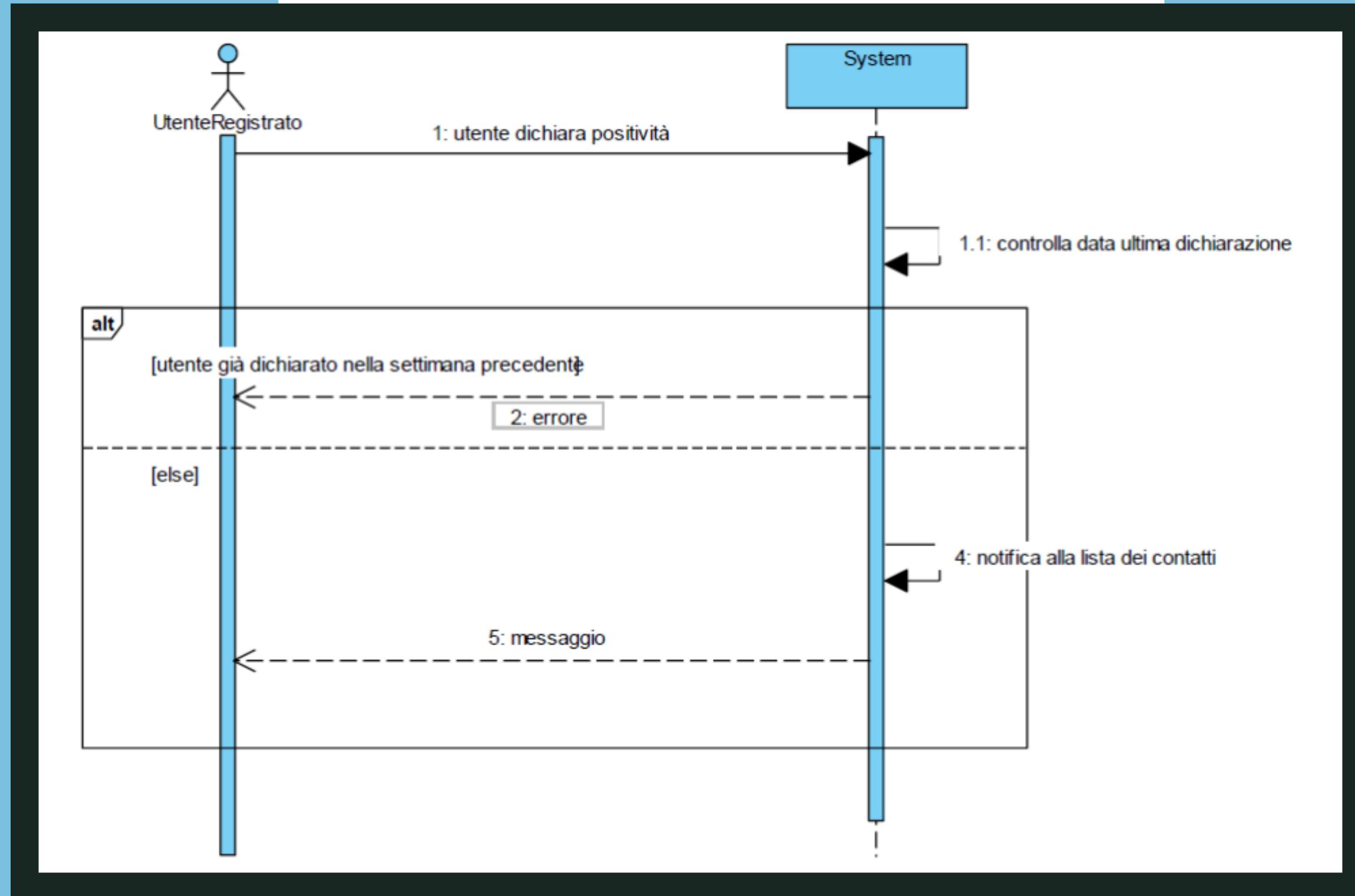
Class Diagrams

In questo Class Diagram mostriamo le classi e le relazioni tra di esse presenti nel Web Service. Il web service è stato implementato utilizzando il framework Spring e pertanto la sua struttura e la sua progettazione sono state vincolate dal framework stesso.



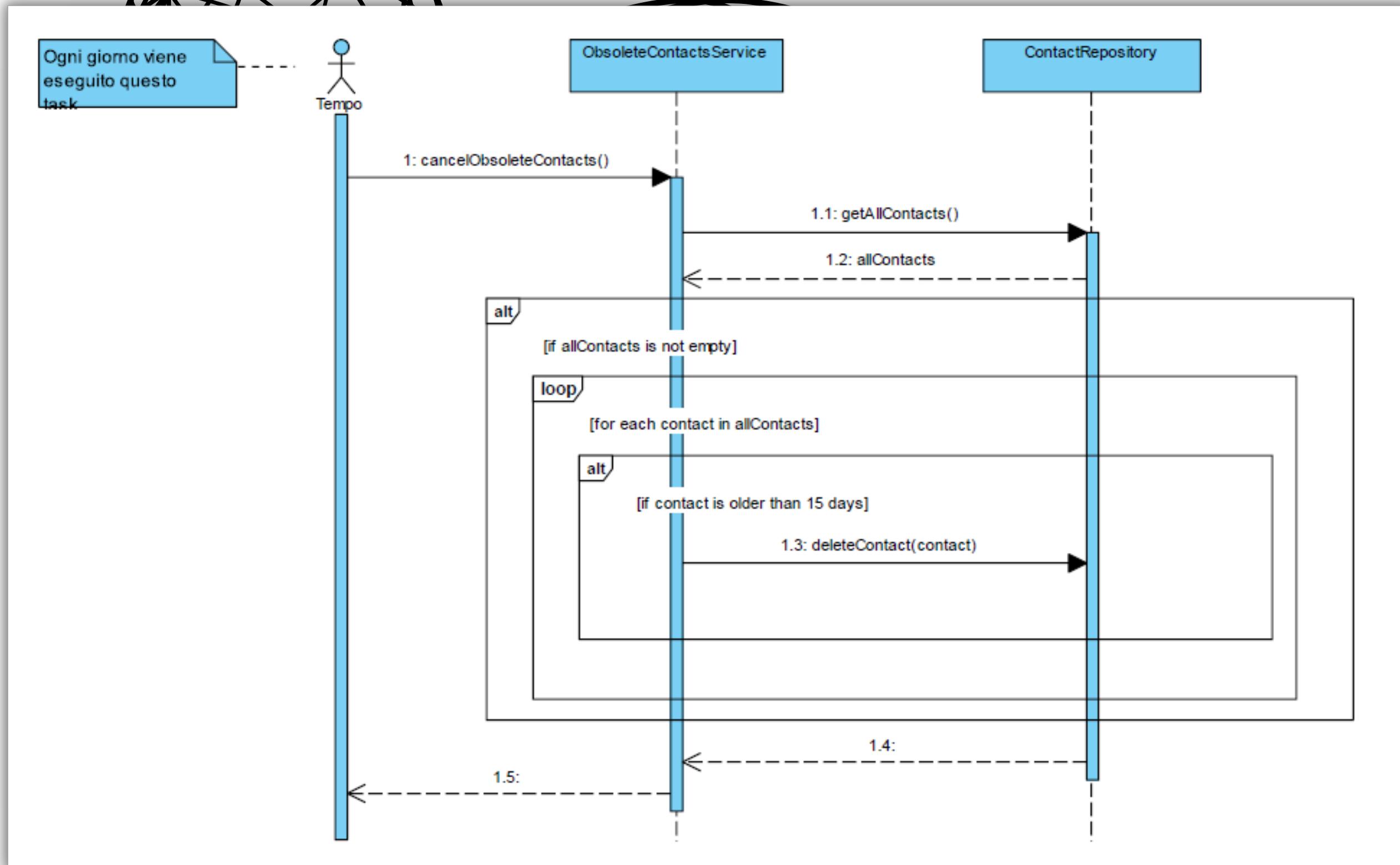
Sequence Diagram

Dichiarazione di positività

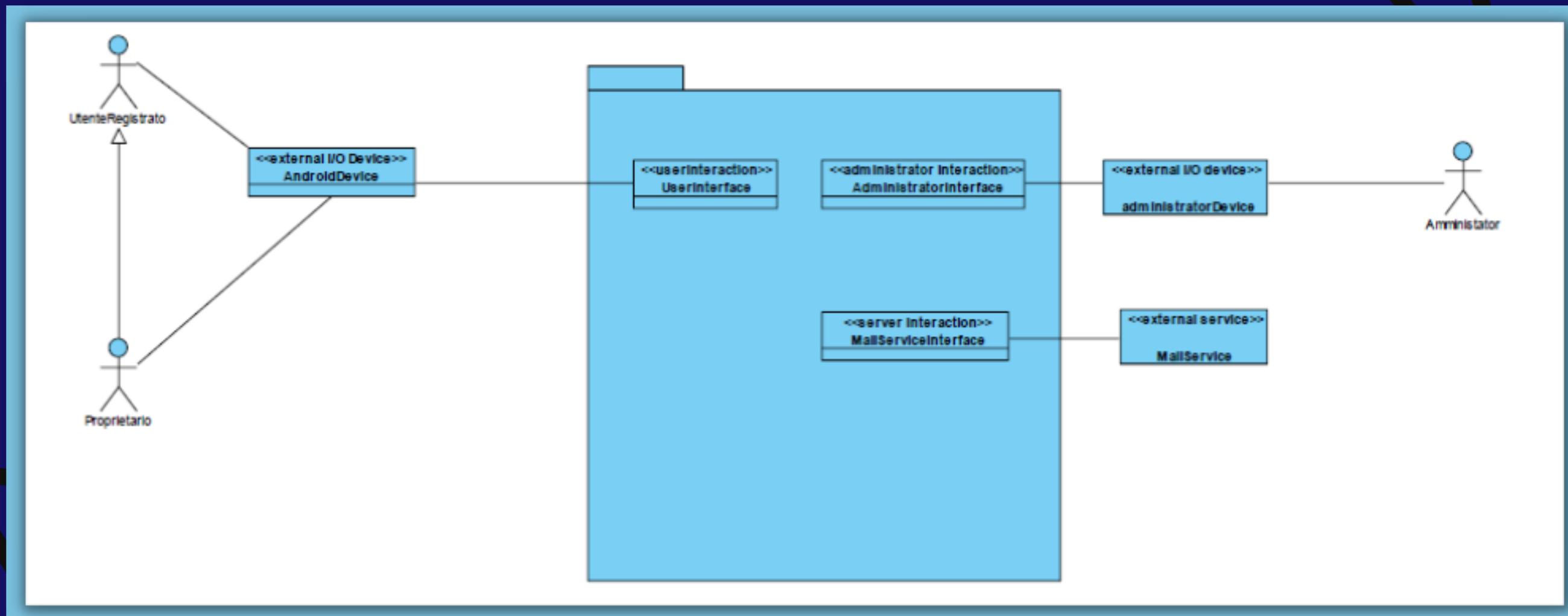


Sequence Diagram

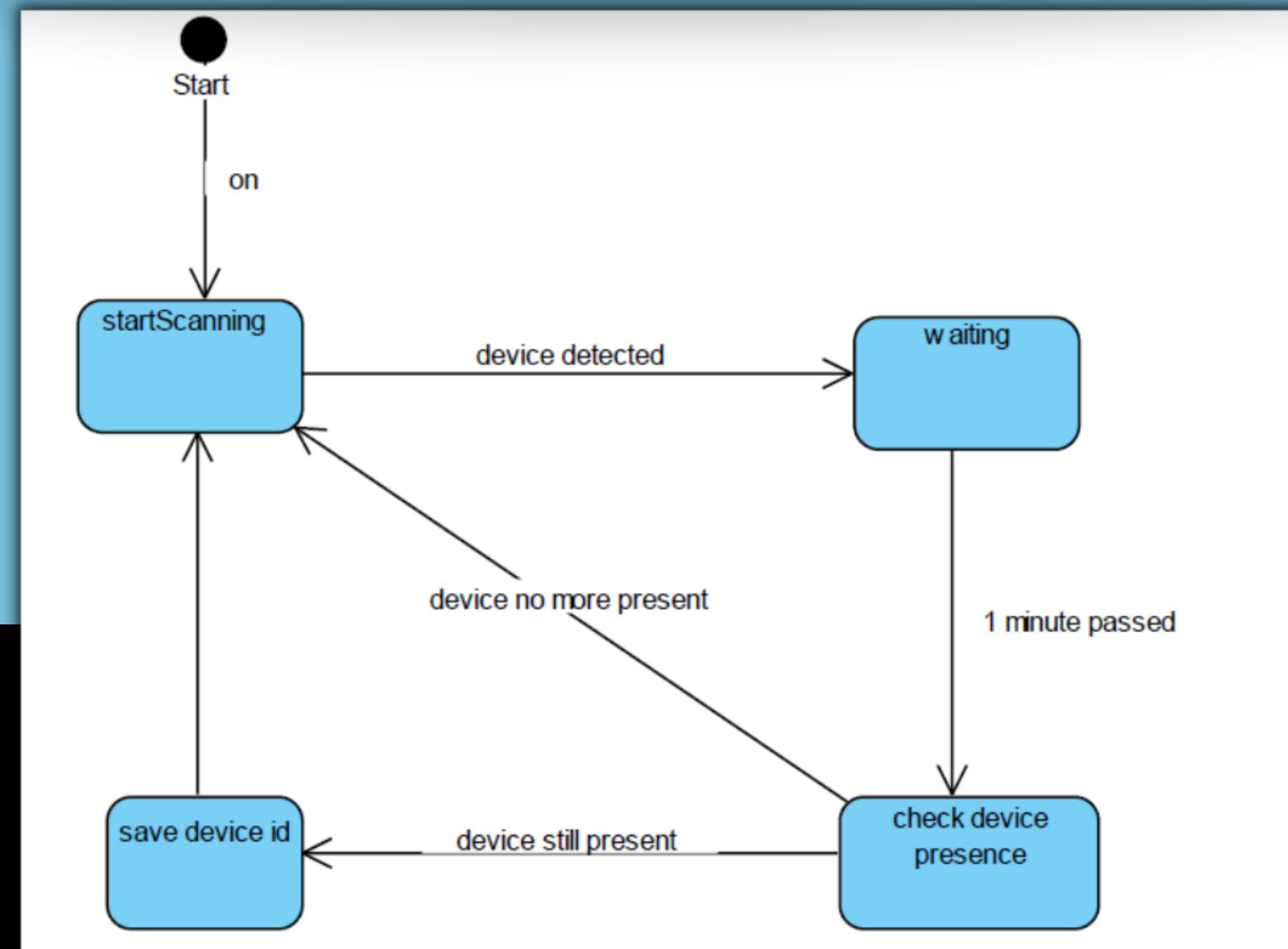
Cancellazione contatti obsoleti



Context Diagram

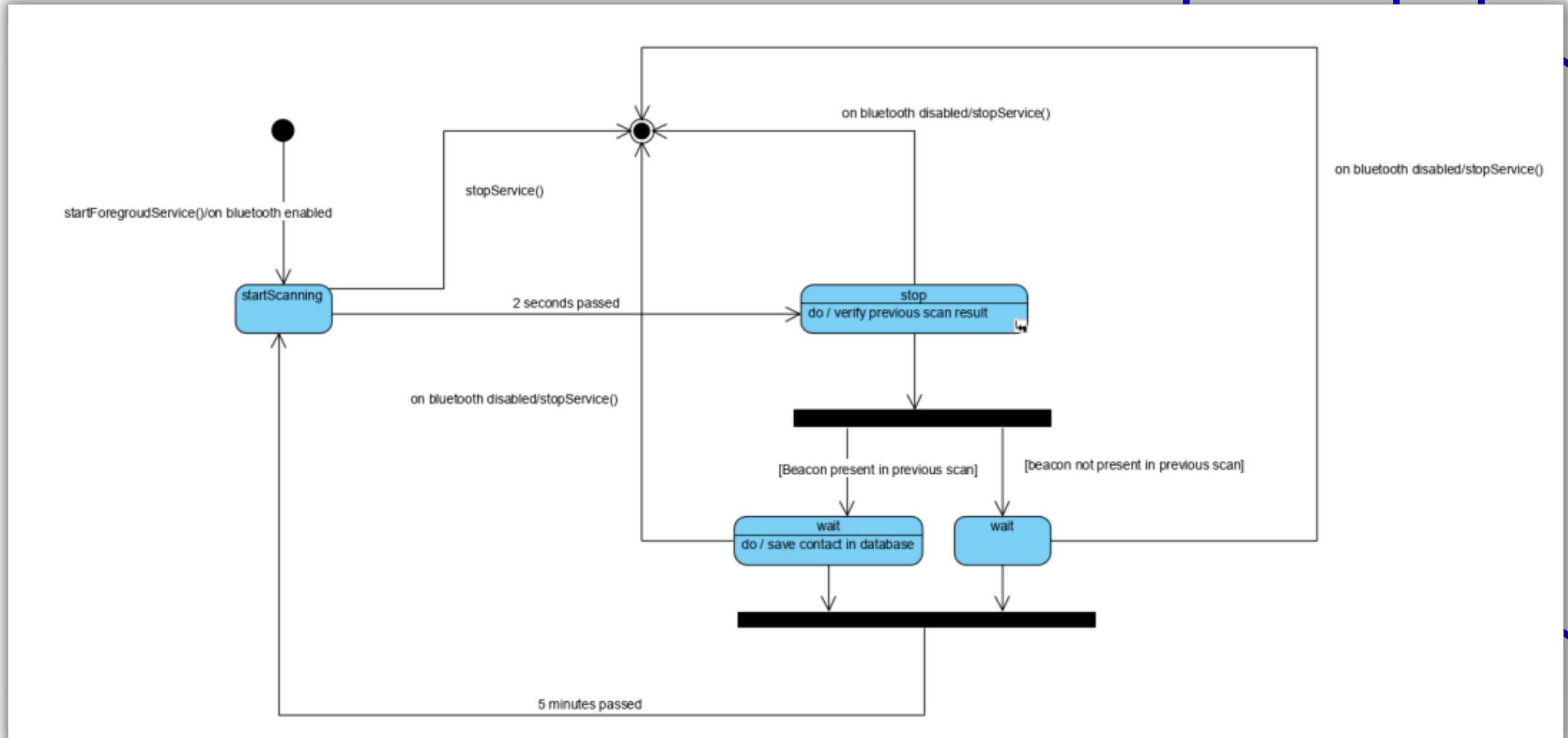


State Chart Diagram



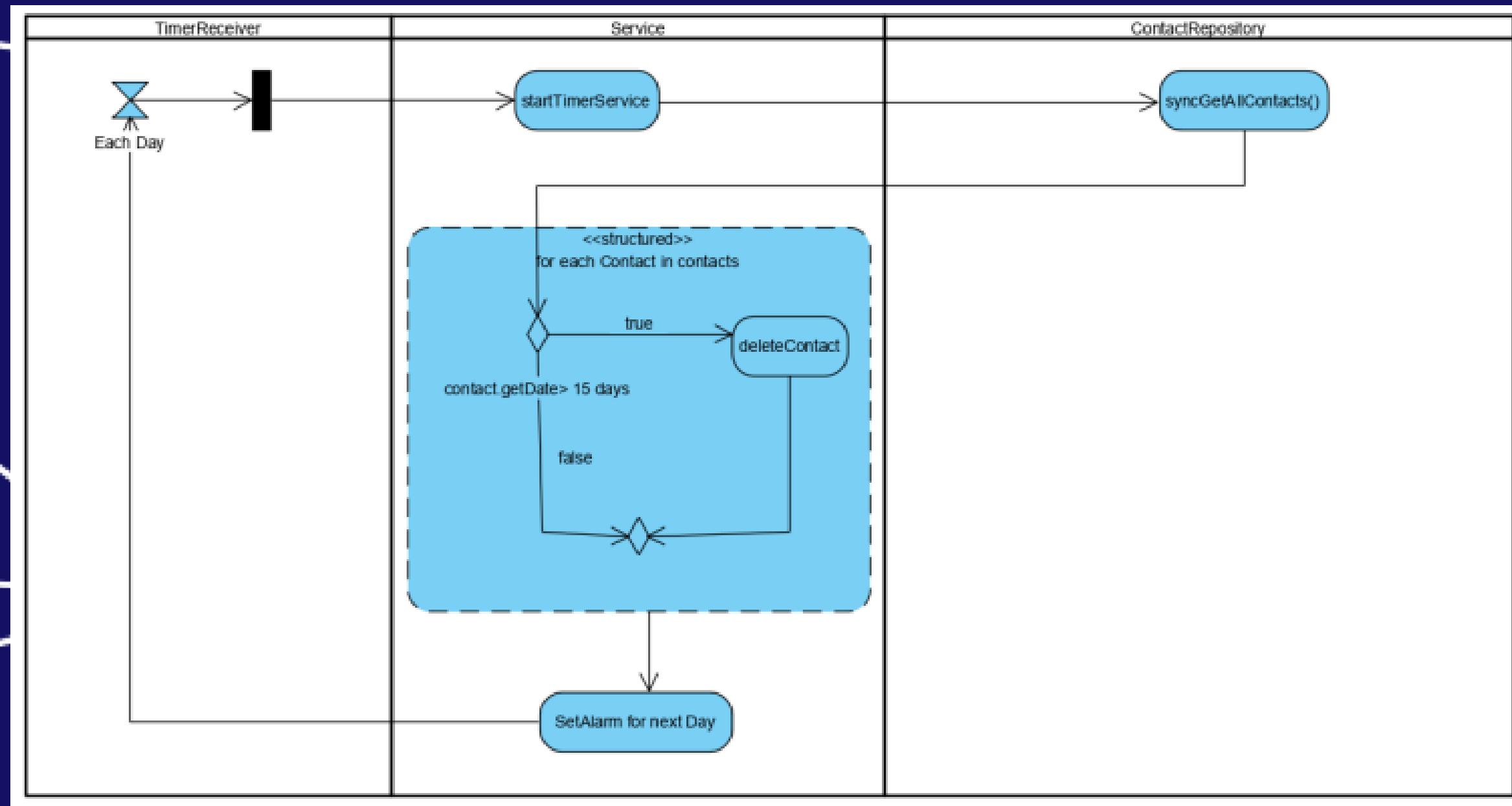
State Chart Diagram

Algoritmo di scansione
dei dispositivi ed il
salvataggio dei contatti.



Activity Diagram

Funzione di cancellazione contatti





Grazie per
l'attenzione.

Garofalo Luigi
Salvatore Niccolò
Torre Luca
Riccetti Nunzio Francesco