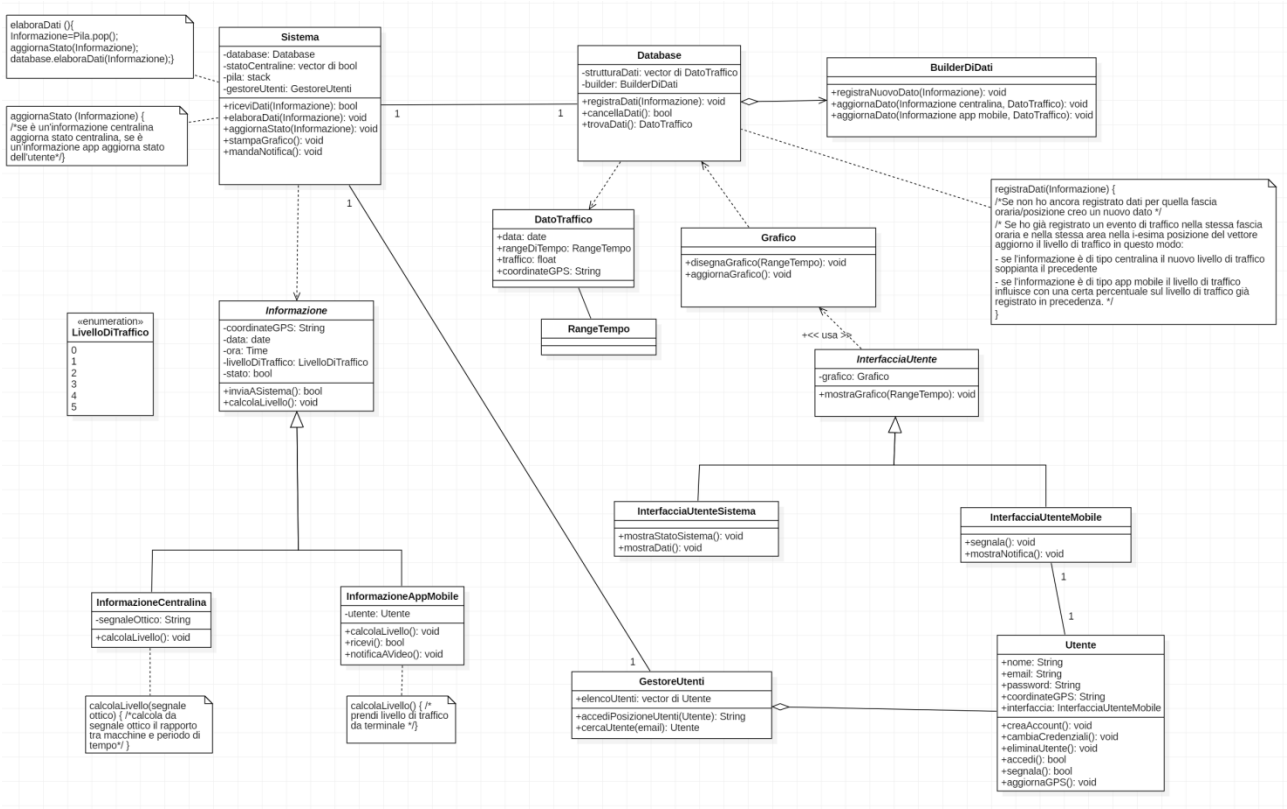
Documento Design UML

Nella stesura del progetto UML si è cercato di concentrarsi sulle parti dell'elaborato che necessitassero di più chiarezza; nello specifico si è voluto affrontare il problema dello scambio dei dati evidenziandone i passaggi con l'ausilio di diagrammi come il Sequence diagram o l'Activity diagram. Un altro obiettivo è stato quello di cercare di mantenere l'uniformità e la coerenza tra i grafici progettando per primo il Class diagram perché ci si potesse riferire ad esso nella stesura di tutti gli altri diagrammi.

CLASS DIAGRAM



Per formare il Class diagram si è deciso di partire da due entità fondamentali ovvero la classe Sistema e la classe Informazione; da quest'ultima ereditano due sottoclassi, Informazione Centralina ed Informazione App Mobile.

Si è inoltre voluto specificare una classe Utente che gestisse i metodi relativi al cliente dell'app mobile, dalla creazione di un account, alla segnalazione del traffico, all'accesso tramite credenziali. Quest'ultima operazione è resa possibile da un'altra classe, la classe Gestore Utenti, appartenente al Sistema, che tiene traccia dei clienti e della loro posizione GPS e implementa al suo interno un algoritmo di ricerca degli stessi.

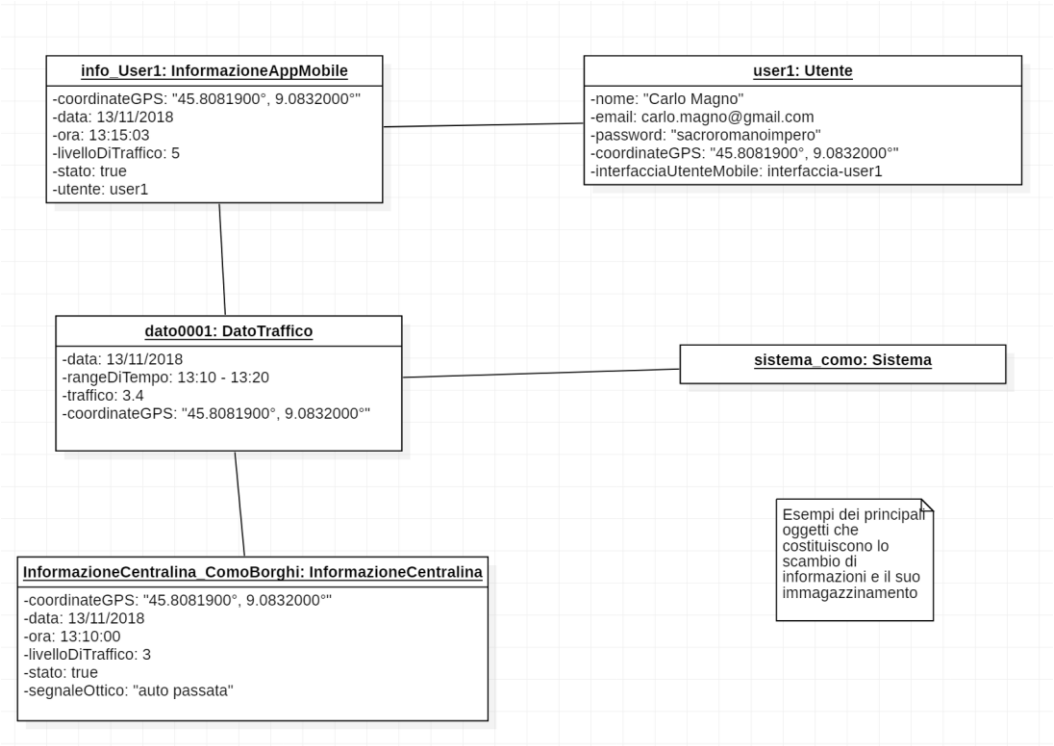
Il Sistema ha come attributo anche la classe Database utilizzata per organizzare ed ordinare i dati. Per facilitare questo processo si è deciso di utilizzare una classe Builder che si occupi dell'effettiva formazione di un'unità di dati relativi al traffico di una determinata zona e di una determinata fascia oraria, che il Database si occuperà quindi di salvare nella propria struttura dati. Questa unità è stata denominata DatoTraffico e ha tra i suoi attributi un range temporale (RangeTempo). La scelta di utilizzare la classe builder è stata giustificata dal desiderio di voler rendere l'implementazione del codice per la creazione effettiva di questi dati estraneo alle altre classi, e quindi rendere più semplici eventuali modifiche allo stesso.

Si è ipotizzato che il Builder attribuisca alle segnalazioni utente un peso minore rispetto alle informazioni fornite dalla centralina, qualora ci sia un sovrapporsi di segnalazioni contraddittorie nella stessa area e nella stessa fascia oraria.

Le classi Informazione Centralina e Informazione App Mobile implementano in modo separato il metodo Calcola Livello relativo alla classe astratta Informazione. Nella classe Informazione Centralina si fa ausilio di un algoritmo per calcolare il traffico dal segnale ottico ricevuto (funzionamento descritto nell'Activity diagram Centralina). Nella classe Informazione App Mobile si fa invece riferimento alla classe Utente, utilizzata per etichettare l'informazione sul traffico e collocarla nello spazio.
Alla classe Informazione nel suo complesso è stato inoltre collegato un set di valori chiamato Livello di Traffico che unifichi le informazioni sul traffico ed identifichi la gravità della congestione.

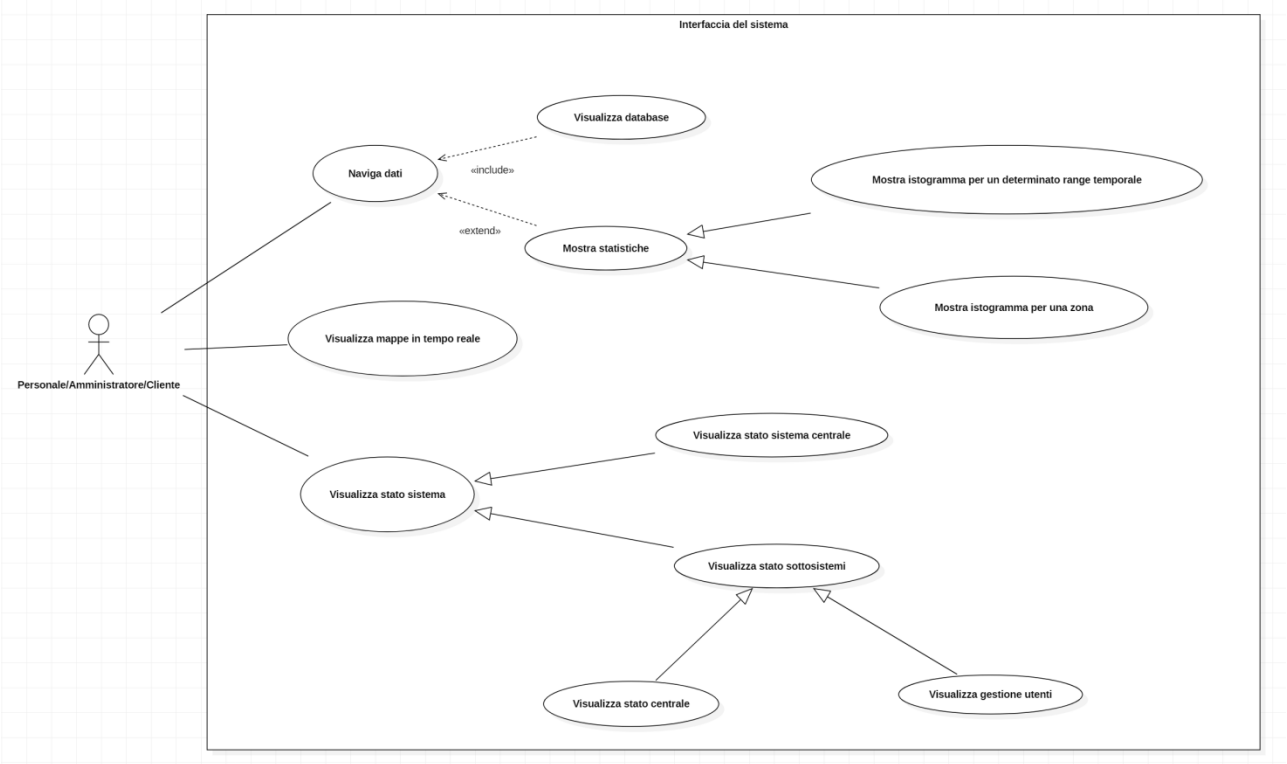
Per l'interfaccia grafica abbiamo ipotizzato una classe astratta collegata alla classe Grafico, da cui implementare separatamente un'interfaccia utente per il cliente app mobile e una per l'amministratore del server.

OBJECT DIAGRAM



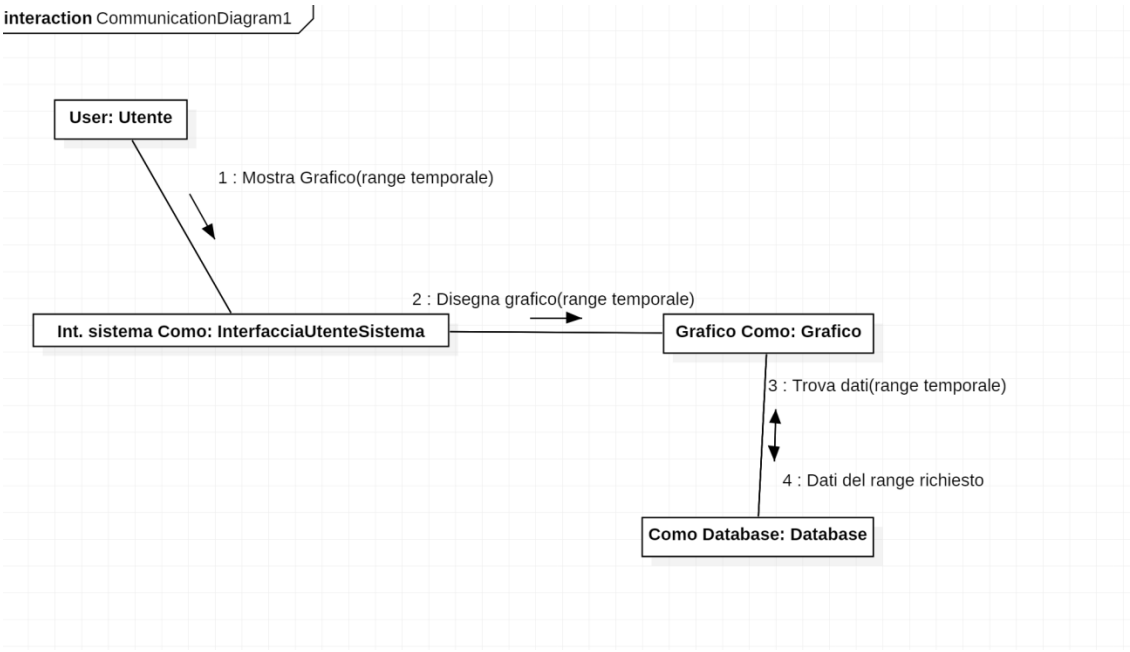
L'Object diagram si occupa di esemplificare alcune delle classi presentandole come oggetti.

USE CASE



La decisione di utilizzare uno Use Case diagram, per mostrare le operazioni disponibili all'amministratore del server principale, è nata dal voler rendere il più chiaro possibile a quali funzionalità egli abbia accesso e come esse si presentino all'interno della classe Interfaccia Utente del Sistema.

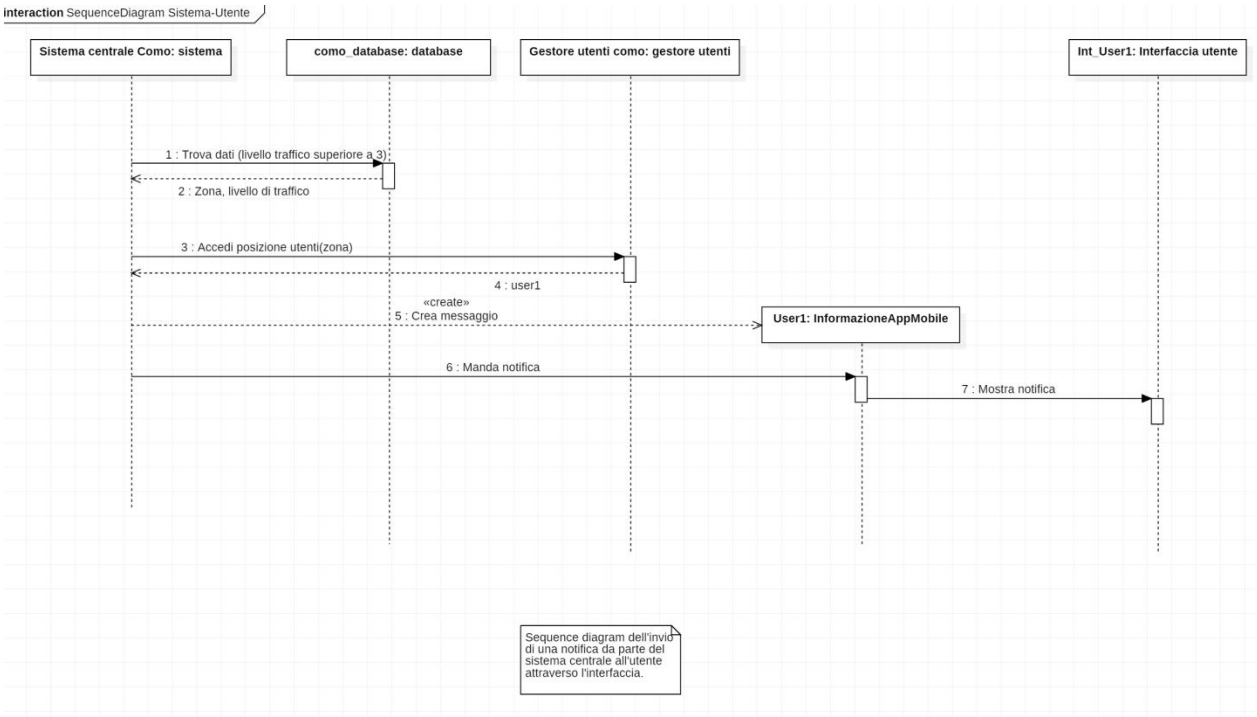
COMMUNICATION DIAGRAM



Con il Communication diagram si entra nello specifico della sequenza di chiamate effettuate tra l'Interfaccia Utente del Sistema, la classe Grafico e la classe Database. Queste chiamate sono volte

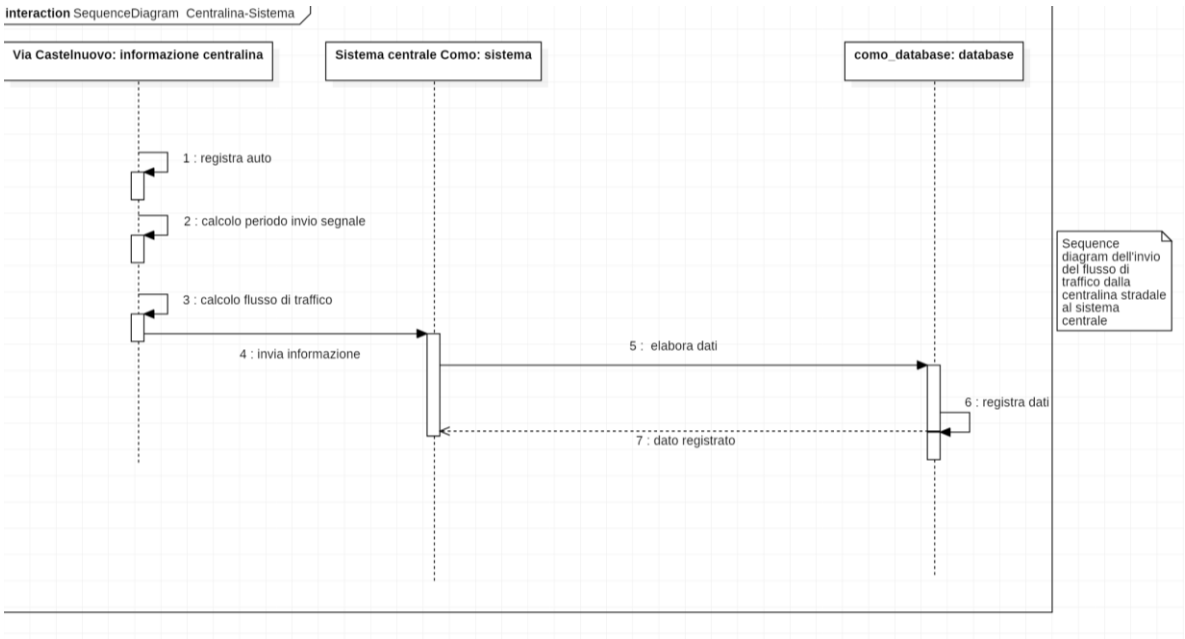
ad ottenere una rappresentazione grafica dei dati, contenuti nel Database, senza che l'utente abbia direttamente accesso ad essi.

SEQUENCE DIAGRAM SISTEMA-UTENTE



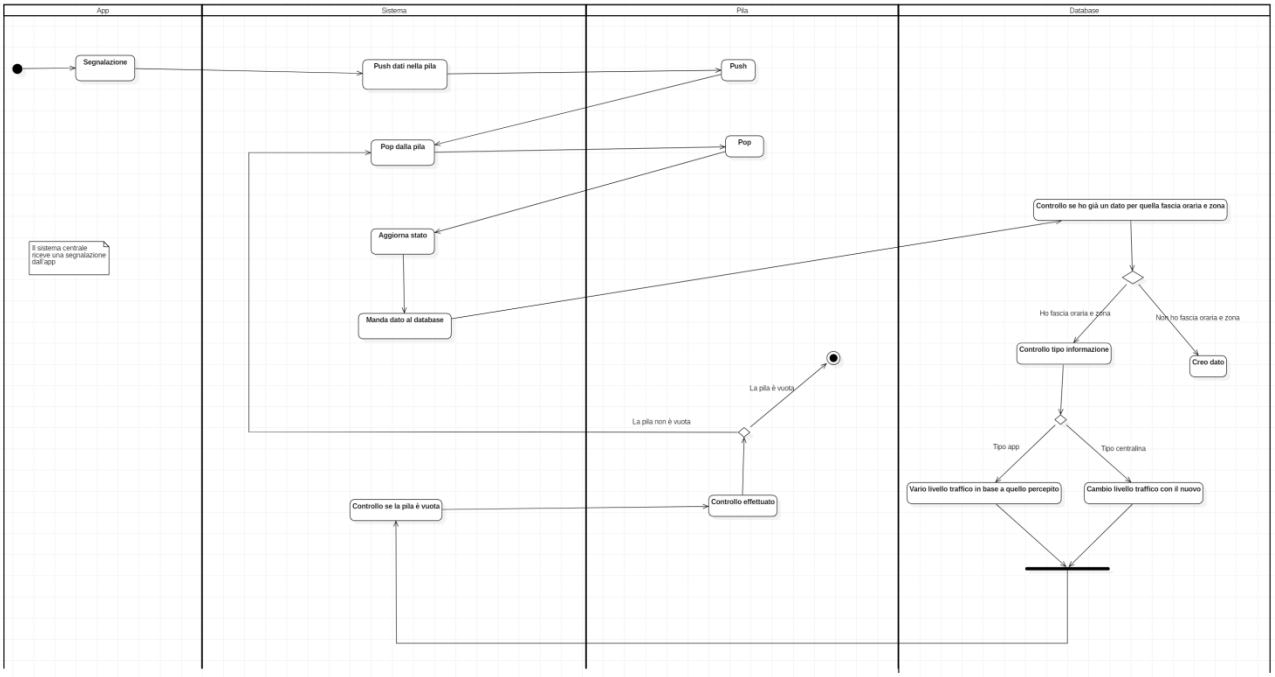
Come già accennato si è voluto utilizzare i Sequence diagram per mostrare il passaggio delle informazioni. Nello specifico in questo diagramma si vuole mostrare la sequenza di passaggi che intercorrono tra il Sistema e le classi ad esso collegato prima dell'invio di una notifica all'app mobile, che si presenta come un oggetto Informazione App Mobile. Esso viene ricevuto sotto forma di server push e mostrato a video all'utente.

SEQUENCE DIAGRAM CENTRALINA-SISTEMA



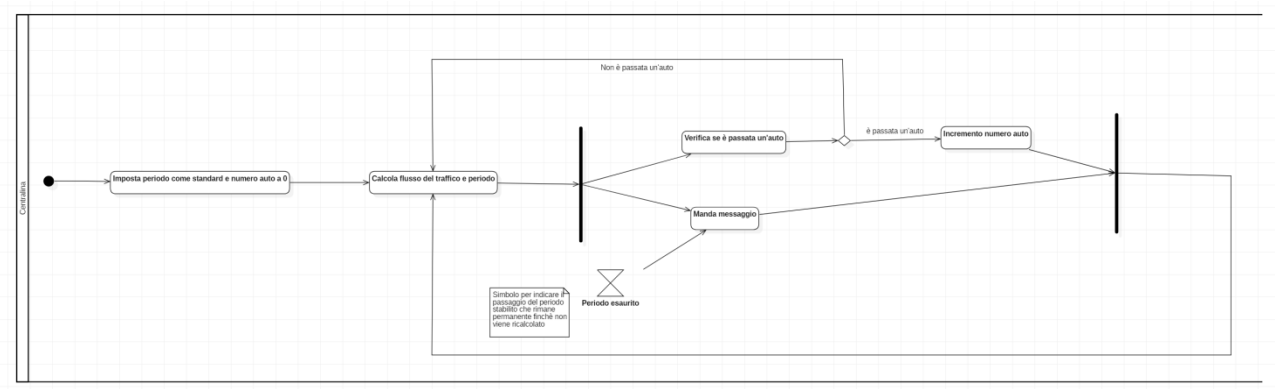
In questo diagramma si è analizzato l'invio delle informazioni riguardo il livello di traffico registrato da parte della centralina e la susseguente registrazione della stessa all'interno del Database. Si è omesso l'utilizzo della stack usata dal Sistema lasciandola implicita.

ACTIVITY DIAGRAM SISTEMA CENTRALE



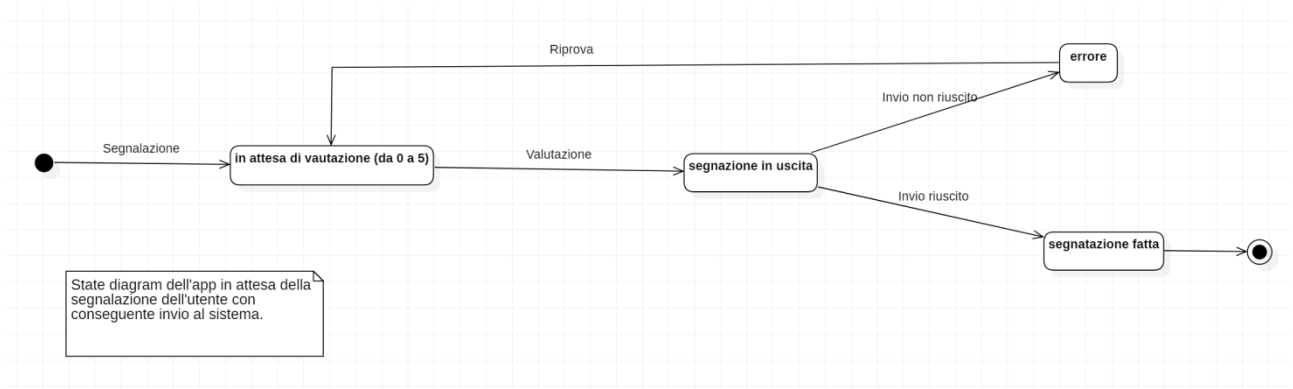
Si è invece voluto mostrare la pila all'interno dell'Activity Diagram riguardante il Sistema al fine di chiarificarne il comportamento nel caso in cui esso riceva più segnalazioni in contemporanea. Nello specifico in questo diagramma abbiamo analizzato l'eventualità di una segnalazione da parte di un utente app mobile.

ACTIVITY DIAGRAM CENTRALINA



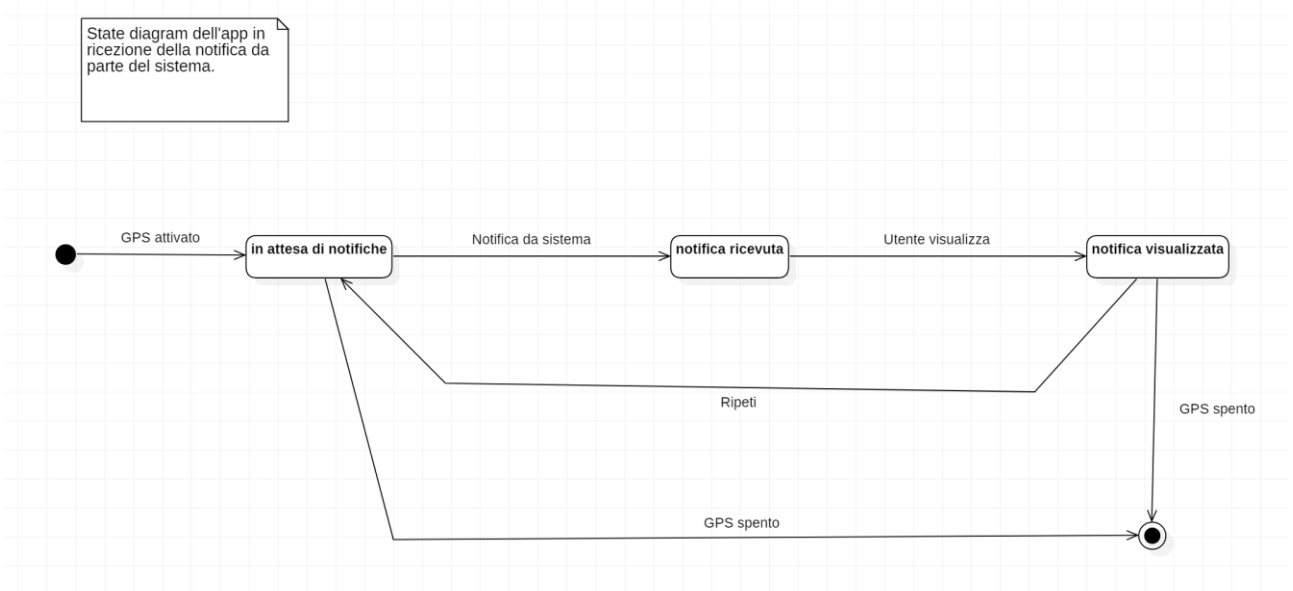
Si è poi deciso di utilizzare un ulteriore Activity diagram per chiarificare il comportamento della centralina e il modo in cui essa aggiorni la variabile periodo (indicante il tempo che intercorre tra una segnalazione al Sistema all'altra). All'attivazione della centralina, infatti, il periodo viene impostato con un tempo standard, e rimarrà invariato finché il passaggio di un'auto non farà riaggiornare il conteggio auto della centralina (impostato inizialmente a 0) e di conseguenza anche il periodo.
Il simbolo della clessidra (Accept time event action) sta infatti ad indicare il passaggio del tempo previsto per l'invio della segnalazione.

STATE DIAGRAM APP IN SEGNALAZIONE



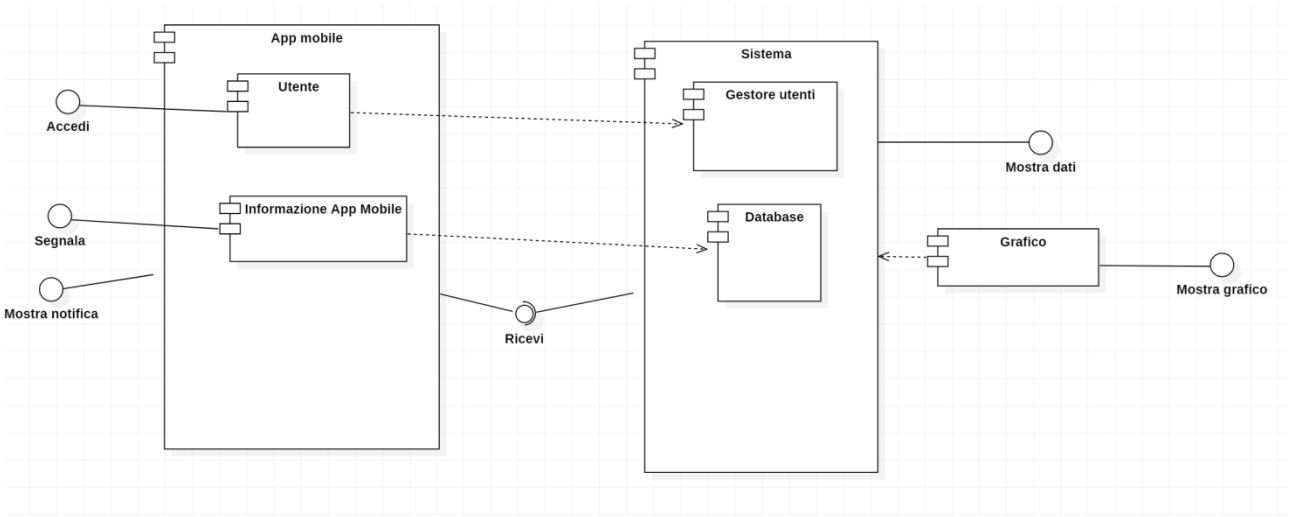
In questo State diagram vengono messi in evidenza i vari stati assunti dall'applicazione mobile quando l'utente richiede di fare una segnalazione (si è omessa, in questo caso, l'interfaccia grafica attraverso la quale avviene ciò). Si è scelto di mostrare anche la richiesta, da parte dell'app all'utente, di specificare la gravità del traffico da egli incontrato (livello di traffico) e si è considerata l'eventualità di un errore nell'invio della segnalazione, al fronte della quale l'app rinnova la richiesta (nel caso la gravità della congestione fosse cambiata nel frattempo) e ritenta l'invio.

STATE DIAGRAM APP IN RICEZIONE



La seconda funzione principale dell'app è di ricevere notifiche push provenienti dal sistema; questo diagramma mostra gli stati da essa assunti dall'attivazione del GPS, quando è cioè possibile per il Sistema notificare l'utente.

COMPONENT DIAGRAM



Il Component diagram è stato utile per mappare l'interezza del microcosmo Sistema-App, evidenziando la presenza delle interfacce e anticipando la distinzione Server-Client analizzata successivamente nel Deployment diagram.

DEPLOYMENT DIAGRAM