

SAMUELE MOSCATELLI, NICOLÒ PINCIROLI,  
ANDREA POZZOLI

## GESTIONE DEL TRAFFICO



DOCUMENTAZIONE DEL PROGETTO DI  
PROVA FINALE

POLIMI

INGEGNERIA INFORMATICA

A.A. 2018 - 2019

Samuele Moscatelli, Nicolò Pincioli,  
Andrea Pozzoli,  
*Gestione del traffico*  
*Documentazione del progetto di prova finale*

E-MAIL:  
Samuele Moscatelli - sem.mosca97@libero.it  
Nicolò Pincioli - nicolopinci.1997@gmail.com  
Andrea Pozzoli - pozzoliandrea97@gmail.com



**Figura 1:** Un esempio di traffico

## INDICE

<b>1 DESIGN . . . . .</b>	<b>1</b>
1.1 Use case Modeling	2
1.1.1 Use case diagram	2
1.2 Structural Modeling - Static diagrams	3
1.2.1 Class diagram	3
1.2.2 Object diagram	9
1.3 Structural Modeling - Implementation diagrams	10
1.3.1 Deployment diagram	10
1.3.2 Component diagram	12
1.4 Dynamic Modeling - Interaction diagrams	14
1.4.1 Sequence diagrams	14
1.4.2 Collaboration diagram	17
1.5 Dynamic Modeling - Statechart diagrams	18
1.6 Dynamic Modeling - Activity diagrams	22
<b>BIBLIOGRAFIA . . . . .</b>	<b>26</b>

# 1

## DESIGN

I diagrammi sono stati studiati, sviluppati e presentati seguendo il seguente schema:

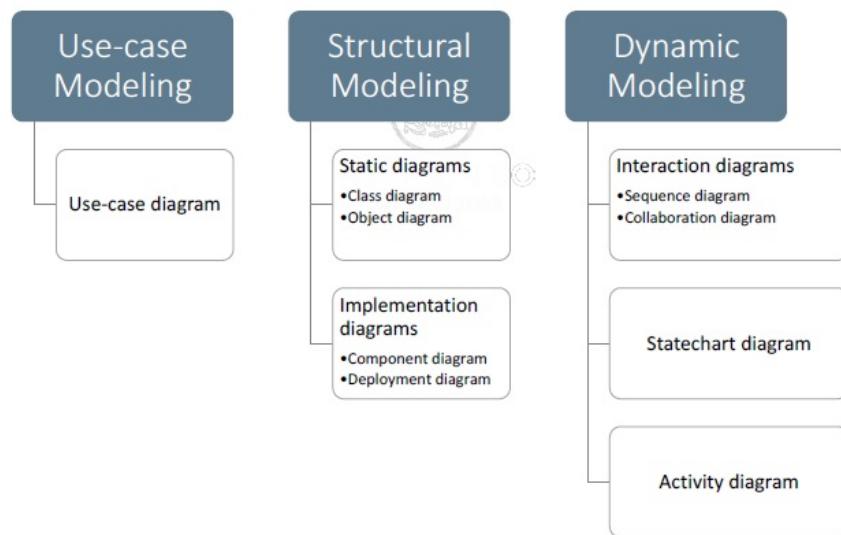


Figura 2: Schema UML

## 1.1 USE CASE MODELING

### 1.1.1 Use case diagram

Lo use case diagram è una tecnica utilizzata per identificare i requisiti funzionali di un sistema e si basa sulla descrizione delle interazioni tipiche tra gli utenti e il sistema.

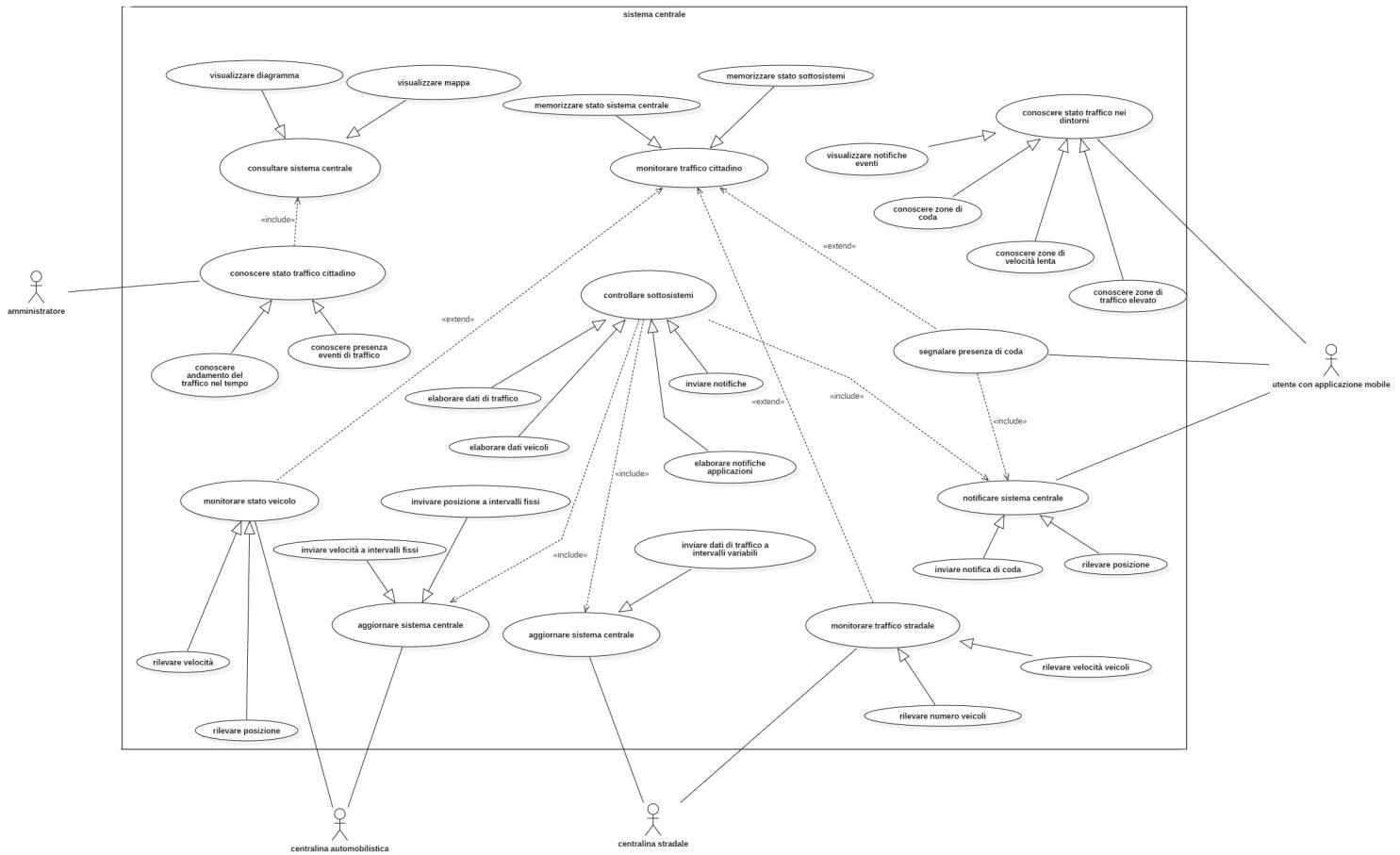


Figura 3: Use case diagram

In questo use case diagram, in particolare, come attori sono stati scelti l'amministratore, la centralina stradale, la centralina automobilistica e l'utente che utilizza l'applicazione. L'amministratore ha il compito di conoscere lo stato del traffico consultando il sistema centrale, il quale gli presenta i dati graficamente con un diagramma e una mappa. Le centraline sono state inserite come attori in quanto assumono un ruolo attivo nei confronti del sistema centrale. Esse infatti monitorano i veicoli su cui sono installate (nel caso delle centraline automobilistiche) e il traffico in una determinata posizione (per le centraline stradali), e inviano i dati raccolti al sistema centrale. L'utente e l'applicazione sono stati uniti in un unico attore in quanto è l'utente

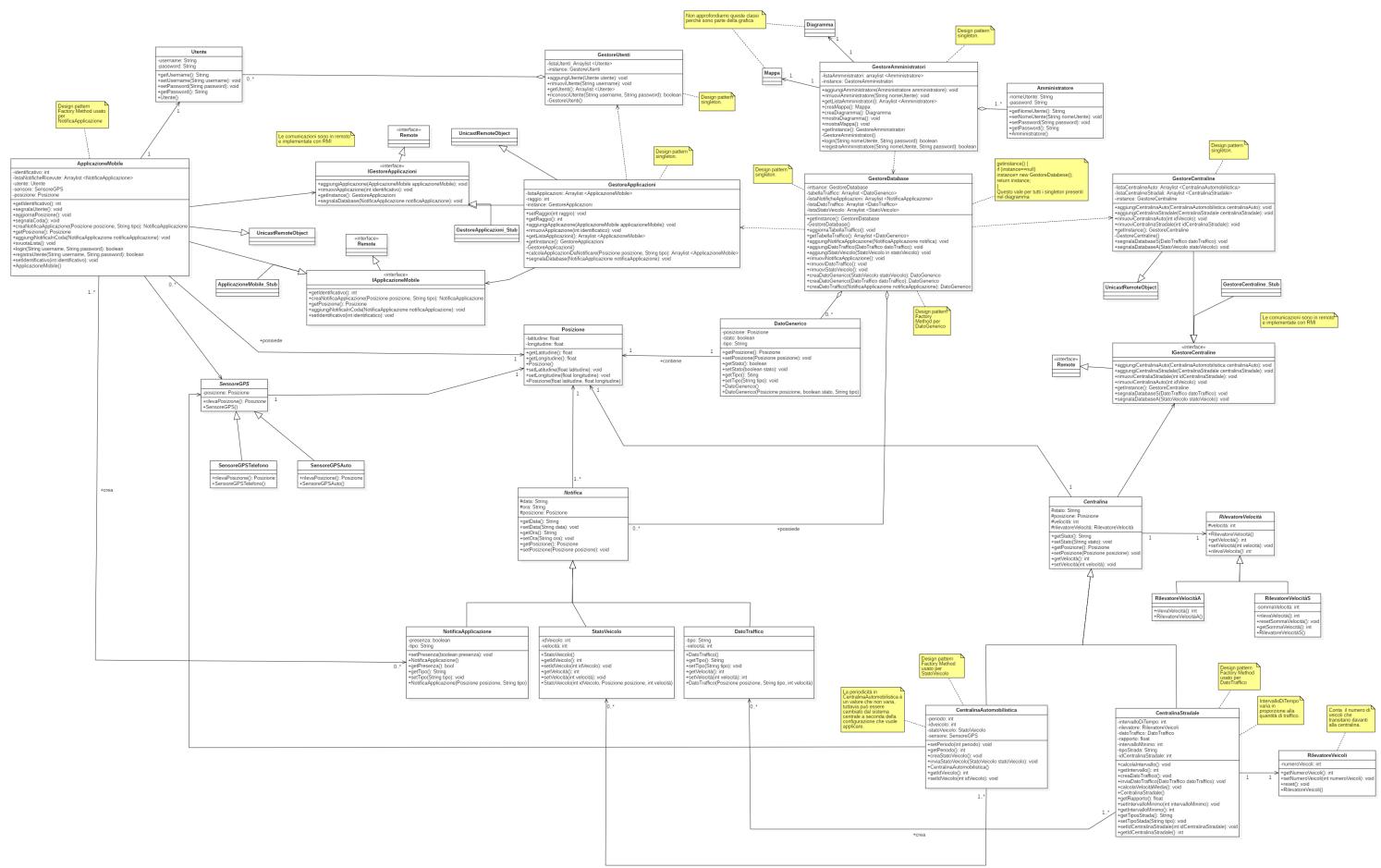
che utilizza l'applicazione a interagire con il sistema. Le interazioni consistono nel segnalare al sistema centrale la presenza di coda in una determinata posizione, e apprendere dal sistema quali sono le zone che presentano eventi di traffico.

L'insieme delle interazioni tra gli attori e il sistema centrale portano al monitoraggio completo del traffico cittadino e al controllo dei singoli sottosistemi da parte del sistema centrale.

## 1.2 STRUCTURAL MODELING – STATIC DIAGRAMS

### 1.2.1 Class diagram

Il class diagram descrive i tipi degli oggetti che fanno parte di un sistema e le varie tipologie di relazioni statiche tra di essi. Inoltre, il class diagram mostra le proprietà e le operazioni di ogni classe e i vincoli che si applicano ai collegamenti tra gli oggetti.



**Figura 4:** Class diagram

In questo class diagram si è deciso di rappresentare il sistema complessivo. In particolare si è suddiviso ciò che nei requisiti era stato individuato come sistema centrale in cinque classi, ognuna adibita a svolgere uno specifico compito di gestione. Due di esse, GestoreApplicazioni e GestoreCentraline, si occupano, rispettivamente, di gestire le applicazioni mobili e le centraline, sfruttando l'utilità dell'interfaccia di comunicazione remota RMI fornita dalla piattaforma Java per scambiare dati con queste ultime. Le restanti tre classi si occupano invece della gestione delle informazioni relative allo stato del traffico (GestoreDatabase), agli utenti che utilizzano le applicazioni mobili (GestoreUtenti) e agli amministratori di sistema (GestoreAmministratori). Sono state poi sviluppate le classi centralina, applicazione, utente e amministratore, con i relativi componenti e i dati che creano. Di seguito vengono illustrate più nel dettaglio ciascuna delle classi individuate.

#### *GestoreDatabase*

La classe GestoreDatabase è costruita usando il design pattern singleton. Il suo compito è sostanzialmente quello di ricevere e memorizzare tutti i dati provenienti dai sottosistemi esterni (centraline e applicazioni mobili), elaborarli e unirli sotto forma di dati di un solo tipo. Per questo motivo, il GestoreDatabase presenta anche il design pattern Factory Method, utilizzato per la creazione del DatoGenerico.

#### *GestoreApplicazioni*

La classe GestoreApplicazioni è costruita usando il design pattern singleton. Essa è adibita alla gestione delle applicazioni mobili: in primo luogo deve salvare tutte le applicazioni che sono state installate e che sono effettivamente funzionanti; secondariamente, in caso di nuovo evento di traffico, deve calcolare le applicazioni che si trovano all'interno della zona circolare centrata nella posizione dell'evento (e con raggio dato) e notificarle; infine, deve agire da "tramite" tra l'applicazione mobile e il GestoreDatabase, ricevendo le segnalazioni dalle applicazioni e inoltrandole a quest'ultimo.

#### *GestoreCentraline*

La classe GestoreCentraline è costruita usando il design pattern singleton. Al pari del GestoreApplicazioni, il suo compito è quello di gestire le centraline e quindi conoscere tutte quelle installate ed effettivamente funzionanti e ricevere da esse i dati per poi inoltrarli al GestoreDatabase. A differenza del gestore delle applicazioni, il gestore centraline non deve occuparsi del processo inverso, ovvero quello di inviare notifiche a queste ultime.

### *GestoreUtenti*

La classe GestoreUtenti è costruita usando il design pattern singleton. Il suo compito è soltanto quello di salvare gli utenti registrati al servizio, memorizzando un elenco di oggetti di tipo Utente.

### *GestoreAmministratori*

La classe GestoreAmministratori è costruita usando il design pattern singleton. Al pari del GetoreUtenti ha il compito di memorizzare l'elenco di oggetti di tipo Amministratore. Oltre a questo compito, il GestoreAmministratore chiede i dati al GestoreDatabase per poter creare e aggiornare la mappa e il diagramma, che sono gli strumenti a disposizione dell'amministratore per visualizzare lo stato del traffico.

### *Applicazione mobile*

La classe ApplicazioneMobile si occupa di raccogliere le notifiche che il GestoreApplicazioni invia alle applicazioni (creando una nuova NotificaApplicazione e inserendola in una lista), e di creare una NotificaApplicazione quando l'utente schiaccia il bottone per segnalare il sistema centrale di un evento di traffico. Per svolgere queste funzionalità, ApplicazioneMobile è costruita utilizzando il design pattern Factory Method per creare NotificaApplicazione. Inoltre gestisce l'accesso e la registrazione degli utenti. Per quanto riguarda la comunicazione con il GestoreApplicazioni, essa avviene tramite RMI, mentre la rilevazione della posizione è svolta attraverso il sensore GPS del telefono su cui è installata.

### *Centralina (astratta)*

La classe astratta Centralina raccoglie gli attributi e i metodi in comune tra CentralinaStradale e CentralinaAutomobilistica. Per questo motivo i suoi attributi sono protetti e non privati. Inoltre è collegata alla classe astratta RilevatoreVelocità.

### *Centralina stradale*

La classe CentralinaStradale è costruita utilizzando il design pattern Factory Method per creare un DatoTraffico. La centralina stradale in un intervallo di tempo raccoglie il numero di veicoli (con RilevatoreVeicoli) e la loro velocità (con RilevatoreVelocitàS) che transitano davanti alla centralina. Al termine dell'intervallo viene creato il Dato-Traffico e inviato al GestoreCentraline tramite RMI e viene calcolato il nuovo intervallo di tempo in proporzione al traffico.

### *Centralina automobilistica*

La classe CentralinaAutomobilistica è costruita utilizzando il design pattern Factory Method per creare uno StatoVeicolo. La centralina automobilistica al termine dell'intervallo di tempo (che rimane costante) raccoglie posizione (con SensoreGPS) e velocità (con RilevatoreVelocitàA) del veicolo, crea uno StatoVeicolo e lo invia al GestoreCentraline tramite RMI.

### *Utente*

La classe Utente ha il compito di descrivere le informazioni relative ad ogni specifico utente che ha installato l'applicazione sul proprio dispositivo mobile e che si è registrato al servizio, consentendo così di effettuare il login.

### *Amministratore*

La classe Amministratore, così come la classe Utente, ha il compito di descrivere le informazioni relative ad ogni specifico amministratore, così da consentire ad essi il login e visualizzare quindi lo stato del sistema in forma di mappa e di diagramma.

### *SensoreGPS (astratta)*

La classe astratta SensoreGPS rappresenta il sensore generico necessario alla rilevazione della posizione. Essa si specializza in SensoreGPTeléfono e SensoreGPSCentralina, differenziando così il rilevatore della posizione del telefono, e quindi quello associato all'applicazione mobile, da quello che caratterizza la centralina automobilistica. Ognuna di queste due classi, nel momento in cui viene chiamato il metodo rilevaPosizione(), instanziano un nuovo oggetto di tipo Posizione salvando in esso le coordinate appena rilevate.

### *Posizione*

La classe Posizione esprime la posizione GPS in termini di coordinate (latitudine e longitudine). In particolare, è necessaria per descrivere la posizione in cui si trovano l'applicazione, la centralina stradale o la centralina automobilistica, così da poter essere usata nell'eventualità in cui debba essere creata la rispettiva notifica di un evento di traffico.

### *RilevatoreVelocità (astratta)*

La classe astratta RilevatoreVelocità raccoglie gli attributi e i metodi in comune tra RilevatoreVelocitàA e RilevatoreVelocitàS, quindi presenta gli attributi protetti in modo tale da poter essere acceduti dalle sottoclassi.

### *RilevatoreVelocitàA*

La classe RilevatoreVelocitàA rappresenta il sensore necessario alla centralina automobilistica per rilevare la velocità corrente del veicolo su cui è installata.

### *RilevatoreVelocitàS*

La classe RilevatoreVelocitàS rappresenta il sensore necessario alla centralina stradale per rilevare la velocità del veicolo che transita in fronte ad essa.

### *RilevatoreVeicoli*

La classe RilevatoreVeicoli conta il numero di veicoli che transitano davanti alla centralina stradale e li salva in una variabile che viene resettata ogni volta che inizia un nuovo intervallo di tempo.

### *Notifica (astratta)*

La classe astratta Notifica rappresenta il formato generico dei dati scambiati all'interno del sistema complessivo di gestione del traffico. In particolare, essa raccoglie gli attributi e i metodi comuni alle sue sottoclassi, le quali rappresentano in modo più dettagliato i tre tipi di dati utilizzati.

### *NotificaApplicazioni*

La classe NotificaApplicazioni rappresenta il dato che viene scambiato tra l'applicazione mobile e il sistema centrale. In particolare, essa definisce sia il formato dell'informazione passata da ApplicazioneMobile a GestoreApplicazioni (il quale poi la passa a GestoreDatabase) sia il formato del dato che viene passato nel verso contrario.

### *DatoTrafico*

La classe DatoTrafico rappresenta il dato che viene scambiato tra la centralina stradale e il sistema centrale. In particolare, essa definisce il formato dell'informazione passata da CentralinaStradale a GestoreCentraline (il quale poi la passa a GestoreDatabase).

### *StatoVeicolo*

La classe StatoVeicolo rappresenta il dato che viene scambiato tra la centralina automobilistica e il sistema centrale. In particolare, essa definisce il formato dell'informazione passata da CentralinaAutomobilistica a GestoreCentraline (il quale poi la passa a GestoreDatabase).

### *DatoGenerico*

La classe DatoGenerico serve per convertire i diversi dati provenienti dalle applicazioni e dalle centraline in un formato uniforme. In questo modo il database contiene dati dello stesso tipo e non di tre tipi differenti.

### *Diagramma e mappa*

Le classi Diagramma e Mappa vengono solo accennate nel class diagram in quanto esse sono parte dell’interfaccia grafica.

### *RMI*

Le restanti classi e interfacce sono quelle necessarie a consentire la comunicazione remota tramite RMI. In particolare, vi sono tre tipi di comunicazione: quella da ApplicazioneMobile a GestoreApplicazioni, quella da GestoreApplicazioni a ApplicazioneMobile e quella da Centralina a GestoreCentraline. Per ognuna di esse è stata creata un’interfaccia contenente i metodi che la classe che la implementa offre per consentire quel tipo specifico di comunicazione. Tale interfaccia estende l’interfaccia Remote resa disponibile dalla piattaforma Java.

### 1.2.2 Object diagram

L'object diagram è una fotografia, in un dato momento, degli oggetti che compongono un sistema. Mostra delle istanze piuttosto che delle classi, e per questo spesso è chiamato diagramma delle istanze.

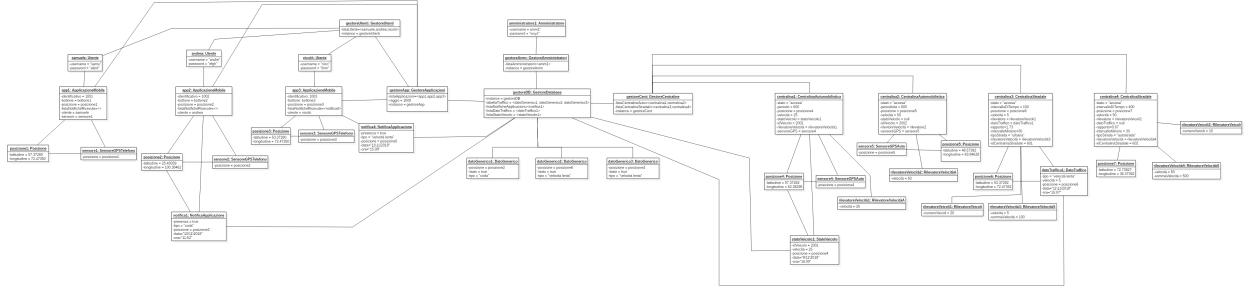


Figura 5: Object diagram

In questo object diagram, in particolare, viene analizzata la situazione in cui sono presenti, oltre ai gestori che costituiscono il sistema centrale, tre applicazioni mobili attive, associate a tre utenti diversi, due centraline automobilistiche differenti e due centraline stradali differenti. Con questa struttura si cerca di mostrare nel modo più chiaro e completo possibile come possa presentarsi il sistema complessivo nel momento in cui è in funzione.

Andando più nel dettaglio, si può notare che due delle centraline, una automobilistica e una stradale, in seguito alla scadenza del loro intervallo di tempo, hanno creato rispettivamente una notifica di tipo StatoVeicolo e una di tipo DatoTraffico. Si può notare che tali informazioni sono state già salvate nella TabellaTraffico sotto forma di dati del tipo DatoGenerico ed elaborate dal GestoreDatabase, il quale, dopo aver trovato un evento di traffico e dopo aver avvisato di ciò il GestoreApplicazioni, ha innescato il procedimento di segnalazione alle applicazioni presenti nella zona critica. È stata quindi creata una notifica del tipo NotificaApplicazioni e associata all'applicazione mobile posta nelle immediate vicinanze dell'evento (si intende che essa si trova all'interno dell'area circolare di raggio preimpostato).

Al fine di mostrare tutto ciò che può accadere con il sistema complessivo in funzione, viene rappresentato anche il caso in cui sia un'applicazione a segnalare la presenza di coda nel punto in cui si trova. Essa crea quindi una notifica di tipo NotificaApplicazioni che, successivamente, tramite il GestoreApplicazioni viene inviata al GestoreDatabase, il quale la salva e la elabora.

## 1.3 STRUCTURAL MODELING – IMPLEMENTATION DIAGRAMS

### 1.3.1 Deployment diagram

Il deployment diagram documenta la distribuzione fisica di un sistema. Gli elementi principali si chiamano nodi e sono collegati da path di comunicazione.

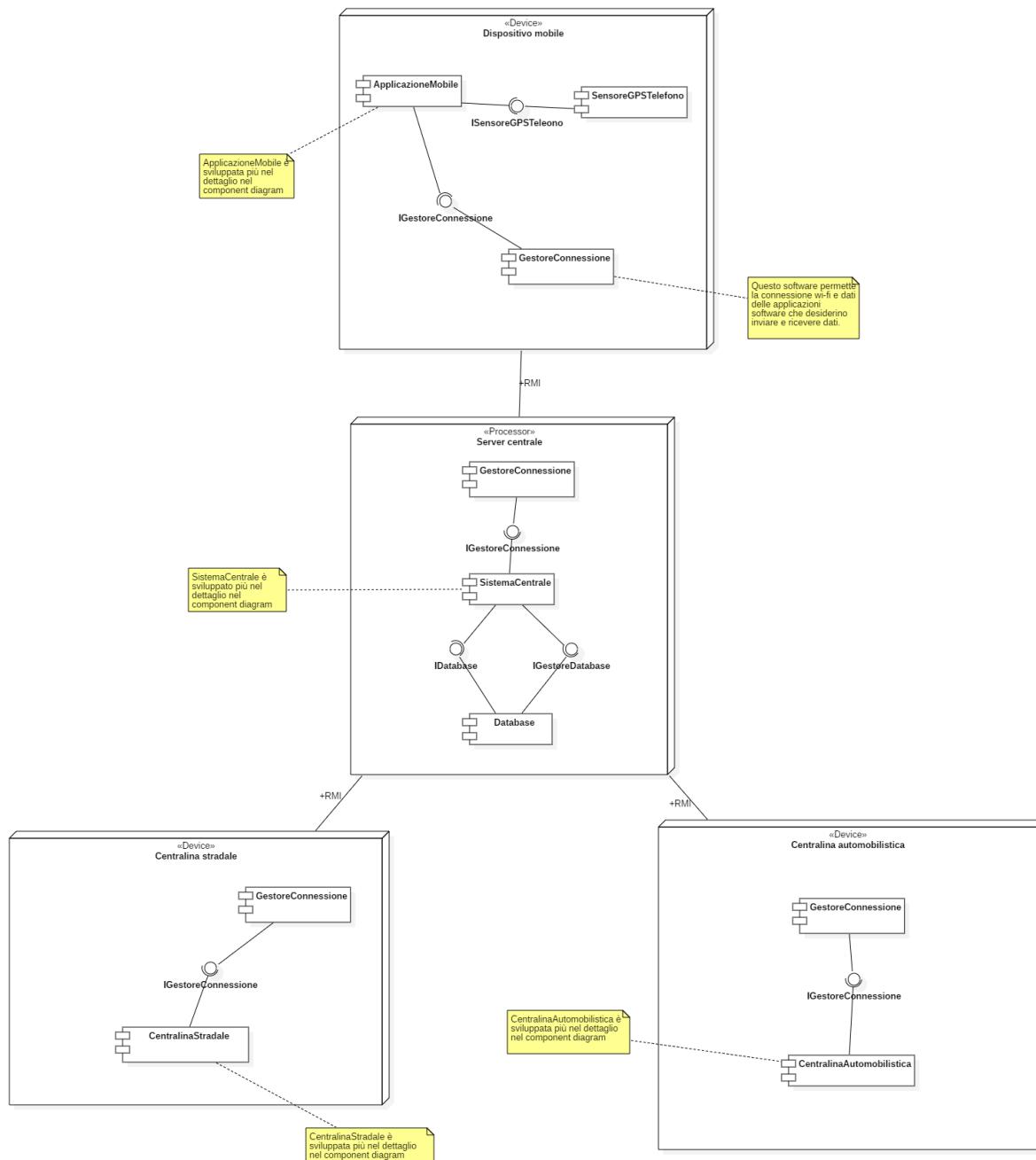


Figura 6: Deployment diagram

In questo deployment diagram vengono mostrati i componenti fisici che compongono il sistema complessivo e come comunicano tra loro. L'utente e l'amministratore, di conseguenza, non vengono rappresentati in quanto non si trattano di componenti fisici. La comunicazione tra il sistema centrale, le centraline e i dispositivi mobili avviene attraverso una connessione remota tramite RMI Java.

Il nodo server centrale rappresenta il centro di elaborazione di tutti i dati che vengono raccolti dai vari sottosistemi e, allo stesso tempo colui che manda informazioni alle applicazioni mobili. Esso è quindi costituito da un componente adibito alla gestione della connessione tramite RMI, un componente DataBase che costituisce il centro di memorizzazione dei dati e un componente SistemaCentrale (spiegato più nel dettaglio con il component diagram) che gestisce ed elabora le informazioni ricevute.

Il nodo Dispositivo mobile rappresenta appunto il dispositivo su cui viene installata l'applicazione mobile, rappresentata dall'omonimo componente (spiegato più nel dettaglio con il component diagram). Vi è poi anche qui un componente che si occupa della gestione della connessione tramite RMI, e infine un componente che gestisce il sensore GPS necessario per la rilevazione della posizione.

I due nodi rimanenti rappresentano invece i due differenti tipi di centralina (stradale e automobilistica) e contengono, rispettando la struttura degli altri nodi, un componente che gestisce le connessioni RMI e un componente rappresentante la centralina dal punto di vista software.

### 1.3.2 Component diagram

Il component diagram rappresenta le componenti software che caratterizzano il sistema complessivo e le interfacce utilizzate dai sottosistemi per comunicare tra loro.

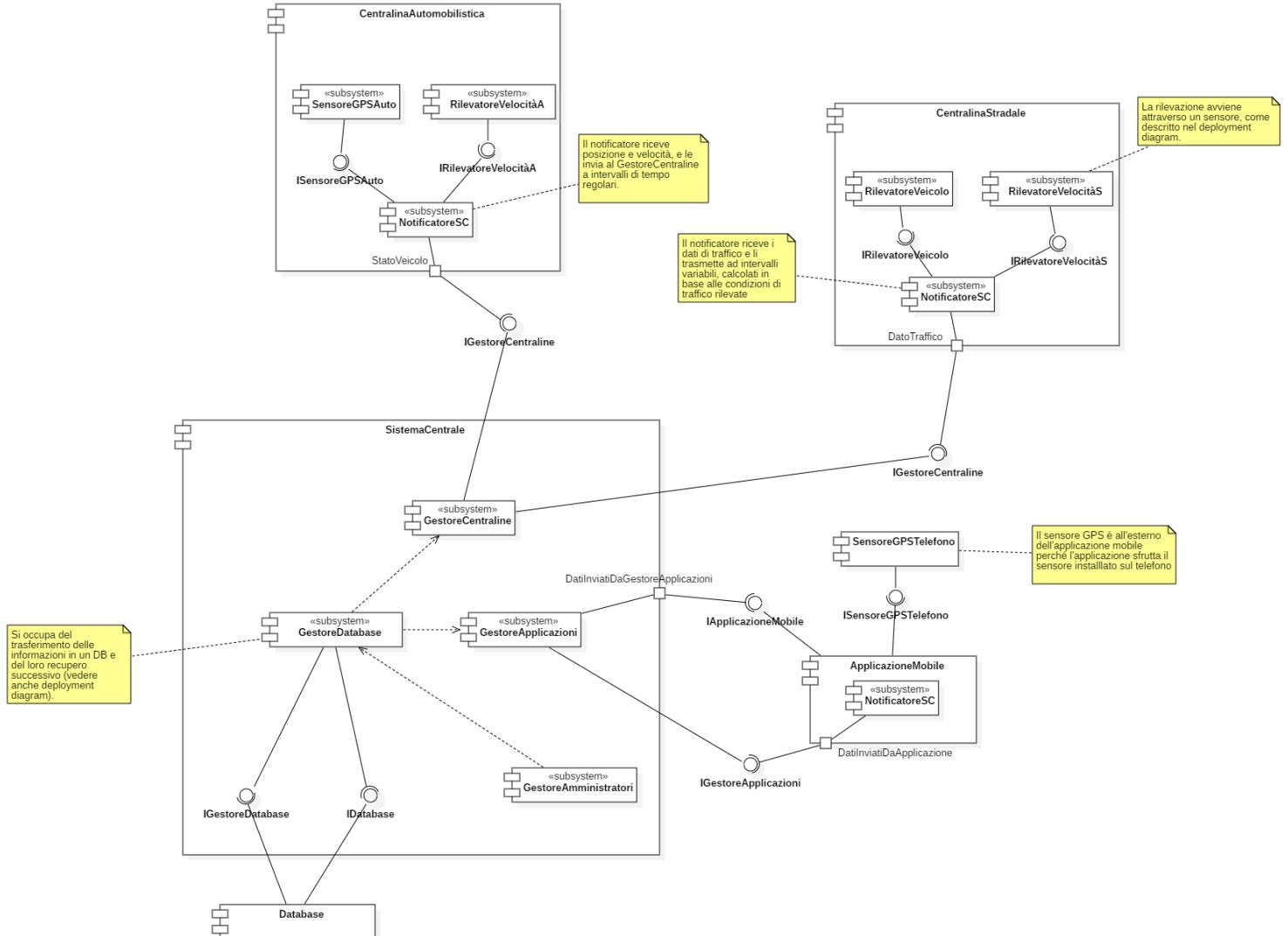


Figura 7: Component diagram

Il componente SistemaCentrale si occupa della gestione dei sottosistemi e della gestione delle informazioni raccolte. Per fare ciò, esso è suddiviso in componenti minori, ciascuno che si occupa di una specifica funzione. In particolare, il GestoreApplicazioni e il GestoreCentraline si occupano rispettivamente di tutto ciò che riguarda le applicazioni mobili e le centraline. Vi è poi un componente, GestoreDatabase, che si occupa della gestione delle informazioni ricevute, elaborandole e interfacciandosi con il DataBase in modo da salvarle e recuperarle nel momento in cui sono necessarie.

Il componente ApplicazioneMobile contiene un componente NotificatoreSC che si occupa di gestire l'invio della notifica di presenza di coda dall'applicazione mobile al sistema centrale. Si interfaccia poi con il componente SensoreGPS per rilevare la propria posizione e poterla fornire quindi al sistema centrale, il quale, in base ad essa, sa quali notifiche deve inviarle e quali no.

Le centraline sono anche esse costituite da un componente NotificatoreSc per la comunicazione con il sistema centrale e dai componenti necessari alla rilevazione dei dati per i quali le centraline sono state installate. In particolare, la centralina stradale possiede un componente per la rilevazione del passaggio di un veicolo e uno per la rilevazione della velocità di tale veicolo, mentre la centralina automobilistica contiene un sensore per la rilevazione della posizione e uno per la misurazione della velocità del veicolo su cui è installata.

## 1.4 DYNAMIC MODELING – INTERACTION DIAGRAMS

### 1.4.1 Sequence diagrams

Il sequence diagram documenta tipicamente il comportamento di un singolo scenario. Il diagramma include un certo numero di oggetti e i messaggi scambiati tra essi durante l'esecuzione in ordine temporale dello scenario.

#### *Centralina stradale*

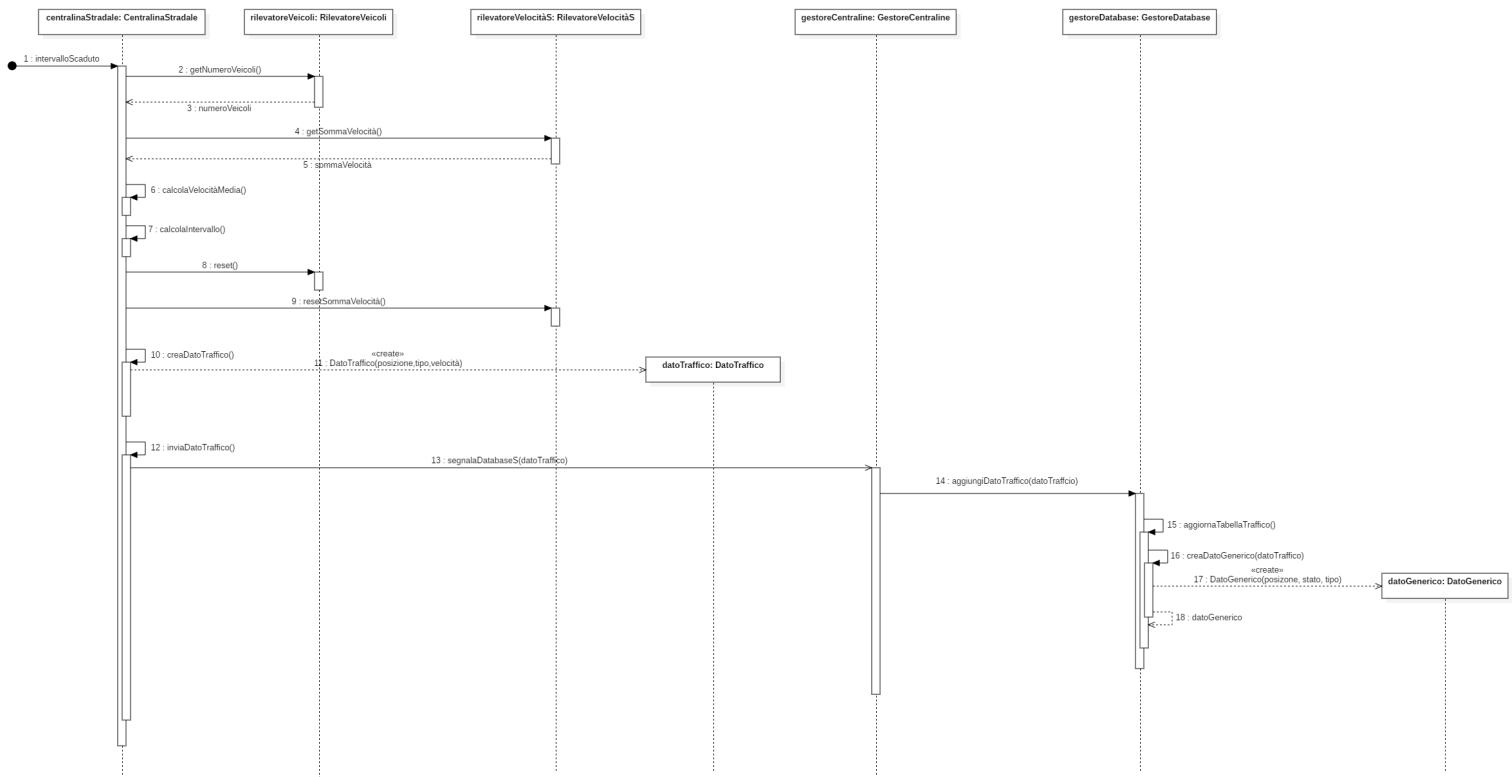


Figura 8: Sequence diagram CentralinaStradale

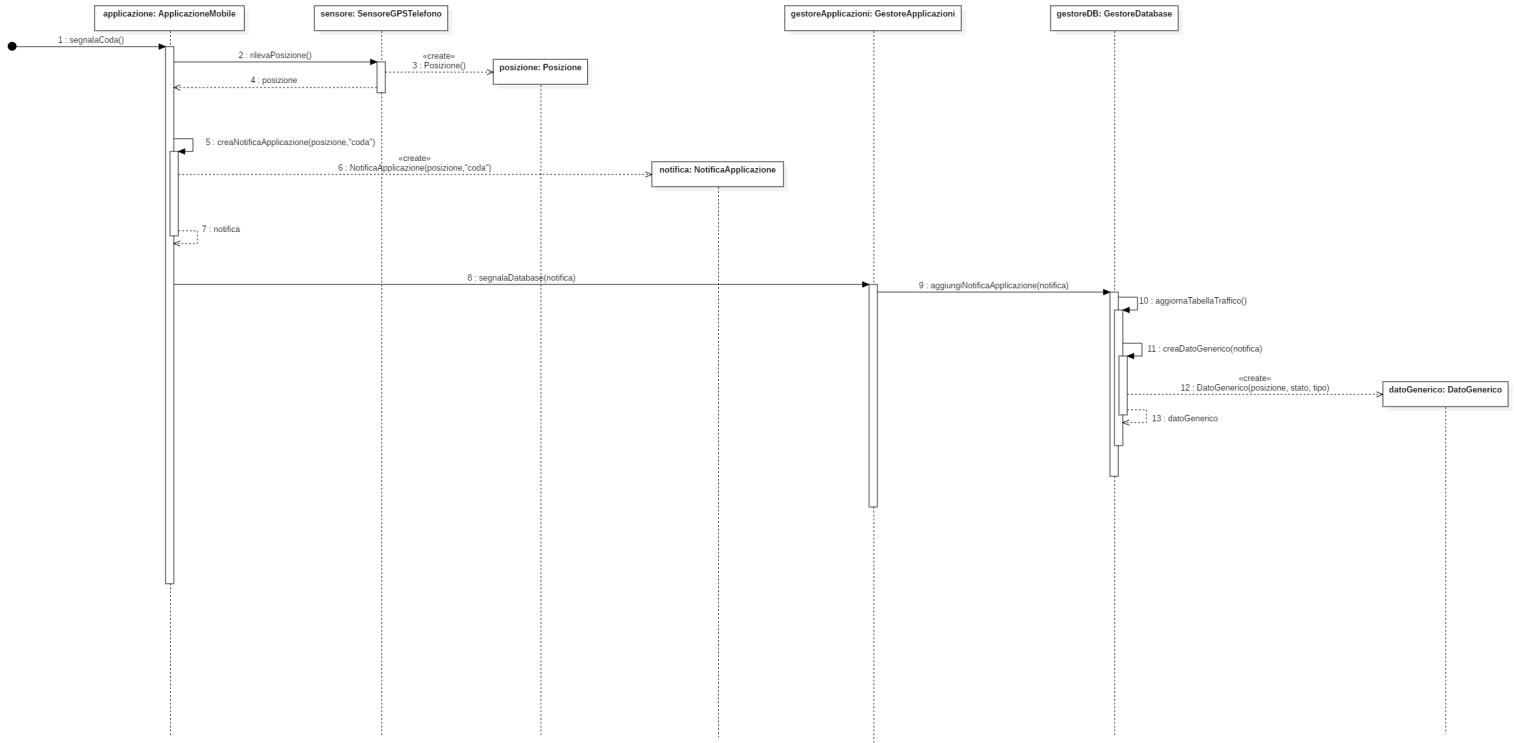
In questo sequence diagram è rappresentato ciò che comporta lo scadere dell'intervallo di tempo di una centralina stradale.

Terminato l'intervallo di tempo, la centralina stradale richiede il numero di veicoli che sono stati rilevati dall'apposito sensore rappresentato dalla classe RilevatoreVeicoli e la somma delle loro velocità al RilevatoreVelocitàS. In base alla quantità di vetture transitate davanti alla centralina, viene calcolato il nuovo intervallo di tempo e la velocità media.

In seguito, la centralina stradale crea un dato di traffico configurando i suoi parametri. La notifica sulla situazione del traffico viene inviata al GestoreCentraline, il quale dopo averla elaborata la inoltra

al GestoreDatabase. La sequenza di azioni termina con la creazione di un DatoGenerico che viene inserito nella TabellaTraffico.

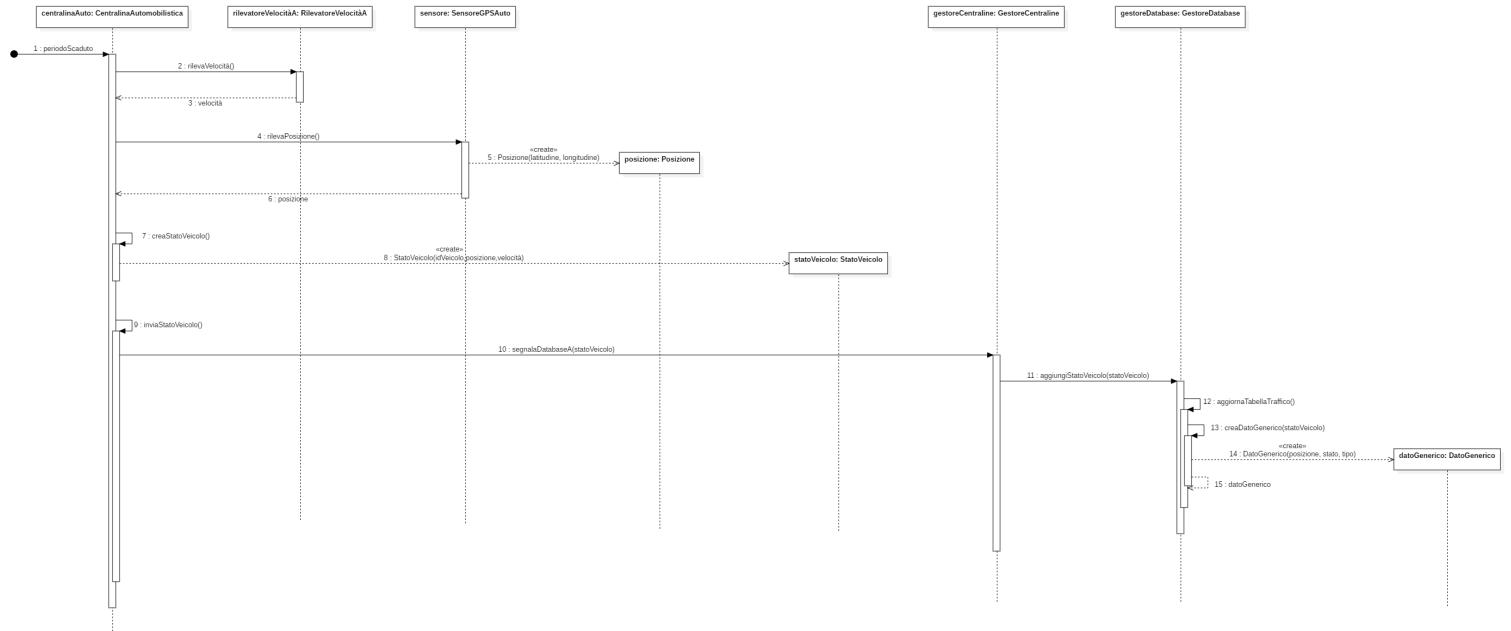
### *Applicazione mobile segnala la coda*



**Figura 9:** Sequence diagram ApplicazioneMobile segnala un evento di coda

In questo sequence diagram è rappresentato il meccanismo che gestisce la segnalazione della presenza di coda da parte di un utente. Nel momento in cui viene premuto il bottone dell’interfaccia grafica offerta dall’applicazione, viene subito avviato il procedimento di segnalazione: l’applicazione richiede la posizione attuale al SensoreGPS a lei associato, successivamente crea una notifica del tipo NotificaApplicazione e la invia immediatamente al GestoreApplicazioni. Quest’ultimo, non appena ricevuto il dato, lo inoltra al GestoreDatabase il quale crea un DatoGenerico, lo salva e aggiorna la TabellaTraffico contenente tutti i dati fino ad ora ricevuti.

### Centralina automobilistica



**Figura 10:** Sequence diagram CentralinaAutomobilistica

In questo sequence diagram viene mostrato cosa avviene quando scade il periodo di tempo della centralina stradale.

La centralina stradale al termine del periodo rileva la velocità del veicolo (grazie al RilevatoreVelocitàA) e la posizione corrente (con il SensoreGPSAuto). Successivamente crea uno StatoVeicolo e lo invia al GestoreCentraline, il quale lo inoltra al GestoreDatabase. Quest'ultimo aggiorna la TabellaTraffic creando e inserendo un DatoGenerico.

### 1.4.2 Collaboration diagram

Il collaboration o communication diagram mostra le comunicazioni che intercorrono tra diversi oggetti in una situazione particolare, enfatizzando lo scambio dei dati tra gli oggetti. La sequenza temporale è rappresentata enumerando i messaggi scambiati durante l'esecuzione dello scenario.

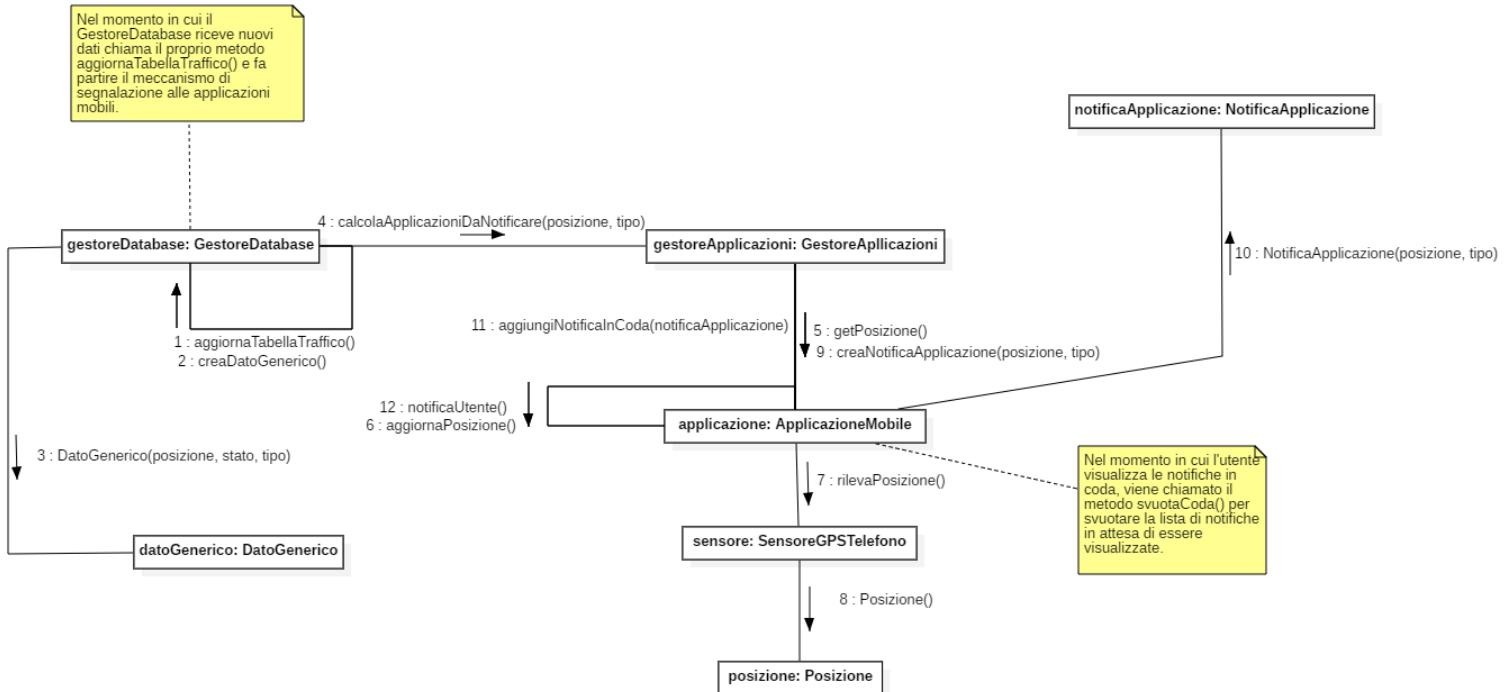


Figura 11: Collaboration diagram

In questo collaboration diagram, si analizzano gli oggetti e i metodi coinvolti quando il GestoreDatabase vuole notificare le applicazioni riguardo a un nuovo evento di traffico appena ricevuto. Questo evento di traffico può essere uno StatoVeicolo, un DatoTraffico o una NotificaApplicazione.

In questa situazione, il GestoreDatabase aggiorna la tabella di traffico creando un DatoGenerico e chiama il metodo di GestoreApplicazioni per calcolare quali sono le applicazioni da notificare in base alla posizione dell'evento di traffico ricevuto.

Il GestoreApplicazioni chiede a ogni applicazione la posizione (che viene calcolata grazie al sensoreGPS) e valuta se è all'interno del cerchio coinvolto nell'evento di traffico. Nel caso lo fosse, viene creata una NotificaApplicazione, aggiunta in coda all'applicazione e infine segnalato l'utente.

## 1.5 DYNAMIC MODELING – STATECHART DIAGRAMS

Lo statechart diagram descrive il comportamento di un oggetto per la durata del suo ciclo di vita, mostrando gli stati che assume, gli eventi a cui risponde, le risposte che fornisce e le transizioni tra gli stati.

*Centralina stradale*

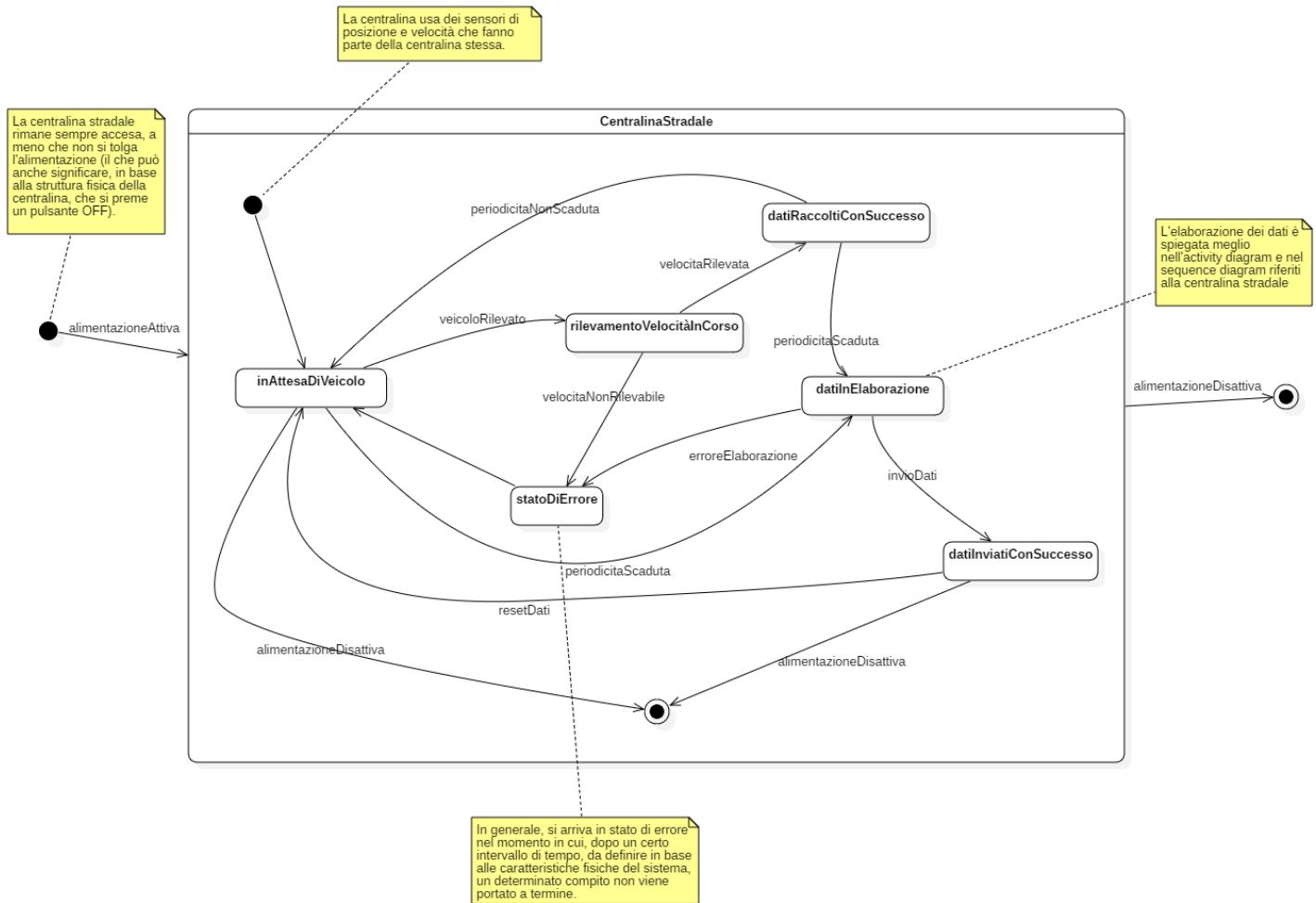


Figura 12: State diagram CentralinaStradale

La centralina stradale si occupa del rilevamento dei dati di traffico, che devono essere inviati al sistema centrale ad intervalli di tempo (periodicità) variabili a seconda delle condizioni del traffico rilevate.

La centralina può essere accesa o spenta. In particolare, potrà essere spenta scollegando l'alimentazione oppure utilizzando un tasto apposito. Questi dettagli, però, vengono definiti sulla base della struttura fisica della centralina e, di conseguenza, non sono parte del progetto.

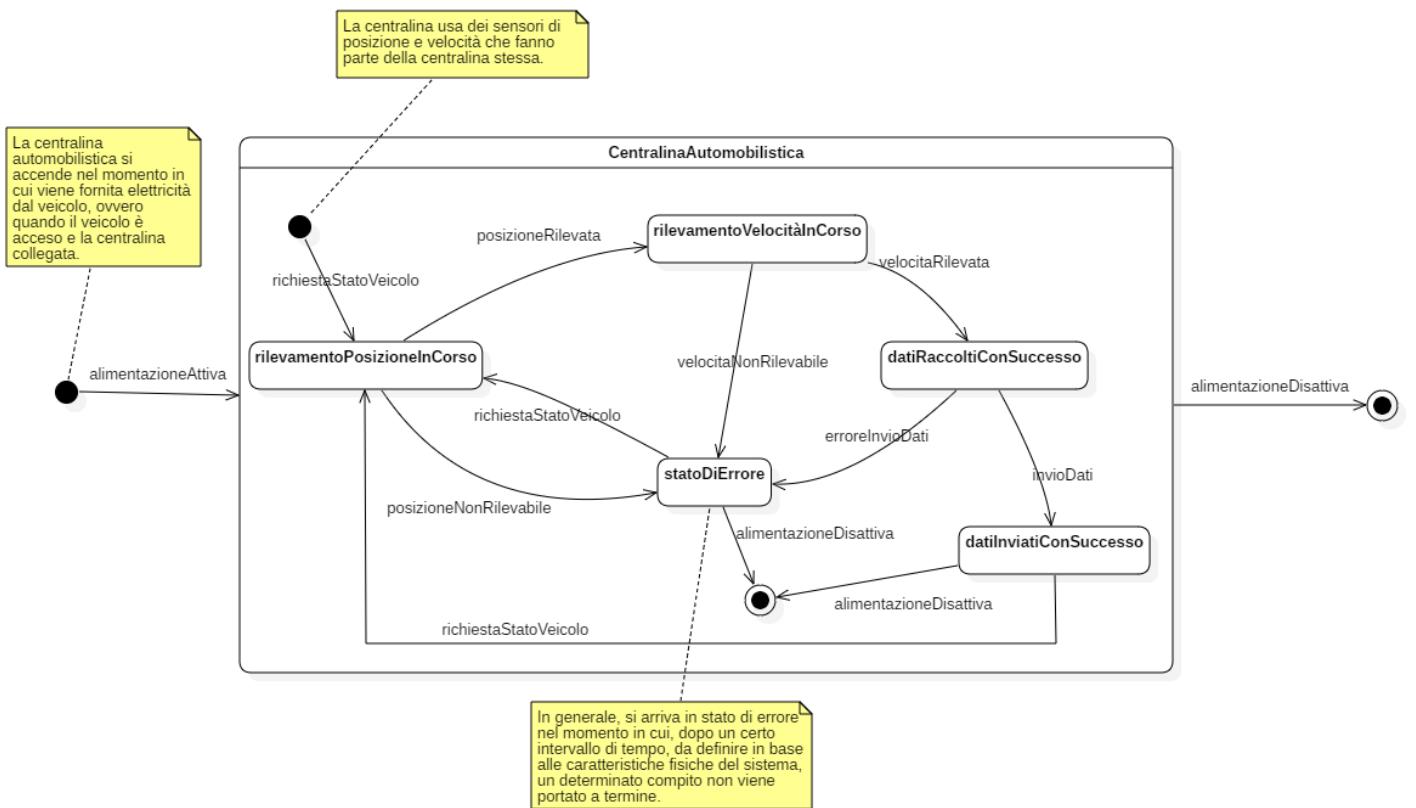
Il sistema centrale richiede i dati di traffico alla centralina, la quale, in seguito alla richiesta, attende che transiti un veicolo da rilevare, continuando ad aggiornare un clock interno per la misurazione del tempo trascorso.

Nel momento in cui un veicolo viene rilevato, ne viene rilevata anche la velocità e viene valutato il tempo rimanente prima che i dati debbano essere inviati al sistema centrale. Se il tempo non è ancora scaduto, si attende un nuovo veicolo e si ripetono i passaggi già descritti. In caso contrario i dati raccolti vengono elaborati e inviati al sistema centrale, resettando il conteggio del tempo.

Se, invece, la periodicità scade mentre la centralina sta attendendo il veicolo, invia i dati raccolti fino a quel momento.

Nel caso in cui una qualsiasi operazione non venga portata a termine, la centralina torna in attesa di un veicolo da rilevare.

### *Centralina automobilistica*



**Figura 13:** State diagram CentralinaAutomobilistica

La centralina automobilistica si occupa del rilevamento dello stato del veicolo, che deve essere inviato al sistema centrale ad intervalli

di tempo che, a differenza di quanto accade nel caso della centralina stradale, sono regolari.

La centralina può essere accesa o spenta. In particolare, potrà essere spenta scollegando l'alimentazione (questa situazione si verifica, per esempio, quando viene spenta l'automobile).

Il sistema centrale richiede lo stato del veicolo alla centralina, la quale, in seguito alla richiesta, rileva la posizione del veicolo in quel momento e la sua velocità. Successivamente i dati raccolti vengono inviati al sistema centrale e si inizia nuovamente il processo.

Nel caso in cui un'operazione non venga portata a termine correttamente (ed entro un certo intervallo di tempo), la centralina inizia nuovamente a rilevare posizione e velocità del veicolo. Il motivo per cui non vengono recuperati i dati da inviare è che questi diventano presto obsoleti e, di conseguenza, poco utili.

### *Applicazione mobile*

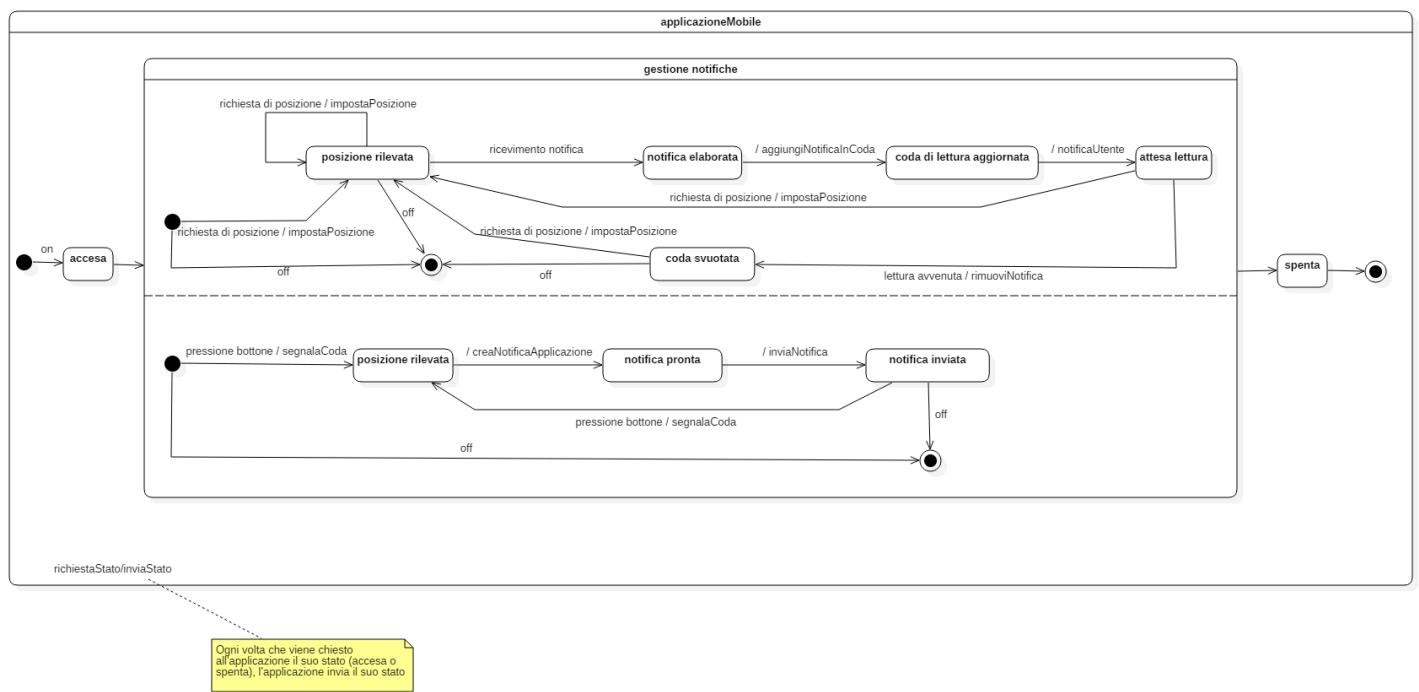


Figura 14: State diagram ApplicazioneMobile

L'applicazione mobile si può trovare in tre stati: accesa, spenta, oppure gestione delle notifiche. In ognuno di questi stati, se dovesse arrivare un evento di richiesta dello stato da parte del gestore delle applicazioni, l'applicazione invia true se è accesa o in gestione notifiche, false se spenta.

Per entrare nello stato "accesa" è necessario un evento "on", in "spenta" è necessario ricevere "off", mentre nello stato "gestione notifiche" si accede immediatamente dopo l'accensione.

L'applicazione può ricevere notifiche dal GestoreApplicazioni oppure può inviare a sua volta notifiche al GestoreApplicazioni. Per questo motivo la gestione delle notifiche è rappresentato come un sottosistema che esegue entrambi i compiti in parallelo.

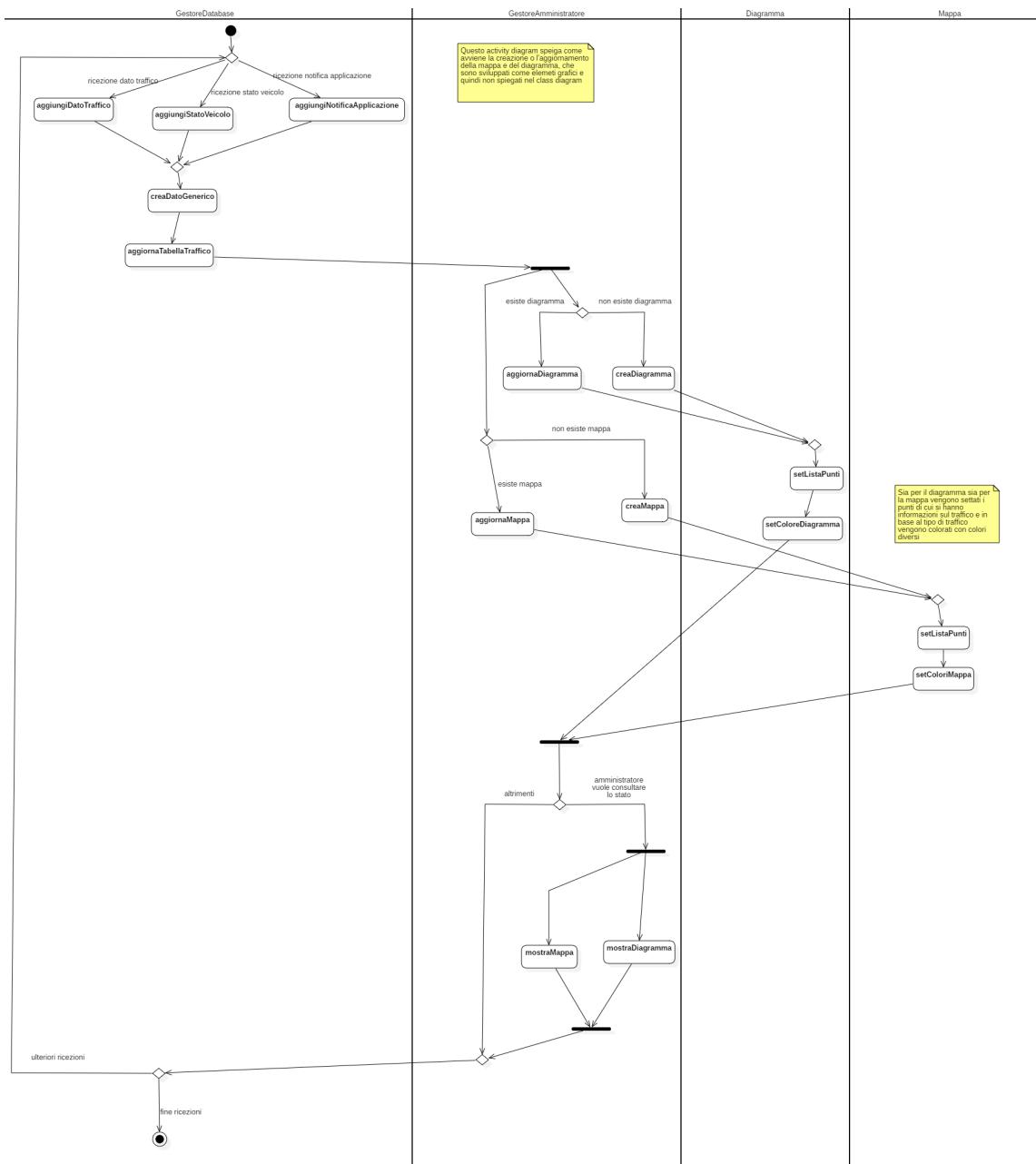
Nella parte superiore viene rappresentata la ricezione di una possibile nuova notifica di traffico da mostrare all'utente, che inizia con una richiesta di impostazione della posizione, se l'applicazione è all'interno del raggio di notifica prosegue con la ricezione della notifica e termina con la gestione della coda di notifiche da leggere, altrimenti rimane in rilevamento posizione aspettando una nuova richiesta.

Nella parte inferiore invece è mostrato come si comporta l'applicazione quando l'utente preme il bottone, cioè dopo la pressione viene rilevata la posizione, scritta la notifica e inviata.

## 1.6 DYNAMIC MODELING – ACTIVITY DIAGRAMS

Un activity diagram illustra i flussi tra le attività associate a un determinato scenario mostrando gli oggetti, le transizioni, i branch, i merge, le fork e le join.

*Visualizzazione stato traffico da parte dell'amministratore*



**Figura 15:** Activity diagram aggiornamento tools amministratore

La situazione che si vuole rappresentare in questo activity diagram è l'aggiornamento dei tools a disposizione dell'amministratore per la consultazione del traffico.

Il GestoreDatabase riceve uno statoVeicolo o un DatoTraffico o una Notifica applicazione, lo aggiunge alla lista corrispondente, lo converte in un DatoGenerico e aggiorna la TabellaTraffico. A questo punto si passa al GestoreAmministratori.

Il gestore degli amministratori esegue due azioni in parallelo e cioè crea/aggiorna il diagramma e crea/aggiorna la mappa settando le loro caratteristiche rispettivamente.

Terminate queste operazioni, il gestore amministratori rende consultabili il diagramma e la mappa agli amministratori e li mostra a coloro i quali vogliono visualizzare lo stato del traffico.

Infine si ritorna al GestoreDatabase che passa alla ricezione successiva se è presente, altrimenti termina l'esecuzione.

### Monitoraggio del traffico stradale e notificare le applicazioni

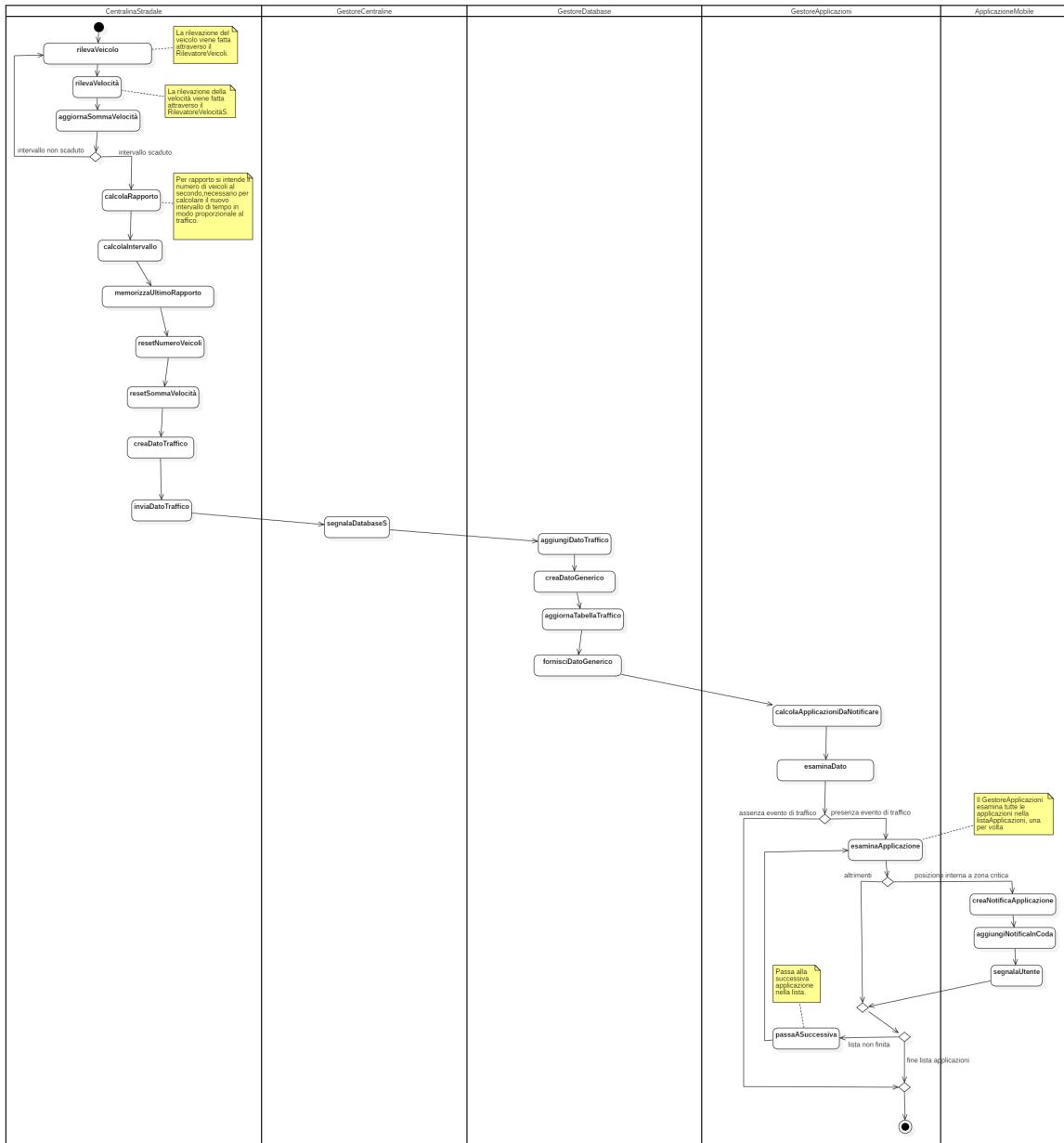


Figura 16: Activity diagram centralina stradale e applicazioni

In questo activity diagram si vuole descrivere il procedimento con cui la centralina stradale monitora il traffico nel segmento di strada in cui è installata. In particolare, attraverso i sensori di rilevazione dei veicoli e di rilevazione della velocità essa tiene traccia del numero di veicoli che le passano davanti, rilevandone per ognuno la velocità.

Nel momento in cui scade l'intervallo di tempo, viene calcolato il nuovo intervallo, resettati i conteggi e creato un DatoTraffico. Il Dato-Traffico creato viene inviato al GestoreCentraline, il quale, a sua volta

lo manda al GestoreDatabase che memorizza il dato convertendolo in DatoGenerico.

Successivamente il controllo passa al GestoreApplicazioni che calcola le applicazioni da notificare, esaminando prima il dato ricevuto e poi chiedendo a ogni applicazione la posizione. Per ogni applicazione da notificare viene creata una NotificaApplicazione e viene segnalato l'utente.

## BIBLIOGRAFIA

- [Fow] Martin Fowler. *UML Distilled*. Pearson. ISBN: 9788871925981.
- [Sco] Kendal Scott. *UML explained*. Addison-Wesley. ISBN: 9788871921204.