# PCD - Report Assignment 3

*Esercizio 2*

Andrea Giulianelli
andrea.giulianelli4@studio.unibo.it

Il problema richiede di creare un'applicazione distribuita che ha lo scopo di monitorare possibili allagamenti in una smart-city adottando un approccio peer-to-peer e basato sul paradigma ad attori.
Per fare ciò sono stati necessari diversi passi che sono descritti, nei loro punti principali, in questo report.

## 1-Analisi del problema

In questa sezione verranno ripresi e rielaborati i concetti presenti nella descrizione del problema al fine di analizzarli e comprenderli nel dettaglio per poter proseguire con il design e l'implementazione del sistema.
Sono presenti 4 concetti fondamentali:

- ***Città***: rappresenta la città soggetta al monitoraggio riguardo i possibili allagamenti. Essa viene suddivisa in zone in cui sono installati diversi pluviometri. La città viene considerata a livello logico come un rettangolo WxH collocato in uno spazio euclideo
- ***Zona***: rappresenta una porzione di città che viene monitorata come un'unità. Ad ogni zona viene associata una caserma dei pompieri responsabile del monitoraggio e della salvaguardia della zona.
- ***Pluviometro***: è il sensore adibito alla misurazione del livello dell'acqua in una determinata posizione all'interno di una zona. Ogni pluviometro è configurato rispetto al luogo in cui viene installato relativamente alla posizione (quindi registrandosi ad una particolare zona), alla soglia limite che l'acqua può raggiungere in quel determinato punto e al periodo di campionamento del livello dell'acqua.
- ***Caserma dei pompieri***: si occupa del monitoraggio e della gestione di ogni eventuale allarme dovuto ad un possibile allagamento di una zona della città.
  La caserma si può trovare in due stati: *libera* e *occupata* a seconda che stia o meno gestendo l'allarme della zona associata.

In particolare abbiamo che la città viene suddivisa, nel caso specifico di questo assignment, in un numero statico di zone, le quali ricoprono completamente la superficie della città senza sovrapposizioni. Ogni zona ha una caserma di pompieri che si occupa della gestione dell'allarme della zona stessa. In particolare, la relazione è biunivoca in quanto una caserma, a sua volta, è associata ad una sola zona.
La zona si può trovare in tre stati:

- *Normale*: è lo stato in cui il livello dell'acqua rientra nei limiti imposti nei vari punti di misurazione.
- *Allarme*: la zona si trova in uno stato di allarme quando la maggioranza dei sensori appartenenti alla zona stessa percepisce un valore sopra la soglia limite.

- *In gestione*: la zona, precedentemente in uno stato di allarme, passa in uno stato di "*gestione*" non appena la caserma dei pompieri prende in gestione l'allarme della zona stessa.

Ogni pluviometro installato viene, come anticipato, configurato per operare (quindi comunicare i propri risultati) in una sola zona. Inoltre, essi possono fallire in un qualsiasi istante, perciò, al fine di effettuare calcoli relativi alla maggioranza, occorre essere consapevoli del loro stato in ogni fase del monitoraggio.
Come descritto, non appena la maggioranza dei pluviometri (appartenenti alla stessa zona) registra un livello sopra la propria soglia limite, la zona va in uno stato di allarme e il sistema deve allertare immediatamente la caserma dei pompieri associata.
L'allarme diramato dal sistema può essere disabilitato solamente dalla caserma autorizzata, perciò questo significa che anche se i vari pluviometri presenti nella zona verificano un ritorno ad una condizione normale lo stato della zona rimane comunque in *allarme*.
Infine, ogni caserma dei pompieri è dotata di una dashboard con la quale monitorare e gestire lo stato della propria zona e monitorare lo stato delle altre caserme/zone.

Le relazioni tra i vari concetti sono le seguenti:

- *Città* <-> *Zona*: la città è consapevole della sua suddivisione in zone ed ogni zona è consapevole della sua responsabilità.
- *Zona* <-> *Caserma*: ogni zona ha una caserma associata a cui comunicare il proprio stato. La caserma a sua volta è associata ad una e una sola zona.
- *Zona* <-> *Pluviometri*: ogni zona tiene costantemente aggiornata la lista di pluviometri che ha a disposizione. Ogni pluviometro viene configurato con la zona di appartenenza.
- *Caserma* <-> *Caserma*: le caserme sono in relazione tra loro in quanto necessitano di comunicare al fine di scambiarsi informazioni riguardo lo stato delle rispettive zone monitorate.

# 2-Design

In questa sezione verranno descritte le principali scelte di design che hanno caratterizzato la soluzione al problema definito nella sezione precedente.
Innanzitutto verranno messe in risalto, riprendendo i concetti descritti in analisi, le scelte relative alle responsabilità e alla distribuzione dei vari elementi di dominio. Dopodiché, considerando le scelte, verranno descritti i vari nodi presenti nell'architettura e le interazioni possibili. Infine, si fornirà una descrizione più dettagliata del design di ciascun nodo.

## Descrizione della strategia risolutiva

Sono state eseguite le seguenti scelte relativamente ai concetti descritti in analisi:

- ***Zona***: le zone della città sono statiche, cioè sono parte della configurazione del problema. Esse simulano la presenza di una centralina che gestisce ogni zona della città.
  Ogni zona è identificata da un ID logico che viene utilizzato per la configurazione dei pluviometri e delle caserme. Grazie a ciò, saranno i pluviometri e le caserme stesse che avvieranno il processo di registrazione verso la zona associata e non viceversa. Infatti, essi non appena avviati invieranno una richiesta di registrazione alla zona associata. La zona, ricevuta la richiesta, potrà decidere se accettarli o meno. In questa prima implementazione non vi è alcun limite per l'accettazione dei pluviometri,

però il pattern di interazione request-response permette una maggior flessibilità qualora si vogliano introdurre delle politiche più stringenti di registrazione.
La strategia scelta per gestire l'allarme è la seguente:

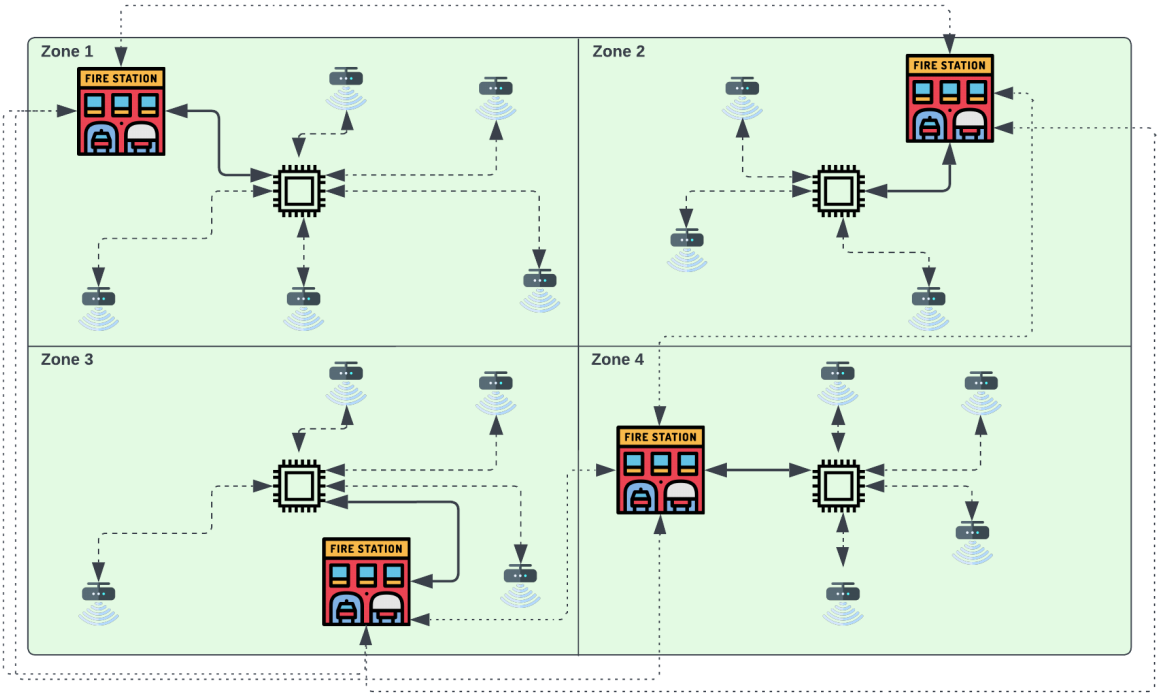
- Il primo pluviometro che segnala un possibile allarme fa passare la zona in uno stato di *pre-allarme* durante il quale richiede una lettura hot, quindi immediata, del livello dell'acqua corrente a tutti i pluviometri presenti nella zona
- Registra tutti i dati in ingresso e calcola la maggioranza
- In caso in cui la maggioranza sia in allarme, segnala l'allarme alla caserma registrata (se presente). La zona rimarrà in uno stato di allarme fino a che la caserma non lo gestisce.
- In caso di falso allarme, ritorna alla normale operatività.

Questo permette anche la gestione di eventuali spike prodotti dai pluviometri.
Inoltre, la zona deve sempre essere reattiva a nuove registrazioni e a possibili failure dei pluviometri.
*Assunzione*: le zone non falliscono mai.

- ***Pluviometro***: ogni pluviometro viene configurato con l'ID logico della zona di appartenenza in aggiunta alle altre configurazioni descritte in analisi (tra cui la frequenza di lettura e la soglia che rappresenta il livello massimo dell'acqua in quello specifico punto).
  Il pluviometro è stato pensato per essere completamente indipendente dallo stato generale del sistema. Come anticipato, all'avvio, dovrà provvedere alla sua registrazione presso la zona associata e dopo questa prima fase il suo unico scopo e quindi la sua unica responsabilità sarà quella di misurare periodicamente (con frequenza, come anticipato, configurabile) il livello dell'acqua e segnalare tempestivamente l'allarme alla "centralina" di zona associata. Inoltre, dovrà anche rispondere ad eventuali richieste di letture immediate provenienti dall'esterno.
  Il punto cruciale è che il pluviometro non è consapevole dello stato della zona in cui si trova. In questo modo si evita una complessità e una intelligenza non necessaria - in questo contesto - a bordo di un semplice sensore.
  *Ogni pluviometro può fallire in un qualsiasi momento.*
- ***Caserma***: ogni caserma viene configurata con l'ID logico della zona di appartenenza in aggiunta alle altre configurazioni descritte in analisi. La caserma, come descritto precedentemente, si registra alla zona associata (grazie all'ID fornito) al fine di monitorare e gestire lo stato di essa.
  Ogni caserma, come anticipato, necessita di essere a conoscenza dello stato di tutte le altre zone. A tal fine si è scelto di fare interagire tutte le caserme tra loro scambiandosi i dati in modo push-based. In particolare, ad ogni cambio di stato (della caserma stessa o della zona associata) la caserma condivide le proprie informazioni con tutte le altre caserme presenti all'interno della città. Si è preferito scegliere questa strada al posto di fare comunicare ogni caserma con tutte le zone al fine di incapsulare in miglior modo i dati di ogni zona. In questo modo ogni caserma controlla l'esposizione di dati a consumatori esterni.
  Una caserma si può trovare in due stati:
  - *Libera*: non sta eseguendo alcuna operazione
  - *Occupata*: sta gestendo un allarme della propria zona

  *Assunzione*: le caserme non falliscono mai.

La seguente immagine illustra quanto descritto e le principali interazioni attraverso un esempio in cui la città è suddivisa in quattro zone.

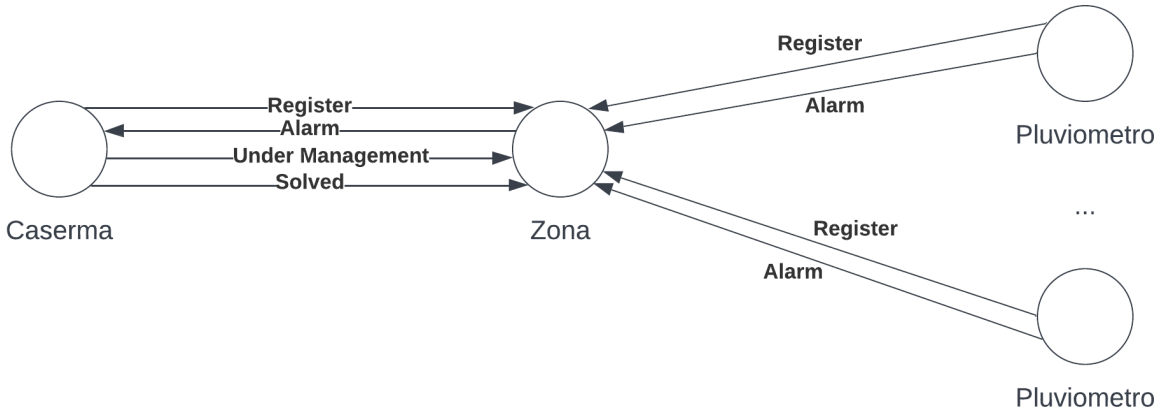



Dopo aver descritto le responsabilità e le interazioni principali è necessario individuare come i vari elementi vengono distribuiti all'interno dell'architettura proposta. La distribuzione degli elementi è diretta rispetto a quanto già detto, in particolare:

- Ogni *Pluviometro* sarà un nodo.
- Ogni "controllore" di *Zona* verrà rappresentato attraverso un nodo.
- Ogni *Caserma* sarà un nodo.

## Descrizione del comportamento del sistema

Il sistema viene avviato fornendo una configurazione la quale imposta le caratteristiche del sistema in oggetto. Dopodiché, viene effettuato il deploy dei vari nodi partendo dalle zone seguite dalle caserme e dai pluviometri in un qualsiasi ordine.
Il sistema è completamente distribuito peer-to-peer basato sul paradigma ad attori.
Qui di seguito è possibile vedere il macro-comportamento di ogni zona del sistema.




Durante la progettazione si è cercato di adottare un approccio il più possibile idiomatico seguendo il principio “everything is an actor” astraendo completamente dalla gestione dei thread ragionando sempre a livello logico e di dominio. Inoltre, si è evitata qualsiasi forma di memoria condivisa, monitor, locks o semafori; è stato adottato unicamente lo scambio di messaggi (immutabili) tra attori. In questo modo, è stato possibile incapsulare il flusso di controllo e decentralizzare le responsabilità tra le parti che interagiscono.
Di seguito verranno descritti i vari nodi astraendo da tecnologie particolari a supporto del paradigma ad attori. Inoltre verrà fornita una descrizione del comportamento degli attori mediante FSM in cui gli stati rappresentano gli stati in cui si trovano i rispettivi attori e le transazioni sono descritte da guardie rappresentate dal messaggio ricevuto e da azioni che indicano l’azione eseguita prima di entrare nello stato successivo. Nella modellazione delle FSM si è astratto inoltre dalla struttura dettagliata dei messaggi, descritta successivamente.

Nelle descrizioni seguenti verrà utilizzato il concetto di registro al fine di rappresentare in modo astratto un’entità/attore che possiede i riferimenti dei vari attori presenti all’interno del sistema.

### *Pluviometro*

La logica che gestisce il nodo del pluviometro è composta da un solo attore.
Esso viene configurato attraverso:

- *id*: utile per l’identificazione dei vari pluviometri interni alla stessa zona
- *id della zona associata*: viene configurato con l’id logico della zona di appartenenza in modo tale che poi possa rivolgersi ad essa
- *sense rate*: rappresenta il periodo di campionamento del livello dell’acqua
- *soglia di allarme*: rappresenta la soglia oltre la quale segnalare l’allarme alla zona associata.

Il comportamento di un pluviometro è il seguente:

- all’avvio ottiene il riferimento della zona dal registro

- procede alla richiesta di registrazione alla zona associata
- in caso di esito positivo entra nello stato di lavoro al fine di eseguire il suo compito
- in caso di esito negativo rimane in idle

Esso può essere descritto attraverso la seguente FSM:


La struttura dei messaggi che può ricevere è la seguente:

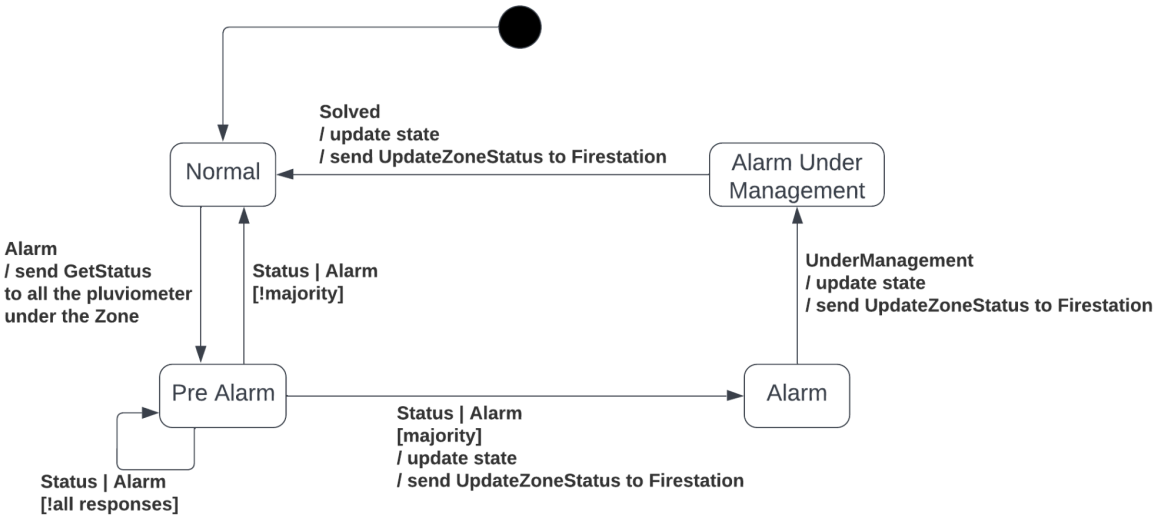
- *Start*: messaggio con cui avviare l'attore
- *PluviometerRegistrationResponse(zoneRef, accepted: Boolean)*: messaggio che rappresenta la risposta alla richiesta di registrazione alla Zona
- *GenerateData*: messaggio che l'attore si auto invia periodicamente al fine di gestire il comportamento con cui vengono generati i dati
- *GetStatus(replyTo)*: messaggio utilizzato dalla Zona al fine di ottenere lo stato corrente

### *Zona*

La logica che gestisce il nodo della zona è composta da un solo attore.
Esso viene configurato attraverso l'*ID* logico associato alla zona stessa.
Il comportamento di una zona è il seguente:

- all'avvio si registra al registro di riferimenti attraverso il suo *ID* logico in modo tale da poter essere trovata dai pluviometri e dalla caserma
- si mette in ascolto della terminazione dei pluviometri
- gestisce le registrazioni dei pluviometri e della caserma
- gestisce l'allarme come descritto nella strategia risolutiva
- durante la fase di allarme rimane in attesa di una gestione da parte della caserma associata
- quando la caserma inizia la gestione dell'allarme, la zona va nello stato di *gestione*
- una volta che la caserma termina la gestione, la zona ritorna in uno stato di normale operatività

Esso può essere descritto attraverso la seguente FSM. La seguente FSM astrae dalle registrazioni dei pluviometri e delle caserme e dalla gestione delle uscite dei pluviometri.




La zona in tutti gli stati si dedica alla gestione delle registrazioni e delle terminazioni dei pluviometri, in particolare:

- *Normal*: registrazione pluviometri, registrazione caserma, uscita pluviometri
- *Pre Alarm*: registrazione caserma, uscita pluviometri
- *Alarm*: registrazione pluviometri, registrazione caserma, uscita pluviometri
- *Alarm Under Management*: registrazione pluviometri, uscita pluviometri

La struttura dei messaggi che può ricevere è la seguente:

- *Alarm(ref)*: messaggio con il quale un pluviometro (ref) segnala una situazione di allarme in un punto della zona
- *RegisterPluviometer(ref)*: messaggio con cui un pluviometro effettua la richiesta di registrazione alla zona stessa
- *RegisterFirestation(ref)*: messaggio con cui la caserma effettua la richiesta di registrazione alla zona stessa
- *Status(ref, alarm: Boolean)*: messaggio di risposta dal pluviometro dopo aver richiesto lo stato.
- *UnderManagement*: messaggio con il quale la caserma segnala alla zona che l'allarme è stato preso in carico
- *Solved*: messaggio con il quale la caserma segnala alla zona che l'allarme è stato risolto

### *Caserma*

La logica che gestisce il nodo della caserma è composta da due attori:

- *Firestation*: gestisce la logica di ogni caserma, come descritta in precedenza
- *FirestationViewActor*: gestisce la dashboard che mostra i dati raccolti

Grazie a questa suddivisione è possibile gestire meglio le responsabilità ma soprattutto si ottiene una maggior flessibilità nel design della logica.
Qui di seguito verrà descritto l'attore principale, cioè *Firestation,* che gestisce tutta la logica.
Esso viene configurato attraverso:

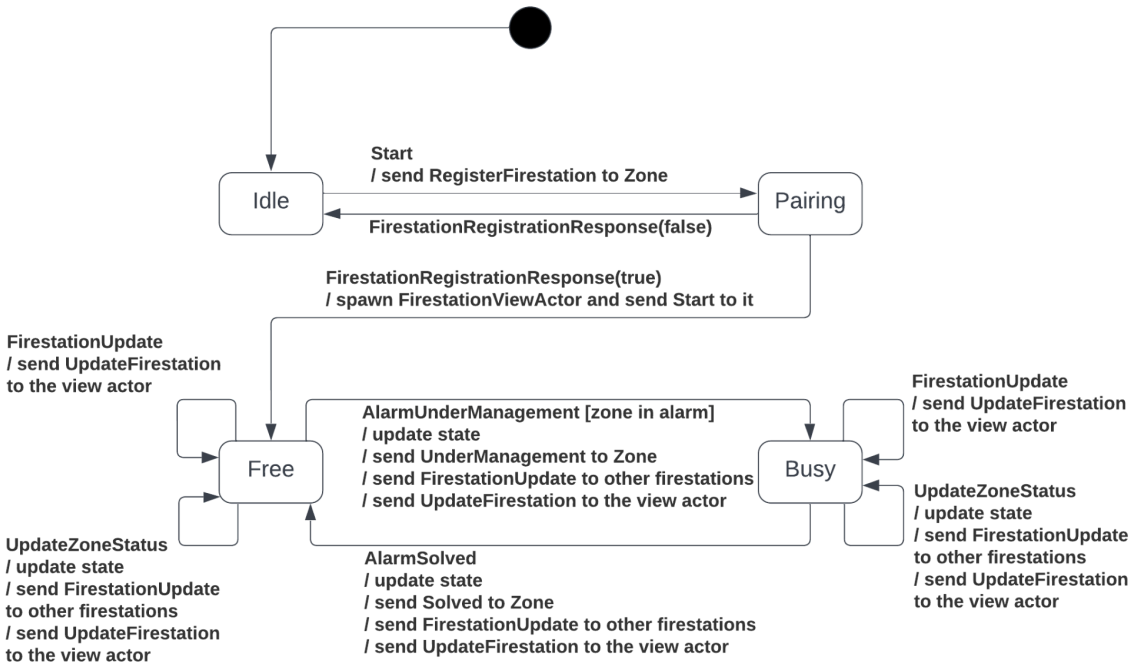
- *id della zona associata*: l'id logico della zona di appartenenza in modo tale che poi possa rivolgersi ad essa.

Il comportamento è il seguente:

- all'avvio ottiene il riferimento della zona dal registro

- procede alla richiesta di registrazione alla zona associata
- in caso di esito negativo rimane in idle
- in caso di esito positivo
    - si registra come caserma in modo tale da poter essere individuato dalle altre caserme per scambiarsi i dati relativi alle zone
    - rimane in ascolto dell'entrata di nuove caserme
    - avvia l'attore che gestisce la dashboard
    - si mette a disposizione della zona
- gestisce, come anticipato, l'arrivo degli aggiornamenti da parte delle altre caserme e da parte della propria zona provvedendo per quest'ultimi a condividerli anche alle altre caserme
- quando la propria zona è in allarme si occupa della logica di presa in carico e poi di risoluzione gestendo i messaggi ricevuti dalla dashboard comandata dal personale della caserma

Esso può essere descritto attraverso la seguente FSM:




La struttura dei messaggi che può ricevere è la seguente:

- *Start*: messaggio con cui avviare l'attore
- *FirestationRegistrationResponse(ref, accepted: Boolean)*: messaggio che rappresenta la risposta alla richiesta di registrazione alla Zona
- *AlarmUnderManagement*: messaggio con cui l'attore che gestisce la dashboard segnala che l'allarme è stato preso in gestione dalla caserma
- *AlarmSolved*: messaggio con cui l'attore che gestisce la dashboard segnala che l'allarme è stato risolto
- *FirestationUpdate(ref, firestationStatus)*: messaggio inviato da un'altra caserma contenente lo stato aggiornato di quest'ultima
- *UpdateZoneStatus(zone)*: messaggio inviato dalla zona associata con il quale segnala il nuovo stato di quest'ultima

In questo modo la caserma ha una panoramica sullo stato di tutta la città. Però, lo stato che ha a disposizione non è veramente uno stato globale consistente di quanto accade in un particolare istante temporale. Considerati i problemi tipici di un sistema distribuito, una possibile soluzione, in questo caso, poteva essere l'utilizzo dell'algoritmo di "shapshot" di Chandy-Lamport visto a lezione, ma considerando i requisiti del problema, non è stato adottato a favore di questa soluzione.

# 3-Implementazione

Per la soluzione di questo problema si è utilizzato il linguaggio di programmazione *Scala* e il toolkit *Akka* per il paradigma ad attori, con l'ausilio di *Akka Cluster* per la gestione della distribuzione. Considerando le caratteristiche del linguaggio utilizzato, si è preferito adottare l'approccio funzionale. Al tempo stesso, come suggerito dalla documentazione di Akka, in alcuni casi è conveniente mettere assieme allo stile funzionale qualche elemento dello stile ad oggetti.
Seguendo lo stile indicato da Akka, ogni attore è stato modellato interamente all'interno di un *object* di *Scala* in cui sono descritti i messaggi che esso può gestire e la factory per crearlo. L'implementazione di esso così rimane incapsulata al suo interno permettendo una maggior agilità di modifica.
Per i messaggi del protocollo utilizzati solo internamente all'attore, la visibilità è stata ridotta all'attore stesso. Inoltre, al fine di gestire i messaggi di risposta e quindi effettuare le traduzioni dei vari protocolli utilizzati dagli attori è stato sfruttato l'interaction pattern *Adapter* reso particolarmente agile dal toolkit *Akka*. In questo modo si riducono i rischi legati all'utilizzo errato del protocollo progettato.

Il comportamento degli attori è stato modellato attraverso l'utilizzo di FSM. Questa modellazione trova spazio anche all'interno dell'implementazione in quanto l'API di *Akka Typed* permette di modellare naturalmente le FSM. Infatti:

- I vari stati della FSM diventano diversi *Behavior*.
- Ogni *Behavior* (il quale rappresenta uno stato) viene modellato attraverso un metodo separato, memorizzando così lo stato all'interno dei parametri del metodo stesso.
- Il Behavior ritornato rappresenta il prossimo stato. Perciò le transazioni della FSM sono abilitate dal valore di ritorno dei suddetti metodi e dalle reazioni dei *Behavior* che gestiscono i messaggi ricevuti.

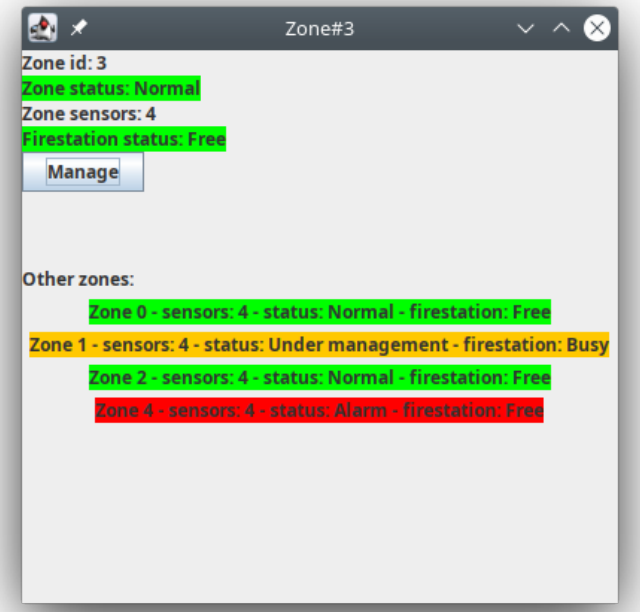
Inoltre, gli elementi comuni a più stati delle FSM sono stati fattorizzati attraverso l'utilizzo di *PartialFunction* separate, come suggerito dalla documentazione di *Akka*, che ha permesso, soprattutto per l'attore *ZoneControl,* di gestire i comportamenti aggiuntivi in modo agile.

Il registro, nominato nella parte di Design, è stato implementato sfruttando il servizio *Receptionist* messo a disposizione da Akka. In questo modo:

- ogni Zona ha il suo *ServiceKey* generato a partire dall'*ID* logico associato, così che i pluviometri e la caserma possano trovarla in maniera più agile, semplificando i protocolli di scambio di messaggio
- le caserme si registrano tutte allo stesso *ServiceKey* che rappresenta il gruppo di caserme della città

Inoltre, ogni zona rimane in ascolto ed è reattiva alla terminazione dei pluviometri appartenenti alla propria zona grazie al *Membership Service* offerto da *Akka Cluster*.

Infine, la dashboard sviluppata per ciascuna caserma è la seguente:




Essa visualizza in alto lo stato della zona associata alla caserma e lo stato della caserma stessa. Il pulsante permette alla caserma di gestire e poi successivamente risolvere un allarme presente nella propria zona.
Dopodiché, in basso è possibile visualizzare invece lo stato di tutte le altre zone e delle rispettive caserme.