Indice

TEMA DI PROGETTO

Attraverso un sistema di soglie di allarme a più livelli, la Protezione Civile dà l'allerta per fenomeni di maltempo (bufere, piogge violente, nevicate intense...) e altri rischi (terremoti, ecc.).

Le allerte sono riferite ai codici di avviamento postale (CAP). I dati sono mantenuti costantemente aggiornati sulla base dei calcoli di vari sottosistemi software esterno che elaborano le previsioni per i diversi fenomeni, nell'arco delle successive 24 ore, con granularità 4 ore.

Il sistema registra tutte le previsioni dei fenomeni/eventi e anche l'effettivo accadimento degli stessi. Ogni evento è descritto da un ID, un testo, un CAP e una sorgente. Ogni evento ha inoltre un tipo. Il sistema tiene traccia di tutti gli eventi ricevuti in ordine di tempo. Gli utenti possono cercare gli eventi per istante temporale, per tipo o per CAP.
Il sistema tiene anche traccia degli eventi correnti. Per evento corrente si intende un evento in corso per ogni CAP nel giorno corrente. I restanti eventi passano in archivio storico. Se per un certo CAP in una certa data sono presenti più eventi, si considera un ordinamento per livello di gravità più elevato.

Esiste un elenco di allarmi in evidenza, che sono quelli di massima gravità per fascia oraria e per CAP. Se in base a previsioni successive un'allerta rientra, questa viene rimossa dalle allerte in evidenza, e si pone in evidenza l'eventuale allerta di livello immediatamente inferiore.

Specificare, progettare e implementare il sistema distribuito necessario, coprendo: sistema centrale, applicazione di previsione, applicazione mobile degli utenti che possono ricevere notifiche degli allarmi. Definire esplicitamente tutti i formati dei dati scambiati e le modalità di scambio.

ANALISI DI FATTIBILITÀ

I. Obiettivi

Il nostro progetto si basa sullo sviluppo di un applicativo software che generi delle previsioni per fenomeni di natura meteorologica e sismica. In questo modo, la Protezione Civile può informare tempestivamente i cittadini, prevenendo e riducendo gli eventuali rischi.
La generazione di una previsione comprende: un ID, il CAP della zona in cui è prevista l'allerta, la data e la fascia oraria, il grado di allerta, la tipologia del fenomeno e una breve descrizione di quest'ultimo.
Vi sono tre diversi gradi di allerta: *heavy (3), strong (2), moderate (1).*

Ogni regione italiana dispone di una propria stazione meteorologica la quale, analizzando i dati forniti dai diversi sensori opportunamente dislocati sul territorio, genera ogni mezz'ora eventuali allerte per i vari CAP di quella determinata regione. Tali allerte sono valide per le successive 24 ore. Nel caso in cui venga generata una previsione di allerta rientrata, questa viene inoltrata al sistema centrale, che si occupa di rimuovere l'allerta corrispondente dal database, purché tale allerta sia rientrata entro le quattro ore precedenti all'evento.

Il sistema centrale riceve, ogni quattro ore, le allerte dall'applicativo esterno e le inserisce in un database. Nel caso in cui un'allerta si riferisca ad una voce già inserita nel database, il sistema si limita ad aggiornarne il grado di allerta. Durante la fascia oraria per cui l'allerta è prevista, il sistema verifica il suo effettivo accadimento, inserendo l'informazione nel database.

Il sistema centrale tiene traccia, per ogni CAP, delle allerte previste per il giorno corrente. Se per un CAP sono previste più allerte lo stesso giorno, queste vengono ordinate per gravità.
Vi è inoltre un elenco di allarmi in evidenza, ovvero gli allarmi di massima gravità per fascia oraria e per CAP. Se nelle previsioni successive un'allerta rientra o diminuisce di grado, questa viene rimossa dall'elenco e viene promossa come allerta più grave quella immediatamente inferiore.
Infine, il sistema centrale inoltra agli utenti, in modo automatico e in base al CAP, le allerte previste per la fascia oraria corrente.

L’utente riceve automaticamente le notifiche delle allerte relative al proprio CAP. Quando l’utente avvia l’applicazione, viene mostrato di default l’elenco degli allarmi validi per il proprio CAP per le successive 24 ore.

L’utente ha la possibilità di consultare l’elenco degli allarmi correnti, ossia gli eventi in corso per ogni CAP nel giorno corrente, oppure l’elenco degli allarmi in evidenza, ovvero gli allarmi di massima gravità per fascia oraria e per CAP.

L’utente può infine consultare i dati sulle allerte presenti nel database del sistema centrale della Protezione Civile, filtrando le ricerche per CAP, fascia oraria e/o per tipo.

II. Costi

- Costi di sviluppo software e testing;
- Costi di manutenzione e aggiornamento del software;
- Costi hardware, tra i quali server e sensori di rilevamento dei fenomeni trattati;
- Costi di manutenzione dell’hardware;
- Costi di elettricità.

III. Risorse necessarie

Abbiamo messo insieme un team di quattro persone in modo da garantire un giusto compromesso tra efficienza, costi e tempistiche di sviluppo.

IV. Rischi e ostacoli

- Inesperienza nello sviluppo di applicativi software;
- Totale fallimento nel caso di un guasto del collegamento della rete Internet;
- Malfunzionamento dei sensori causa esposizione costante ad intemperie;
- Scheduling delle date in cui il team si incontra.

V. Soluzioni

- Costante confronto con persone esperte in materia, quali docenti e tutor, al fine di risolvere dubbi e perplessità;
- Stipulare accordi/convenzioni con compagnie telefoniche per invio di sms che notifichino agli utenti le allerte gravi;
- Utilizzare un numero di sensori più ampio in modo tale da sopprimere eventuali disagi causati dal guasto di un singolo sensore. Questa soluzione comporta un aumento dei costi per quanto riguarda la parte hardware;
- Utilizzare il più possibile strumenti di lavoro collaborativo come GitLab.

VI. Conclusioni

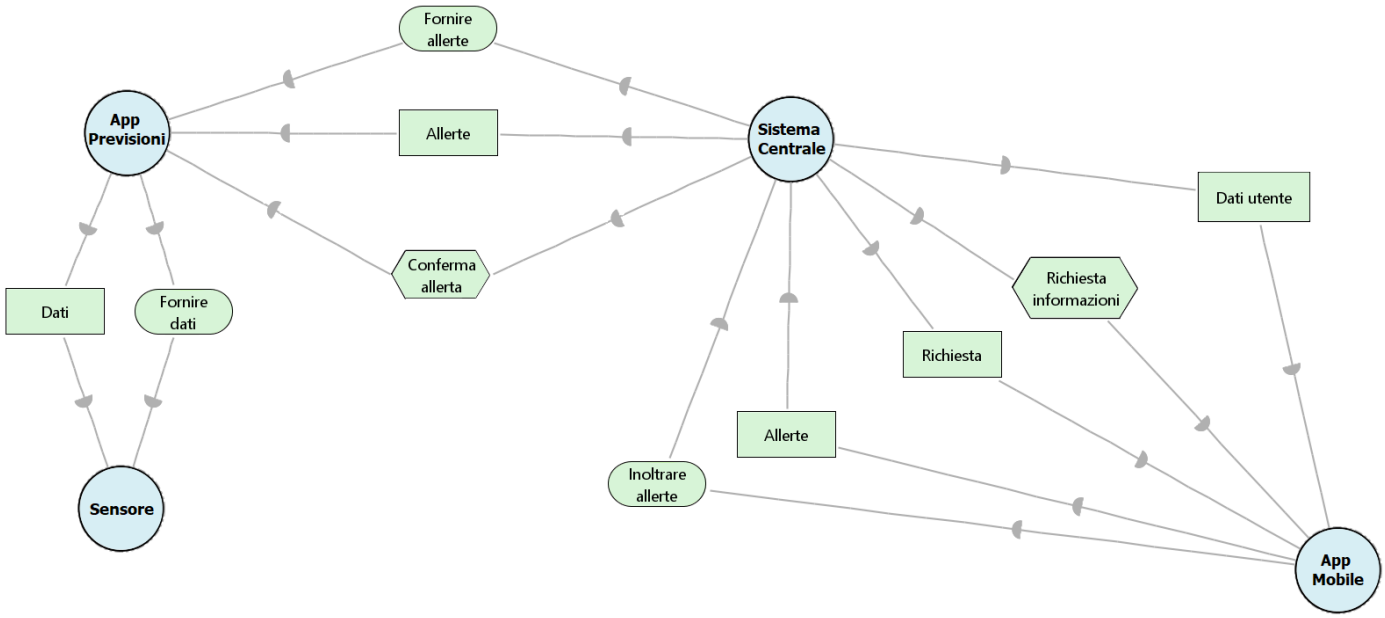
A seguito della precedente analisi, possiamo concludere che il progetto è fattibile e realizzabile nei tempi e nei costi prestabiliti.

DATA DICTIONARY

NOME	SINONIMI	DESCRIZIONE	ESEMPI D'USO	TIPO DI DATO	COMPONENTI	RELAZIONI
Sensore	-Rilevatore -Misuratore	Dispositivo che misura una grandezza fisica in ingresso e fornisce un segnale in uscita a fini di misurazione o di controllo del sistema in cui è impiegato.	Rileva 5mm di pioggia, invia il dato all'App Previsioni.		-Rilevazione -Dato	-App Previsioni -Sistema Centrale
Rilevazione	-Misura -Osservazione	Grandezza fisica rilevata dal sensore.	-2mm; -4km/h;	Numerico		-Sensore
App Previsioni	-Stazione meteorologica	Applicazione esterna al sistema centrale, al quale invia allerte.	Se riceve in ingresso 80mm di pioggia invia un'allerta al Sistema Centrale.			-Sensore -Sistema Centrale
Sistema Centrale		Sistema tramite il quale la Protezione Civile gestisce le allerte.	Riceve un'allerta di terremoto, aggiorna il Database e notifica gli utenti interessati.		-Database Allerte -Database Utenti	-App Previsioni -App Mobile -Sensore
App Mobile	-Utente	Applicazione che la Protezione Civile mette a disposizione dei suoi utenti per visualizzare le allerte e ricevere notifiche.	Mostra l'elenco degli allarmi in evidenza.			-Sistema Centrale
Dato	-Rilevazione -Informazione	Descrizione elementare.				-Sensore -App Previsioni
Previsione	-Congettura -Ipotesi -Supposizione	Il fatto di supporre ciò che accadrà basandosi sui dati rilevati dal sensore.				
Allerta	-Allarme	Segnale di pericolo a seguito solo di previsioni di eventi pericolosi.	50cm di neve per il CAP 22100 tra le ore 20:00 e le 23:59.	String	-Dato	-App Previsioni -Sistema Centrale -App Mobile
Notifica	-Comunicazione -Messaggio	Messaggio che ha lo scopo di portare a conoscenza del pericolo tutti gli utenti interessati.	"Attenzione! Allerta bufera intorno alle 16:00."	String		-Sistema Centrale -App Mobile
Livello Gravità	-Grado di pericolo	Indica se un evento è più grave di un altro e di conseguenza, se un'allerta è prioritaria rispetto ad un'altra.	Allerta di grado medio.	String		-App Previosioni -Sistema Centrale
Dati Utente	-Dati Registrazione	Racchiudono tutte le informazioni che la Protezione Civile richiede agli utenti della propria applicazione mobile.	Luciano Bozzini, 34070.	String		-Sistema Centrale -App Mobile
Database Allerte	-Storico delle Allerte	Una base di dati dove vengono salvate tutte le allerte ricevute dall'applicazione delle previsioni.		Tupla generica.	-Allerta	-Sistema Centrale
(Elenco) Allarmi In Evidenza		Allarmi di massima gravità per fascia oraria e per CAP.		String	-Allerta	-Sistema Centrale
(Elenco) Eventi Correnti		Eventi in corso per ogni CAP nel giorno corrente.		String	-Allerta	-Sistema Centrale

SINTESI GOAL

- **SDM**



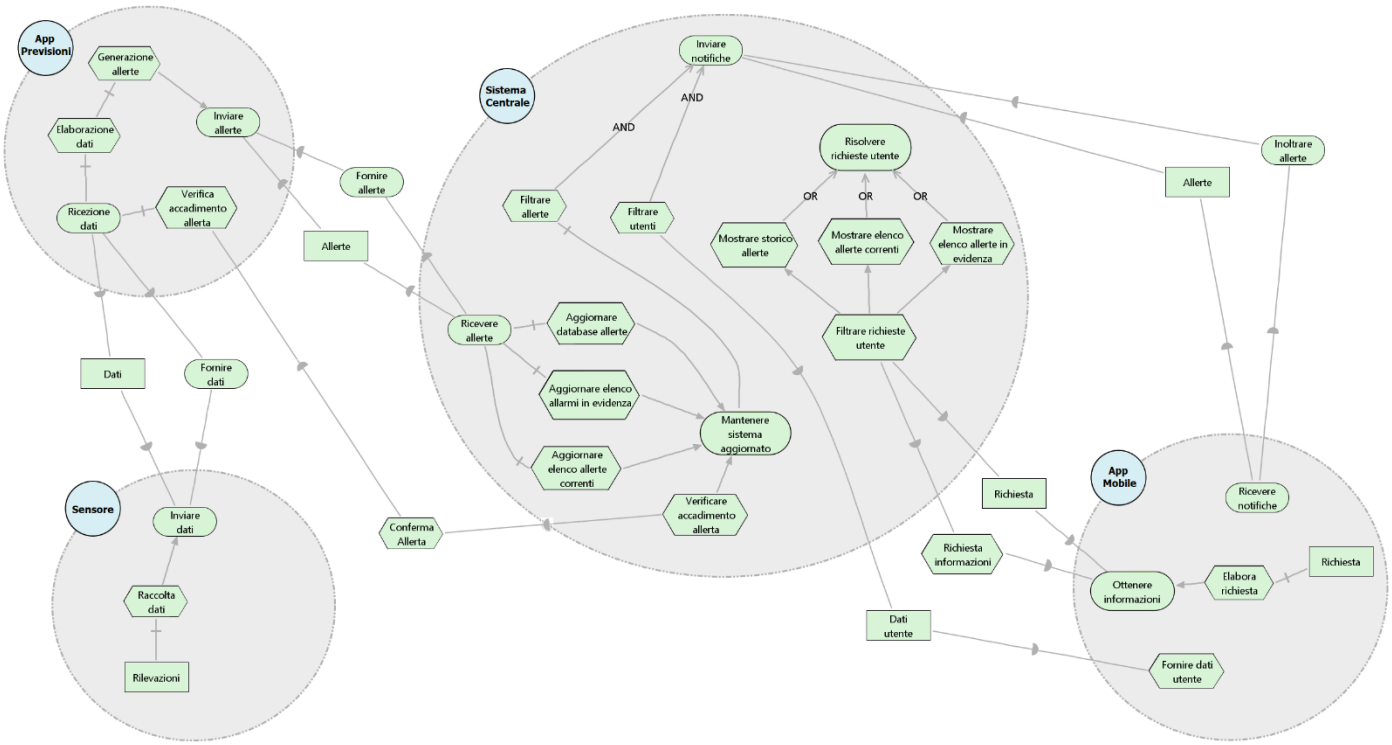
Valutando le richieste, abbiamo individuato quattro attori:

- Sensore;
- App Previsioni;
- Sistema Centrale;
- App Mobile.

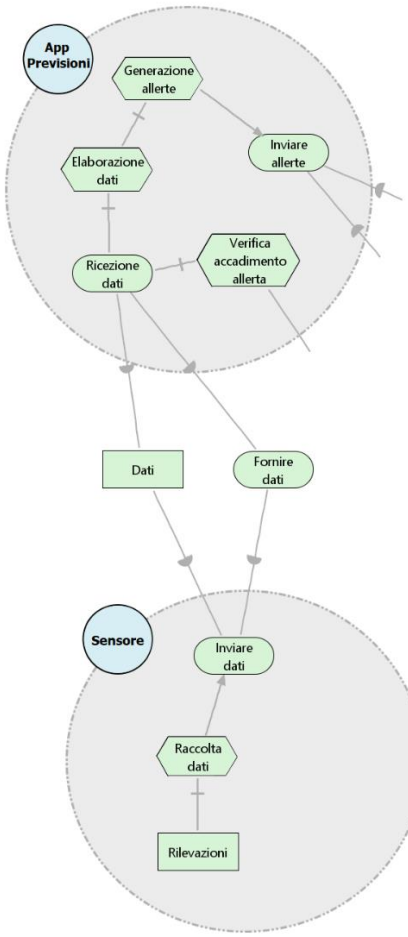
Il Sensore ha il compito di effettuare i rilevamenti per raccogliere dati utili alla generazione delle previsioni. I dati vengono inoltrati all'App delle Previsioni, che si occupa di stabilire le possibili allerte. Queste ultime vengono poi inoltrate al Sistema Centrale, che si occupa del loro invio all'App Mobile degli utenti. Infine, il Sistema Centrale verifica l'effettivo accadimento di un'allerta.

L'App Mobile invia al Sistema Centrale i dati dell'utente, necessari per poter ricevere le varie allerte. Inoltre, può richiedere al Sistema Centrale di accedere ad altre informazioni specifiche inerenti alle allerte.

- **SRM**



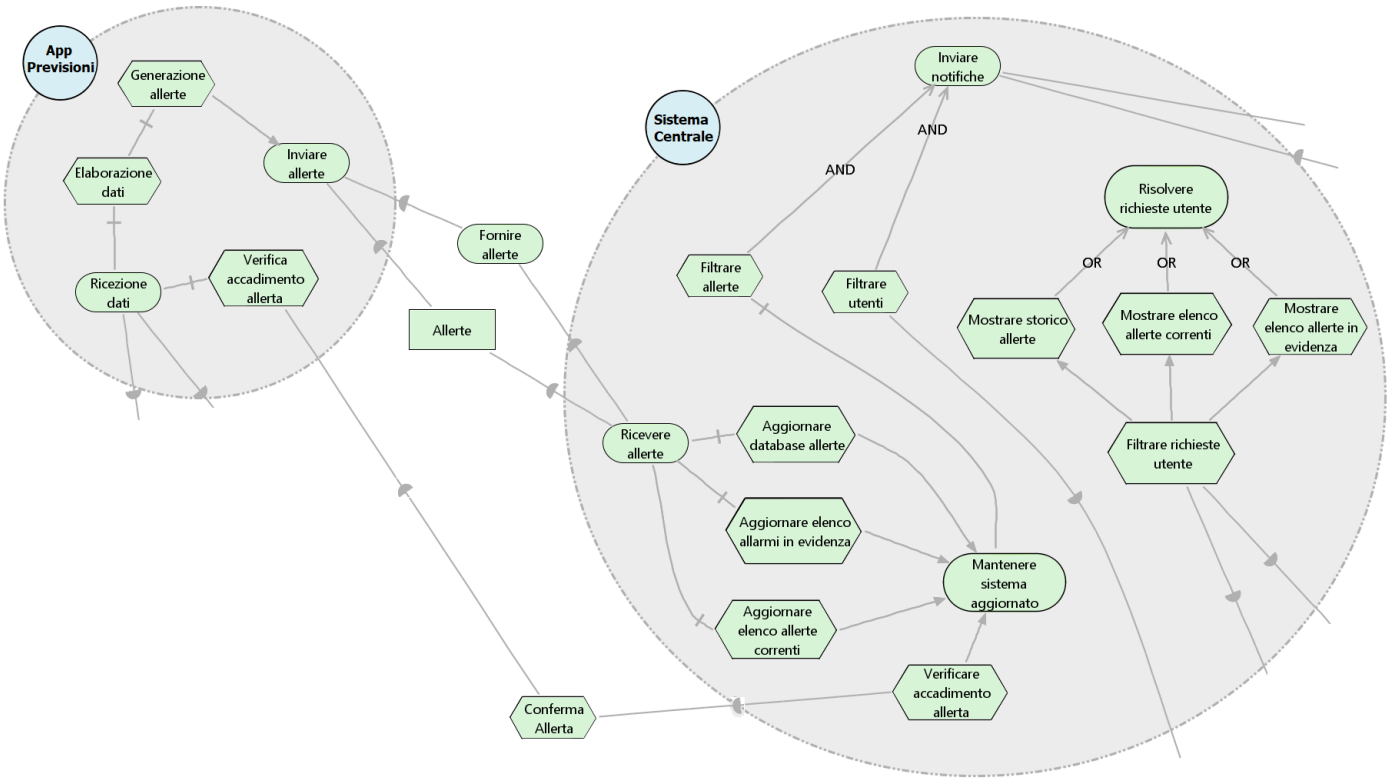
Sensore – App Previsioni



Il Sensore ha il compito di effettuare dei rilevamenti e fornire i relativi dati all'App delle Previsioni.

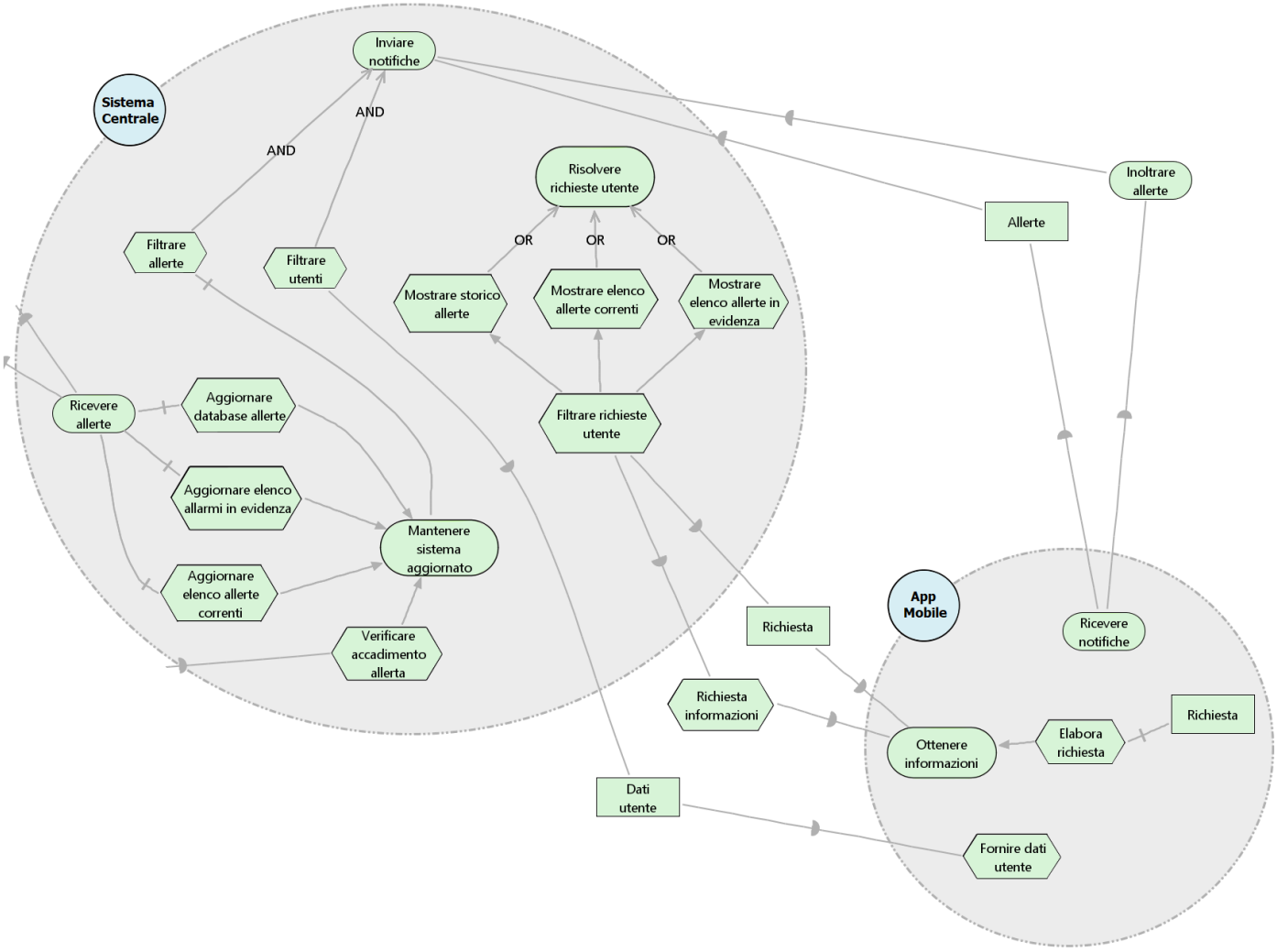
L'App delle Previsioni riceve i dati dal Sensore e a seguito di un'elaborazione, giunge alla vera e propria generazione dell'allerta, che verrà poi inviata al Sistema Centrale.

App Previsioni – Sistema Centrale



L’App delle Previsioni invia l’allerta al Sistema Centrale. Dopo la sua ricezione, il Sistema Centrale aggiorna il database delle allerte, l’elenco delle allerte correnti e l’elenco degli allarmi in evidenza.
Il Sistema Centrale deve poter verificare l’effettivo accadimento di un’allerta. Per farlo, riceve i dati per la conferma dall’App delle Previsioni.

Sistema Centrale – App Mobile



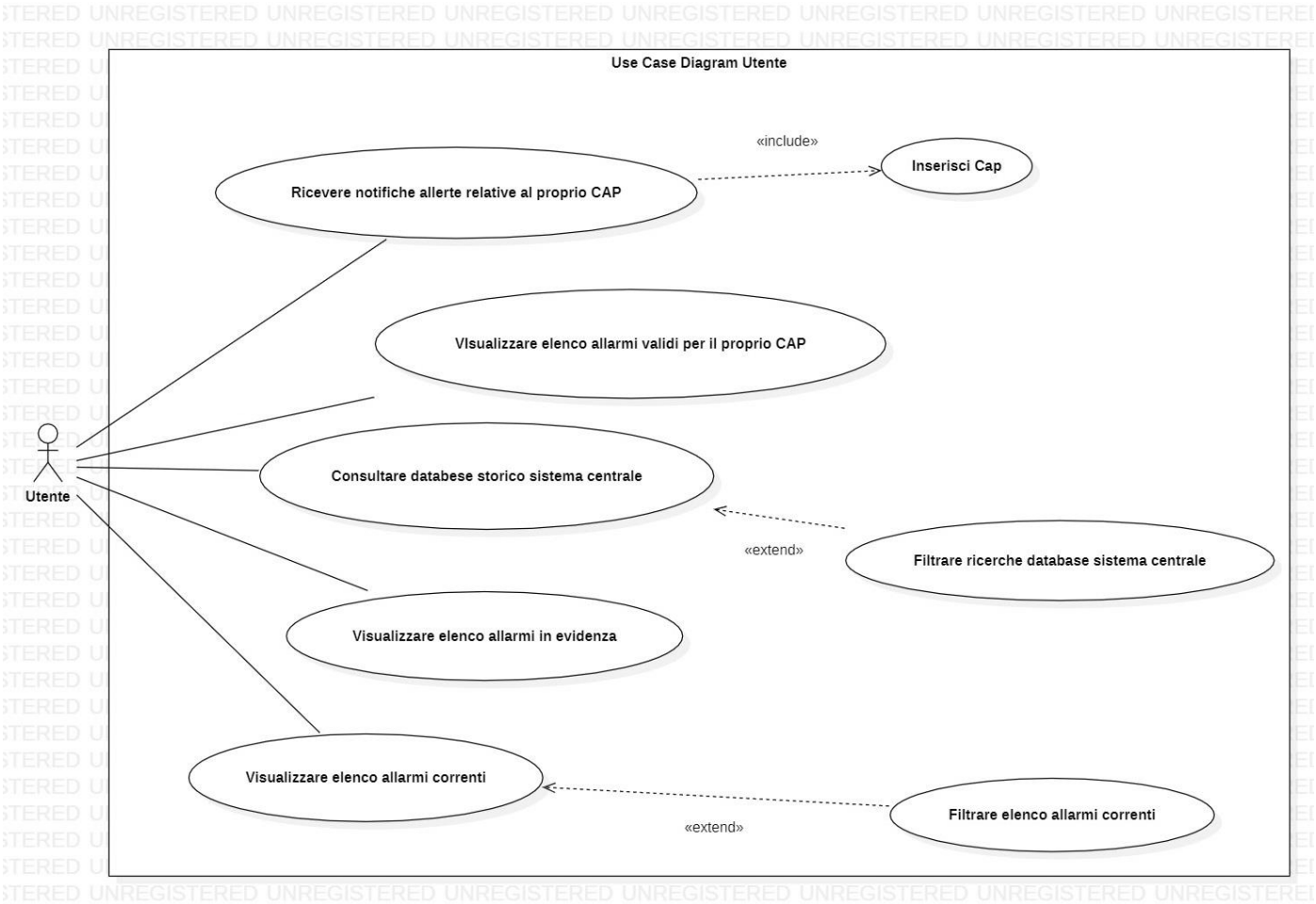
L'App Mobile invia i dati utente al Sistema Centrale. Questi dati sono necessari per notificare poi le eventuali allerte agli utenti.

L'App Mobile riceve dal Sistema Centrale, previo apposito filtraggio del database delle allerte e dei dati utente, le notifiche delle diverse allerte.

Infine, l'App Mobile può inviare altre richieste di informazioni specifiche al Sistema Centrale, quali accedere al database delle allerte, visualizzare l'elenco degli allarmi in evidenza o consultare l'elenco delle allerte correnti.

DESIGN UML

- **Use case diagram**

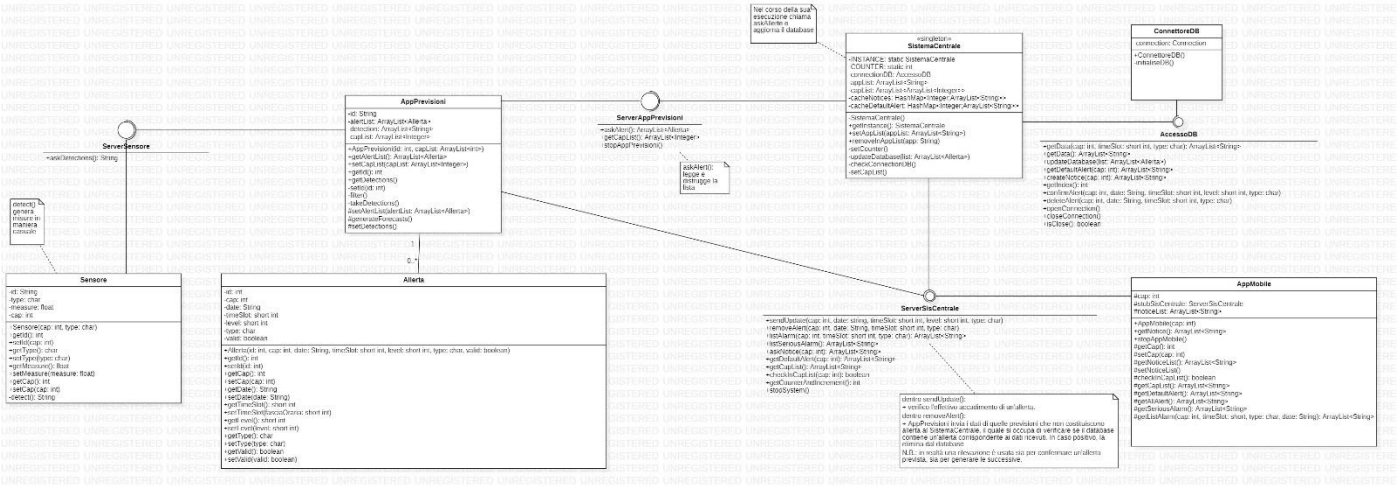


Il diagramma sovrastante descrive tutto ciò che l'utente può fare attraverso l'App Mobile. In esso sono rappresentate inoltre le varie interazioni con il Sistema Centrale.

L'utente può:

- **Inserire il proprio CAP**, questa operazione è necessaria affinché egli possa ricevere notifiche personalizzate delle varie allerte;
- **Ricevere le notifiche delle allerte relative al proprio CAP**;
- **Visualizzare l'elenco degli allarmi validi per il proprio CAP**, per le successive 24 ore. Questo elenco corrisponde alla schermata di default quando viene avviata l'applicazione;
- **Consultare il database storico del Sistema centrale**, filtrando le ricerche per CAP, data, fascia oraria e/o per tipo di evento;
- **Visualizzare l'elenco degli allarmi in evidenza**;
- **Consultare l'elenco degli allarmi correnti**, filtrandolo in base al CAP;

- **Class diagram**



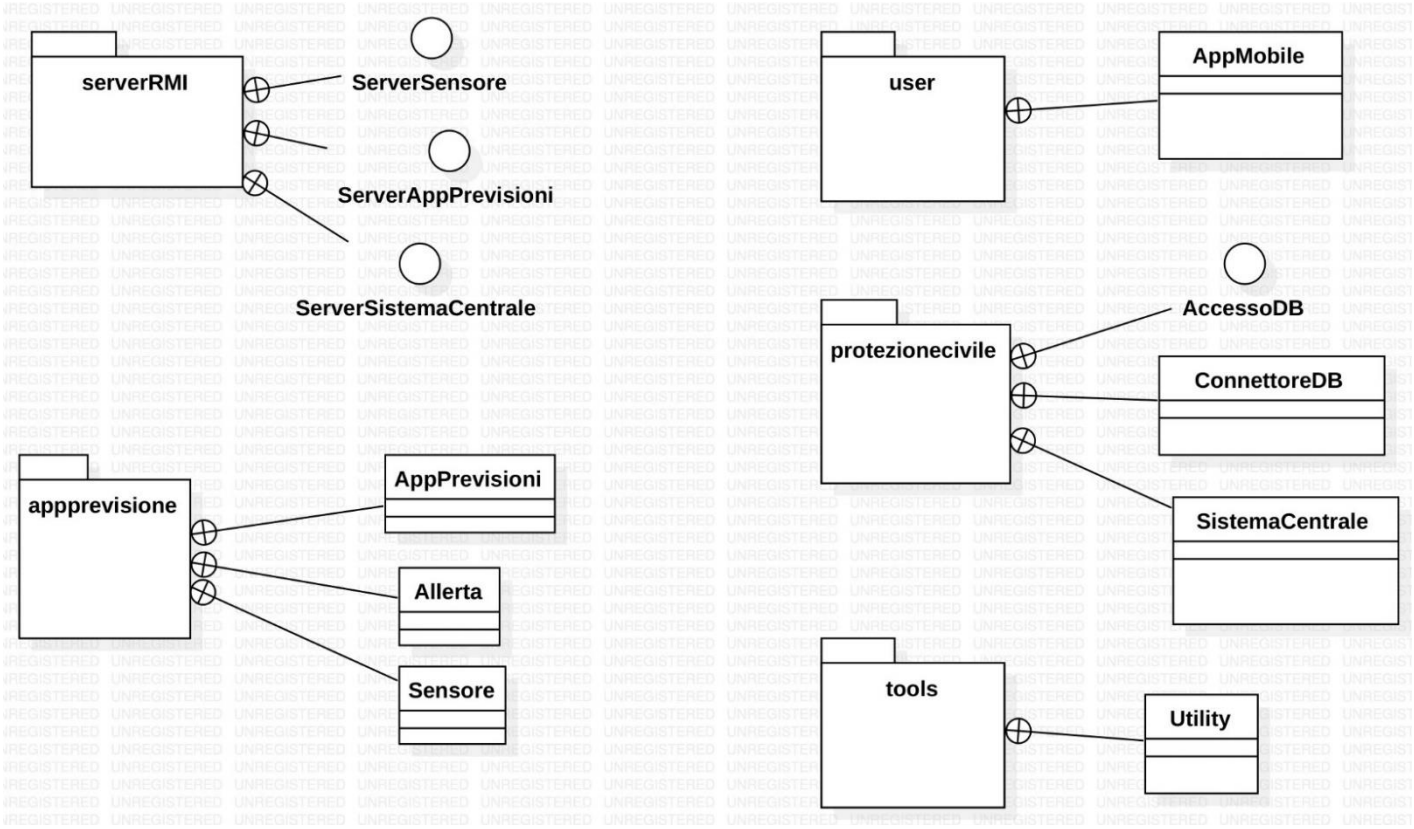
Nella progettazione del Class Diagram si è cercato di mantenere il più fedelmente possibile la struttura già individuata nella sezione Sintesi Goal.

Di seguito, le classi individuate e il loro funzionamento:

- **Sensore**, esegue le rilevazioni che vengono utilizzate per la generazione delle allerte e la conferma delle allerte già presenti nel database;
- **AppPrevisioni**, ottiene le rilevazioni effettuate dal Sensore, invia al SistemaCentrale le informazioni necessarie per confermare, rimuovere e generare allerte;
- **Allerta**, classe utilizzata per memorizzare i dati riguardanti un'allerta;
- **SistemaCentrale**, aggiorna il database con i dati e le allerte che ottiene da AppPrevisioni, crea le notifiche da inoltrare ad AppMobile e si occupa di soddisfarne tutte le richieste;
- **ConnettoreDB**, classe utilizzata da SistemaCentrale per tutte le operazioni da eseguire sul database;
- **AppMobile**, riceve le notifiche dal SistemaCentrale, invia le richieste dell'utente al SistemaCentrale e ne mostra i risultati.

Utilizziamo inoltre le seguenti interfacce: **ServerSensore**, **ServerAppPrevisioni**, **ServerSisCentrale** e **AccessoDB**. Queste rendono possibile la comunicazione rmi tra le varie classi.

- **Package diagram**



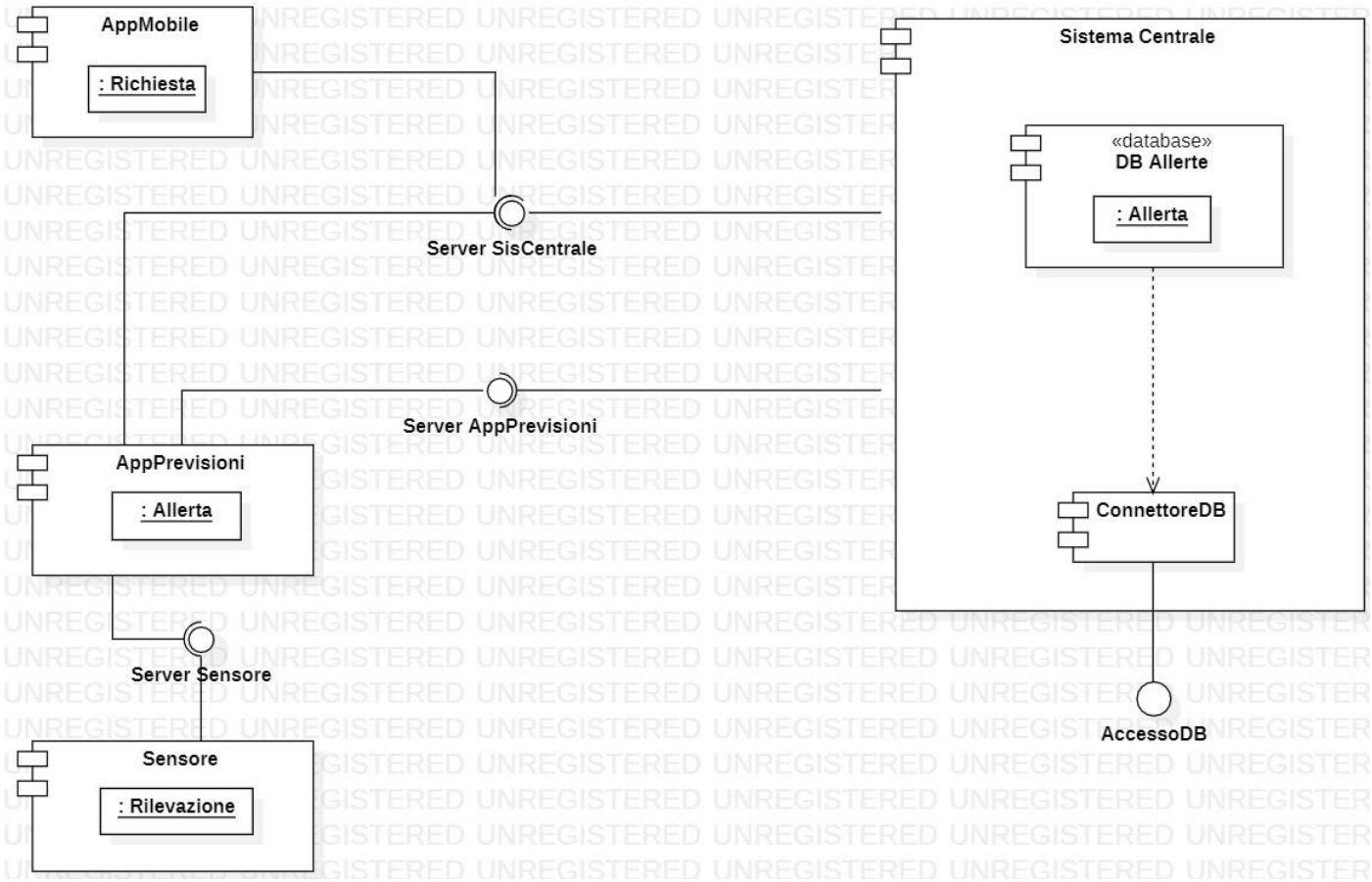
Il diagramma mostra un'organizzazione in package dei vari elementi.

In questo modo è fornita un'organizzazione gerarchica e una visione più chiara e sintetica.

All'interno di esso si distinguono i seguenti packages:

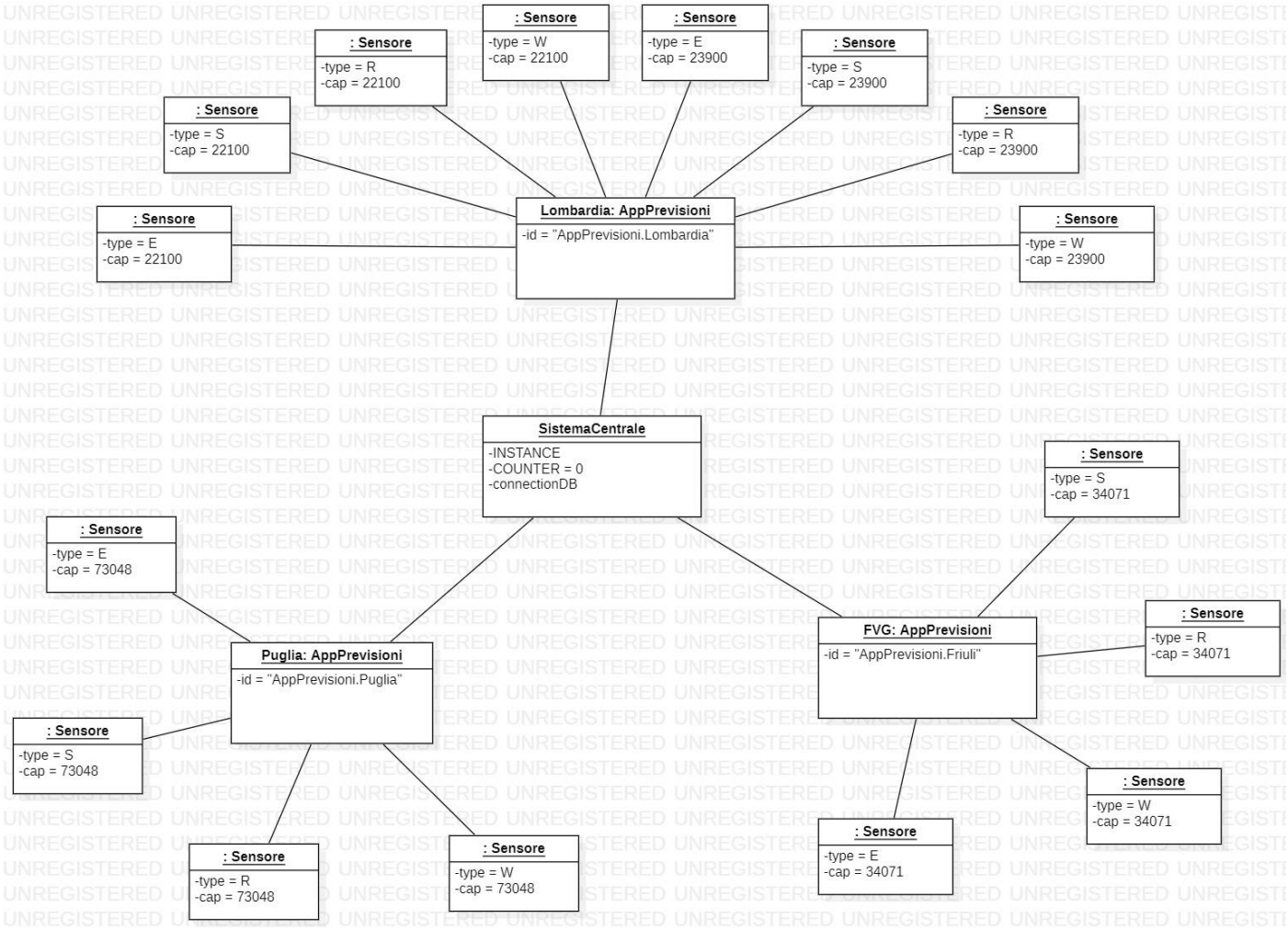
- **serverRMI**, composto dalle interfacce ServerSensore, ServerAppPrevisioni e ServerSistemaCentrale;
- **appprevisione**, composto da AppPrevisioni, Allerta e Sensore;
- **user**, composto da AppMobile;
- **protezionecivile**, composto dall'interfaccia AccessoDB e dalle classi ConnettoreDB e SistemaCentrale;
- **tools**, composto dalla classe Utility.

- **Component diagram**



Il diagramma mostra i componenti dell'intero sistema e le loro modalità d'interazione. Le comunicazioni tra i vari componenti avvengono tramite interfacce RMI (Server SisCentrale, Server AppPrevisioni, Server Sensore). L'interfaccia AccessoDB è ad uso esclusivo del Sistema Centrale e consente a questo solo componente di effettuare operazioni di ricerca e di modifica sul database.

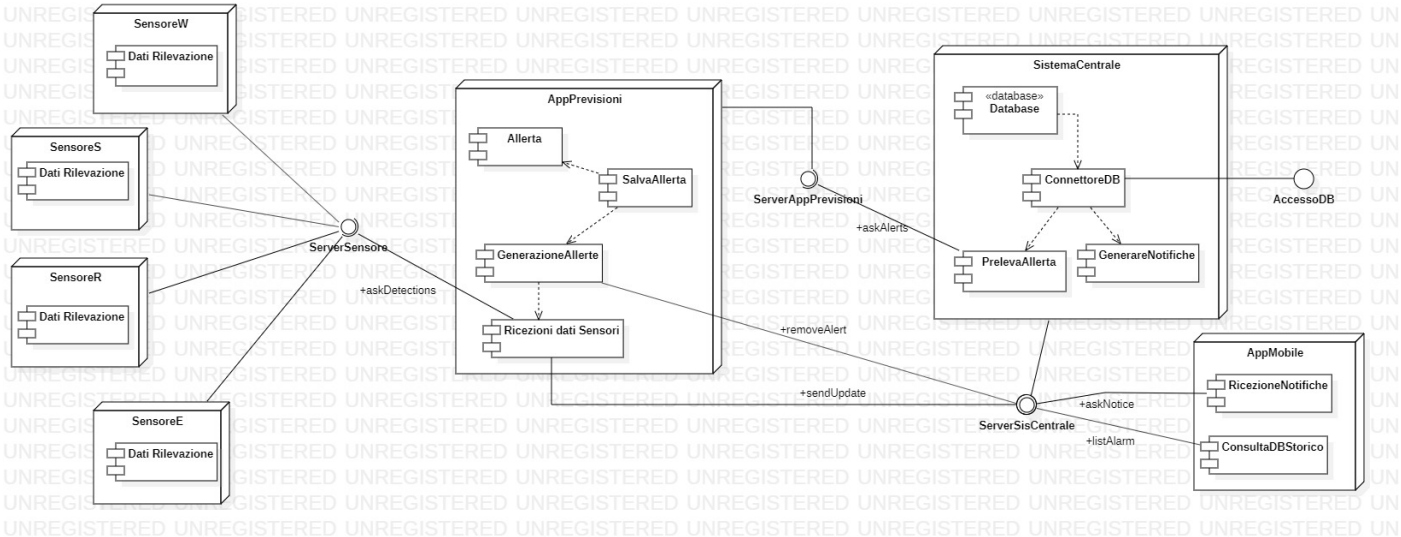
- **Object diagram**



Il diagramma fornisce un chiaro esempio di come sia organizzato il sistema della generazione e gestione delle allerte. Per ogni CAP vi sono quattro sensori, uno per ogni tipologia di dati da rilevare. Ciascun sensore comunica con l'applicazione di previsione della propria regione, fornendogli ogni mezz'ora tutti i dati dell'ultima rilevazione effettuata. L'applicazione di previsione, analizzati i dati ed effettuate le proprie previsioni, comunica a sua volta con il sistema centrale della Protezione Civile per:

- comunicare nuove allerte;
- rimuovere le allerte rientrate;
- verificare l'effettivo accadimento di un evento previsto.

- **Deployment diagram**

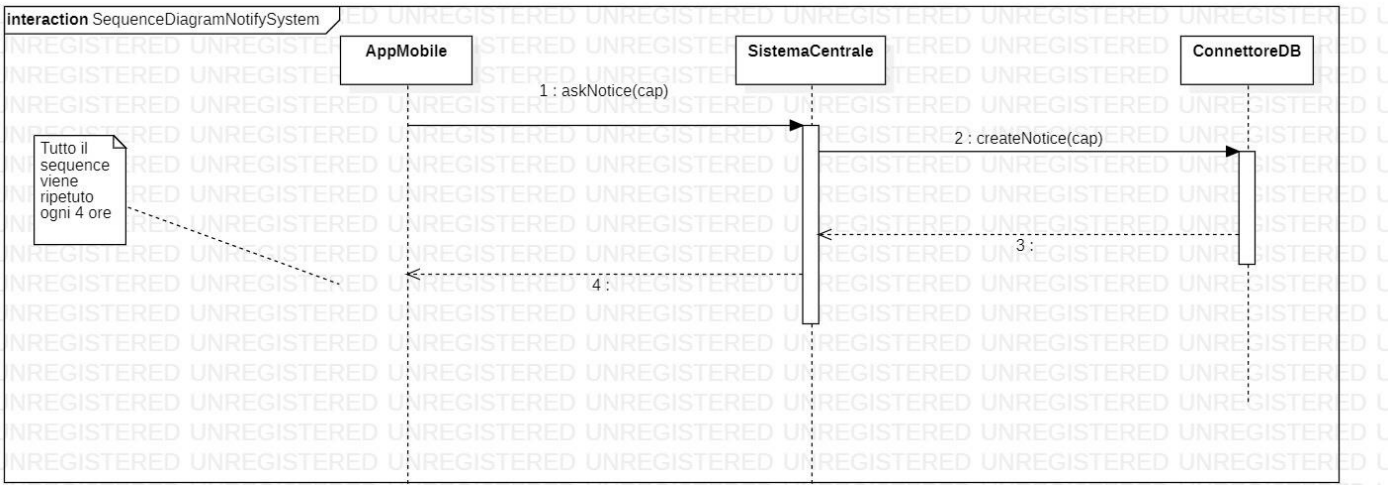


Il diagramma ci mostra l'organizzazione dei componenti interni ai nodi: Sensori, AppPrevisioni, Sistema Centrale e AppMobile. È inoltre possibile visualizzare tutte le interfacce RMI che collegano i vari nodi del nostro progetto.

Segue una breve descrizione dei componenti:

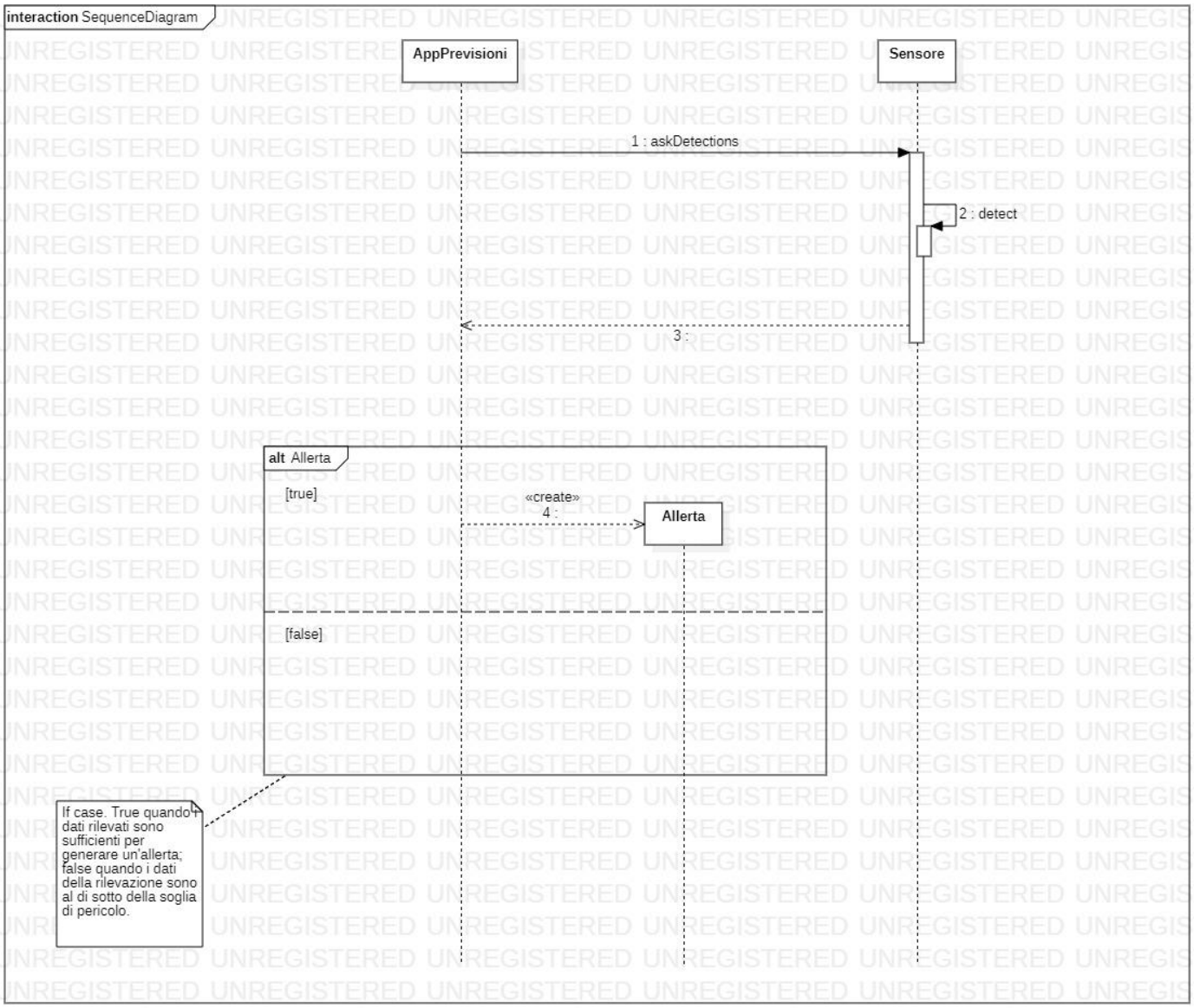
- **Sensori**, si occupano di fornire i dati relativi alle rilevazioni. Ogni sensore fornisce dati relativi alla propria tipologia;
- **AppPrevisioni**, si occupa di raccogliere i dati dai sensori (quattro per ogni CAP che gestisce l'AppPrevisoni) e di generare delle previsioni che possono o meno diventare allerte. Nel caso in cui diventino allerte, queste vengono memorizzate al fine di consentire al SistemaCentrale di prelevarle;
- **SistemaCentrale**, si occupa di prelevare le allerte dalle AppPrevisioni, di memorizzare tali allerte attraverso il ConnettoreDB e di generare le notifiche che saranno prelevate dall'AppMobile;
- **AppMobile**, si occupa di ricevere le notifiche dal sistema centrale e di consentire all'utente di effettuare delle ricerche nel database storico gestito dal sistema centrale;

- **Sequence diagram – Gestione notifiche utente**



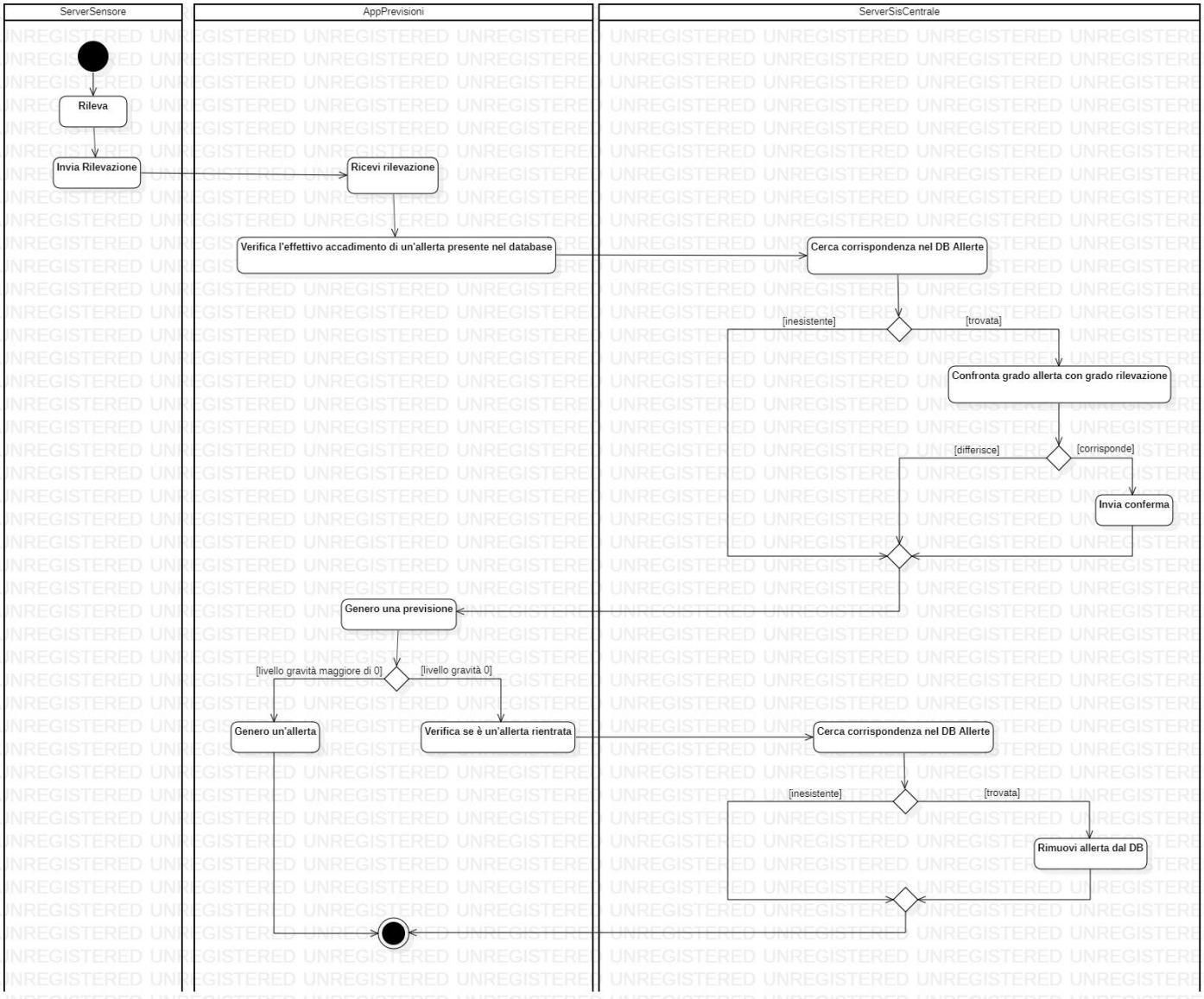
Il diagramma ci mostra quella che è una parte fondamentale del nostro progetto, ovvero come l'AppMobile richieda le notifiche. Per farlo, deve instaurare una connessione rmi verso il SistemaCentrale e invocare il metodo rmi "**askNotice"**. Quest'ultimo a sua volta chiama "**createNotice**" del ConnettoreDB, che si occupa di estrapolare a run-time i dati che compongo la nostra notifica.

- **Sequence diagram – Generazione allerta**



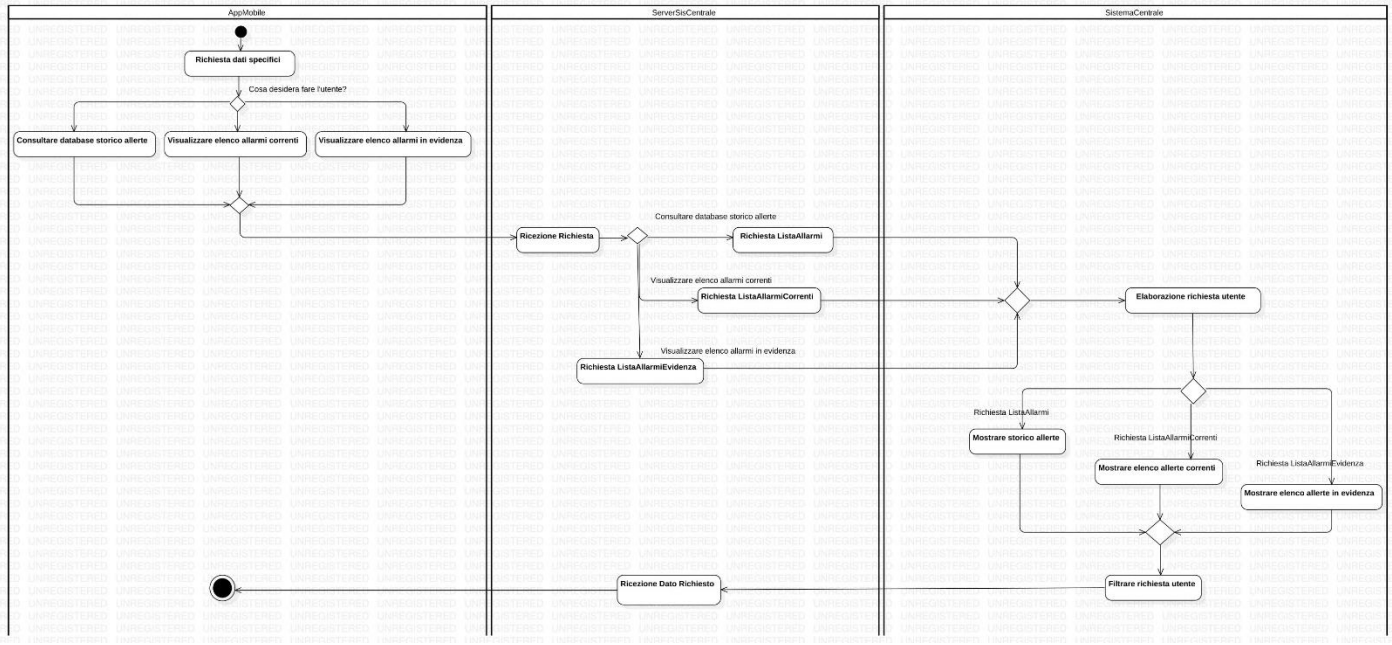
Il diagramma illustra la sequenza di passaggi necessari affinché un'allerta venga generata. L'applicazione di previsione, attraverso il metodo '**askDetections**', "chiede" una rilevazione ad uno dei suoi sensori. Il sensore effettua, quindi, una rilevazione attraverso il proprio metodo '**detect**' e risponde all'applicazione inviando i dati ottenuti. Ottenuta la rilevazione, l'applicazione procede con l'esaminarne il dato '**measure**'; se la misura risulta essere al di sopra di una certa soglia e quindi indice di pericolo, allora viene generata una nuova allerta sulla base dei dati della rilevazione ottenuta dal sensore; se, invece, la misura è al di sotto della soglia di pericolo, la rilevazione viene ignorata e si procede ad esaminarne un'altra.

- **Activity diagram – Gestione allerte rientrate e correnti**



Il diagramma espone, in modo semplice, le azioni che vengono eseguite dal sistema per aggiornare il database. Si parte dal sensore, il quale effettua ogni 30 minuti una rilevazione che viene ricevuta dall'applicazione di previsione a cui appartiene. Subito dopo, l'applicazione invia i dati della rilevazione al sistema centrale della Protezione Civile per permettergli di verificare che nel database vi sia un'allerta corrispondente e che quest'ultima abbia lo stesso livello di gravità, in modo da poterne confermare l'effettivo accadimento. Fatto ciò, l'applicazione genera una previsione basandosi sulla rilevazione del sensore. Se il livello di gravità è superiore allo 0, allora vuol dire che si prevede una situazione di pericolo nella fascia oraria assegnata alla previsione in questione, viene pertanto generata una nuova allerta, che viene inviata al SistemaCentrale. Se, invece, il livello di gravità è pari a 0, allora la previsione smentisce un’eventuale allerta generata in precedenza a seguito di una previsione errata e viene mandata una segnalazione al SistemaCentrale per rimuoverla dal database.

- **Activity diagram – Gestione richieste utente**

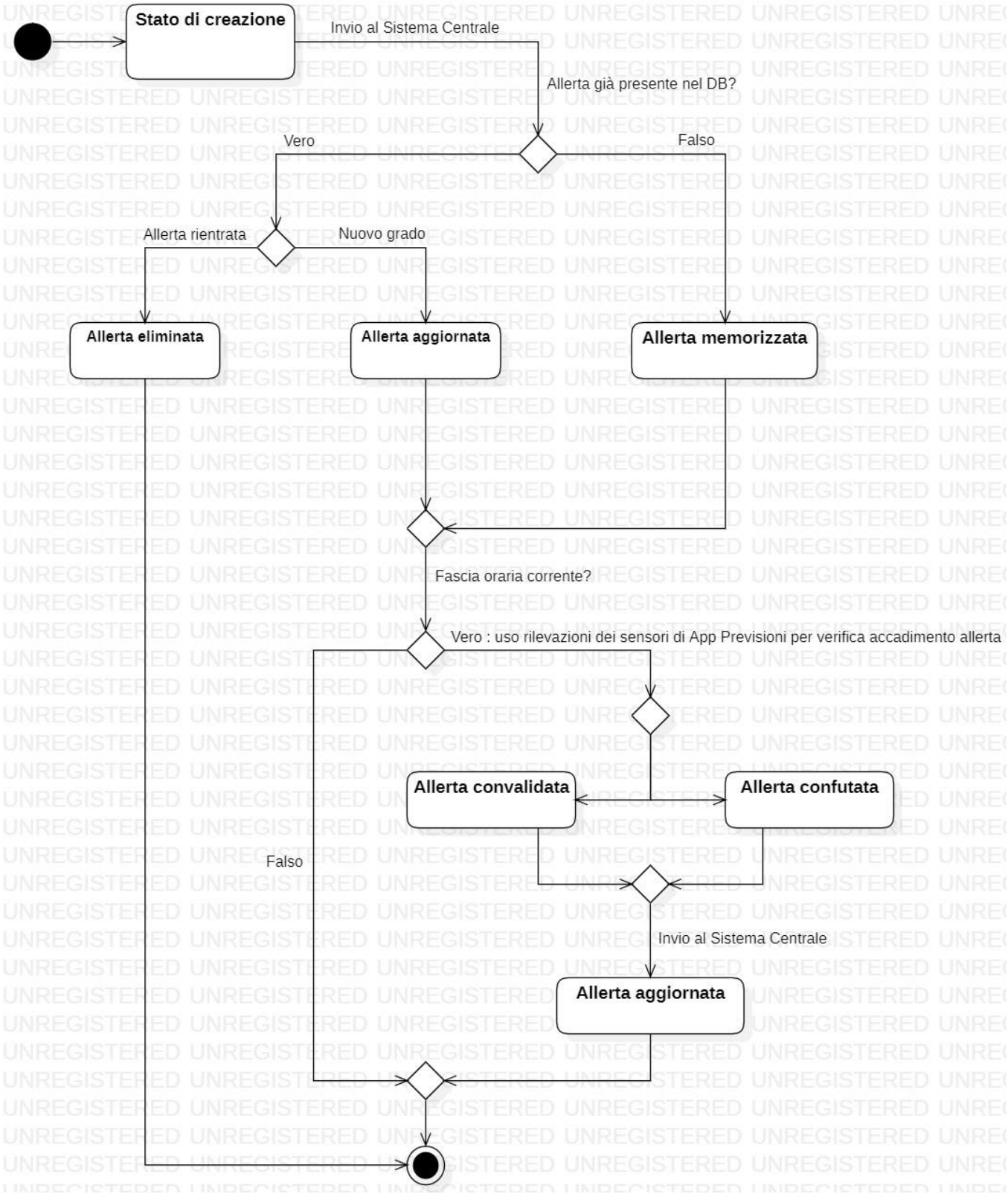


L'Activity mostra le varie fasi per la richiesta di dati specifici da parte dell'utente.

Per prima cosa, l'utente effettua una specifica richiesta, tra Consultare lo storico delle allerte, Visualizzare l'elenco degli allarmi correnti e Visualizzare l'elenco degli allarmi in evidenza.

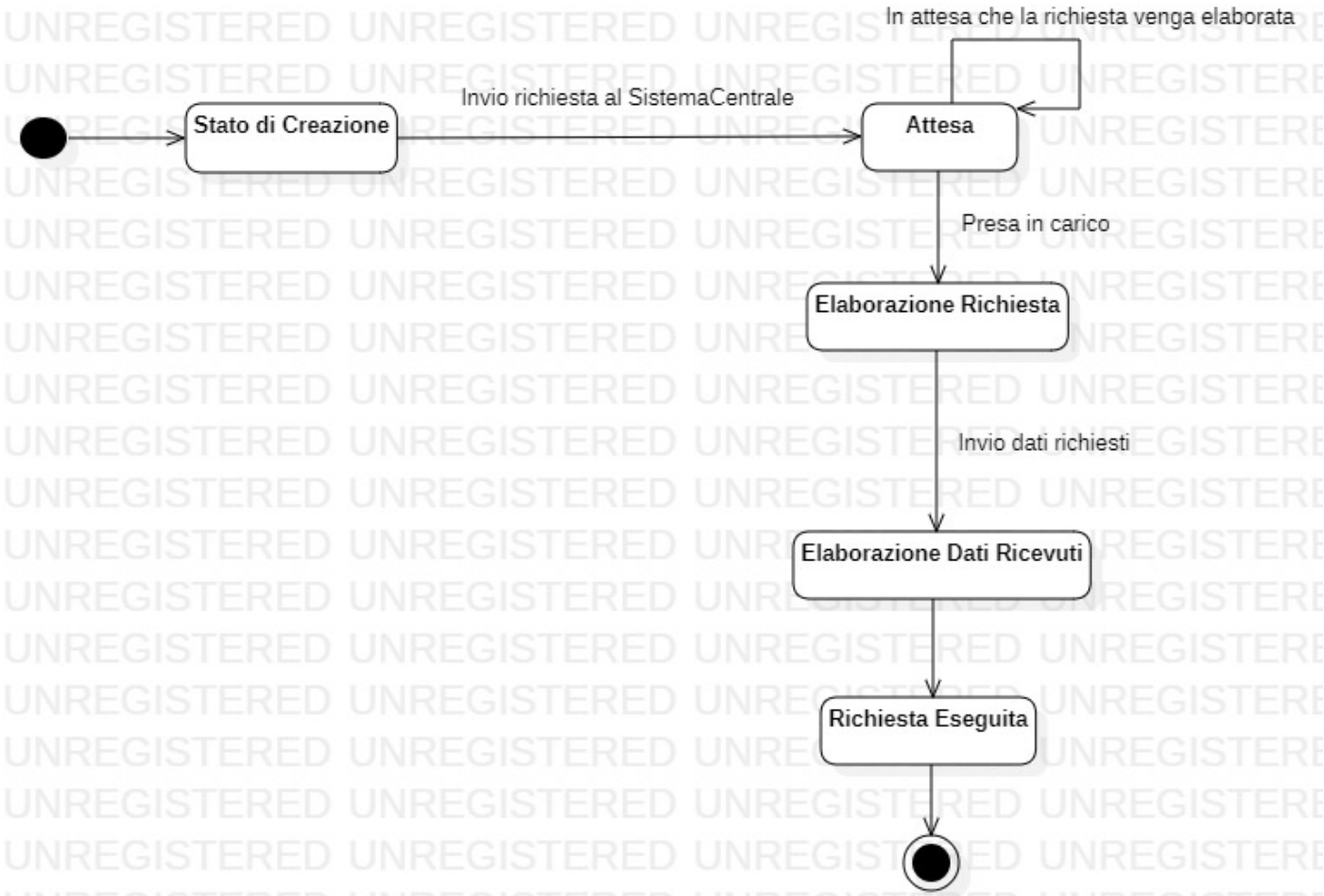
Questa richiesta viene inoltrata al Sistema Centrale (attraverso l'invocazione di un metodo opportuno). Il Sistema Centrale, dopo aver elaborato la richiesta, filtrerà i dati in base a quanto specificato dall'utente, inoltrandoli all'AppMobile.

- **State diagram – Allerta**



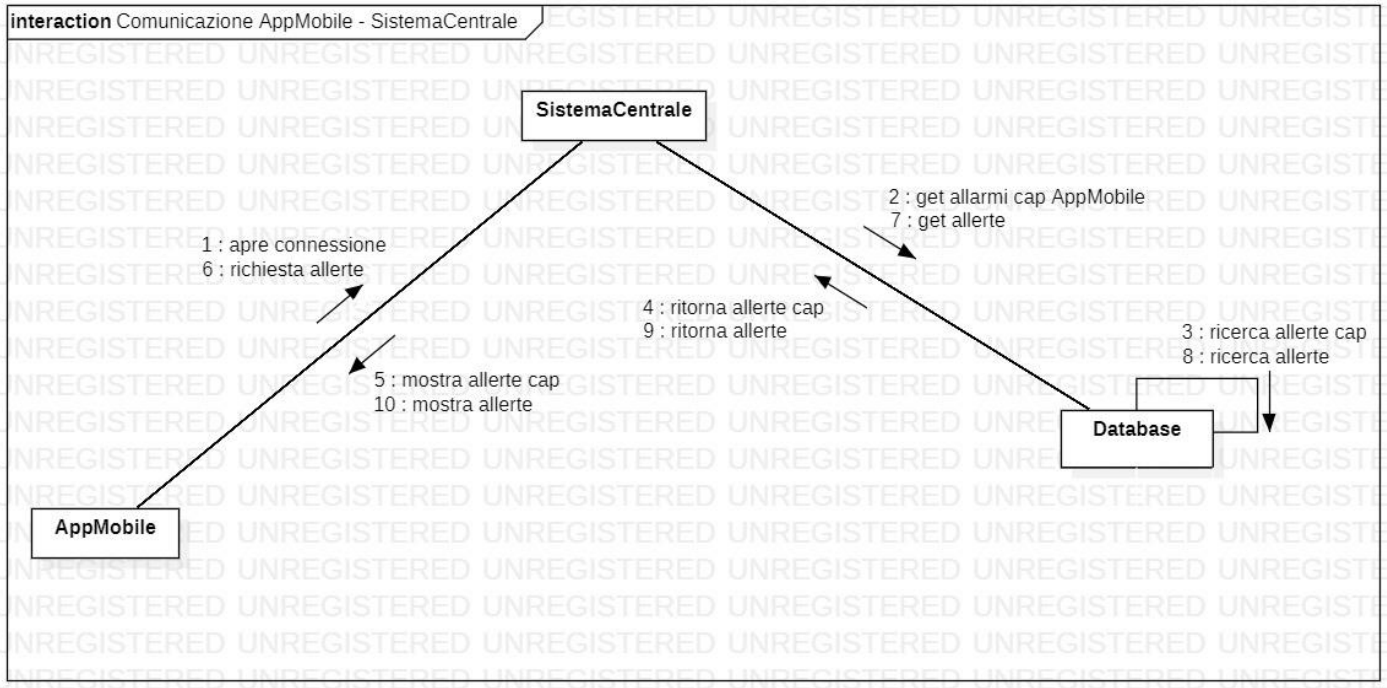
Il diagramma descrive tutti i possibili stati che un'allerta può assumere nel suo ciclo di vita.
Dopo la sua creazione, l'allerta viene inviata al SistemaCentrale, il quale si occupa di verificare se questa sia già presente nel database. In caso negativo, l'allerta viene memorizzata. In caso positivo, l'allerta già presente nel database viene aggiornata con il nuovo grado oppure, nel caso sia rientrata, viene eliminata.
Successivamente, se la fascia oraria corrente corrisponde a quella per cui l'allerta è prevista, si utilizzano le rilevazioni dei sensori per stabilire l'effettivo accadimento dell'allerta. Se anche una sola rilevazione conferma l'allerta, essa viene convalidata. Nel caso in cui, scaduta la fascia oraria per cui l'allerta è prevista, questa non sia mai stata convalidata, viene confutata.

- **State diagram – Notifica utente**



Il diagramma ci mostra quelli che sono gli stati che deve attraversare una qualsiasi richiesta effettuata al SistemaCentrale.

- **Communication diagram**



Il diagramma descrive l'interazione tra AppMobile e SistemaCentrale.
Nel momento in cui AppMobile apre la connessione, il SistemaCentrale ne acquisisce il CAP e ricerca nel database le allerte previste per le successive 24 ore per quel determinato CAP.
L'elenco viene poi inviato ad AppMobile e mostrato all'utente.
Da questo momento, AppMobile può effettuare tutte le interrogazioni che desidera al SistemaCentrale.

IMPLEMENTAZIONE JAVA

- **Sensore**

  Contenuta nel package *appprevisioni*, la classe **Sensore** contiene i seguenti attributi:

  - **id**: univoco per ogni sensore;
  - **type**: indica la tipologia del sensore fra Earthquake (E), Rain (R), Snow (S) e Wind (W);
  - **measure**: indica la misura della rilevazione;
  - **cap**: rappresenta il CAP della zona in cui è posto il sensore.

  Per quanto riguarda i metodi, la classe **Sensore** implementa il metodo **detect**. Esso simula il comportamento reale di un sensore, generando una rilevazione.

  [N.B. Per cercare di rendere le varie rilevazioni più veritiere possibili, il metodo detect non le genera in modo del tutto casuale ma secondo un criterio prestabilito. Ad esempio, è molto più probabile rilevare (generare) un terremoto di magnitudo 3.0 piuttosto che uno di magnitudo 7.0.]

  Infine, generata la rilevazione, il metodo ritornerà CAP e tipologia del sensore, istante temporale e valore della misurazione.

  Il risultato di **detect** verrà poi ritornato dal metodo **askDetections**, contenuto nell'interfaccia **ServerSensore**, attraverso la quale la classe Sensore comunica con la classe AppPrevisioni.

- **App Previsioni**

  La classe **AppPrevisioni,** contenuta nell'omonimo package, ha il compito di analizzare le rilevazioni effettuate dai sensori di ogni CAP, di generare allerte e di inviare aggiornamenti al sistema centrale della protezione civile. Per svolgere al meglio queste funzioni, la classe si avvale dei seguenti attributi e metodi.

  Attributi:
  - **id**, identifica ogni singola applicazione in modo da poterle distinguere l'una dall'altra;
  - **capList**, contiene i CAP assegnati all'applicazione;
  - **detection**, memorizza (finché necessario) le rilevazioni ricevute dai sensori;
  - **alertList**, memorizza (finché necessario) le allerte generate.

  Metodi:
  - **takeDetections**, prende le rilevazioni di ogni singolo sensore di tutti i CAP;
  - **stopAppPrevisioni**, arresta l'esecuzione del thread associato.
  - **generateForecasts**.

  Particolare attenzione va riposta sul metodo **generateForecasts**, il quale esegue più di un compito al suo interno. Per ogni rilevazione, ricevuta attraverso il metodo 'askDetections', estrapola i dati, determina il livello di gravità a seconda del tipo di rilevazione e converte l'ora esatta in fascia oraria, poi conferma o smentisce l'effettivo accadimento di un'allerta corrente, presente nel database del Sistema Centrale, attraverso il metodo 'sendUpdate'. A questo punto, la funzione simula la generazione di una previsione meteorologica (generalmente compito di menti umane o software molto più complessi e progettati con una conoscenza approfondita in materia) assegnando alla rilevazione una fascia oraria, delle prossime 24 ore, casuale. Se il livello di gravità è superiore allo 0, allora vuol dire che si prevede una situazione di pericolo nella fascia oraria assegnata alla previsione in questione, viene pertanto generata una nuova allerta e memorizzata in 'alertList'. Se, invece, il livello di gravità è pari a 0, allora la previsione smentisce un'eventuale allerta generata in precedenza a seguito di una previsione errata. L'AppPrevisioni provvede, dunque, ad inviare un ulteriore aggiornamento al Sistema Centrale attraverso il metodo 'removeAlert', per permettere a quest'ultimo di rimuovere l'allerta rientrata, se presente nel database.

  La classe sfrutta un thread, il quale si occupa di far eseguire ogni 30 minuti i due metodi principali: **takeDetections** e **generateForecasts**.

- **Sistema Centrale**

  Contenuta nel package *protezionecivile*, la classe **SistemaCentrale** contiene gli attributi e i metodi necessari per l'aggiornamento del database, la creazione delle notifiche per l'utente e la risoluzione delle sue richieste.

  Gli attributi della classe sono:
  - **counter**, assegna alle allerte generate dalle AppPrevisioni id univoci;
  - **connectionDB**, consente la connessione al database;
  - **appList**, contiene le AppPrevisioni del programma;
  - **capList**, contiene i CAP presenti nel programma, indicizzati per AppPrevisioni;
  - **cacheNotices**, memorizza le notifiche, indicizzate per CAP, da inviare agli utenti;
  - **cacheDefaultAlert**, memorizza le allerte delle prossime 24 ore, indicizzate per CAP, da mostrare agli utenti quando questi creano un'AppMobile.

  La classe estende l'interfaccia **ServerSisCentrale**, implementandone i metodi:
  - **listAlarm**, interroga il database restituendo le tuple filtrate in base ai parametri passati dall'utente;
  - **listSeriousAlarm**, interroga il database restituendo l'elenco degli allarmi in evidenza;
  - **sendUpdate**, conferma un'allerta presente nel database;
  - **removeAlert**, rimuove un'allerta presente nel database;
  - **askNotice**, aggiorna l'attributo cacheNotices e restituisce l'elenco delle notifiche per il CAP dell'utente;
  - **getDefaultAlert**, aggiorna l'attributo cacheDefaultAlert e restituisce l'elenco delle allerte delle prossime 24 ore per il CAP dell'utente;
  - **getCounterAndIncrement**, ritorna il valore del contatore e ne incrementa il valore di 1. Metodo thread-safe invocato dalle AppPrevisioni per assegnare id univoci alle allerte create;
  - **checkInCapList**, stabilisce se un CAP è presente nel programma;
  - **getCapList**, ritorna la lista dei CAP presenti nel programma;
  - **stopSystem**, arresta l'esecuzione del thread associato e chiude la connessione col database.

  Altri metodi, di particolare interesse, della classe sono:
  - **setCounter**, all'avvio del programma, interroga il database restituendo l'id più grande presente e ne assegna il valore, incrementato di 1, all'attributo counter;
  - **updateDatabase**, aggiorna il database inserendo nuove allerte oppure modificando quelle già presenti;
  - **checkConnectionDB**, verifica che la connessione con il database sia ancora attiva e, nel caso non lo fosse, la rinnova.

  La classe utilizza un thread, la cui esecuzione avviene ogni quattro ore, che si occupa principalmente di richiedere alle varie AppPrevisioni le nuove allerte e aggiornare il database.

- **ConnettoreDB**

  Contenuta nel package *protezionecivile*, la classe **ConnettoreDB** contiene gli attributi e i metodi necessari per stabilire la connessione al database ed eseguire operazioni sullo stesso.

  L'unico attributo della classe è **connection**, che contiene le informazioni relative alla sessione tra l'applicativo e il database;

  La classe estende l'interfaccia AccessoDB, implementandone i metodi. Per la maggior parte, le loro funzioni sono le medesime dei metodi già descritti per la classe **SistemaCentrale**.

  Nel momento in cui viene creato un oggetto di questa classe, viene invocato il metodo **openConnection**, che si occupa di stabilire la connessione con il database.

  Una volta che la connessione col database è stabilita, viene invocato il metodo **initialiseDB**, che verifica la presenza all'interno del database della tabella "alert" e, nel caso non ci sia, procede con la sua creazione.

- **App Mobile**

  Contenuta nel package *user*, la classe **AppMobile** contiene gli attributi e i metodi necessari all'utente per poter effettuare le diverse operazioni di ricerca nel database ed ottenere le notifiche per il proprio CAP.

  Gli attributi della classe sono:
  - **cap**, indica il cap dell'utente;
  - **stubSisCentrale**, è lo stub RMI alla quale l'AppMobile si connette per effettuare le operazioni;
  - **noticeList**, array per memorizzare la lista delle notifiche ricevute dal sistema centrale.

  La classe implementa i seguenti metodi:
  - **checkInCapList,** metodo che richiama, attraverso RMI del sistema centrale, il metodo checkInCapList al fine di controllare se un dato CAP è presente nella lista dei CAP del sistema centrale;
  - **getCapList,** metodo che richiama, attraverso RMI del sistema centrale, il metodo getCapList al fine di ottenere la lista dei CAP del sistema centrale;
  - **getNotice,** metodo che richiama, attraverso RMI del sistema centrale, il metodo askNotice al fine di ottenere le notifiche per il proprio CAP;
  - **getDefaultAlert,** metodo che richiama, attraverso RMI del sistema centrale, il metodo getDefaultAlert al fine di ottenere le allerte di default;
  - **getAllAlert,** metodo che richiama, attraverso RMI del sistema centrale, il metodo listAlarm al fine di ottenere tutte le allerte presente nel database;
  - **getSeriousAlarm,** metodo che richiama, attraverso RMI del sistema centrale, il metodo listSeriousAlarm al fine di ottenere la lista di allarmi seri;
  - **getListAlarm**, metodo che richiama, attraverso RMI del sistema centrale, il metodo listAlarm al fine di ottenere la lista di allarmi filtrata per CAP, fascia oraria, tipo, data;
  - **stopAppMobile**, arresta l'esecuzione del thread associato.

  La classe utilizza un thread, grazie all'aiuto di un executorService, la cui esecuzione avviene ogni quattro ore, che si occupa principalmente di richiedere al sistema centrale le nuove notifiche.

  Osservazione:
  Abbiamo deciso di implementare altre due classi relativa all'AppMobile: l'AppMobileCreator e l'AppMobileWindow. La prima ci consente di andare a creare delle nuove AppMobileWindow, la seconda invece è la classe che l'utente finale andrà ad utilizzare. La nostra scelta è giustificata dal fatto che utilizzare un'interfaccia grafica rispetto ad una testuale è molto più semplice.
  Da notare che abbiamo mantenuto la logica funzionale separata dall'interfaccia utente. Infatti, la logica funzionale è tutta all'interno dell'AppMobile e l'AppMobileWindow si occupa solo della visualizzazione, richiamando i metodi dell'AppMobile. Così facendo è possibile andare a sostituire in un qualsiasi momento la parte di visualizzazione senza andare ad intaccare la parte funzionale.