**“Software Engineering”**

**Course**

**a.a. 2018-2019**

**Template version 1.0**

**Deliverable #3**

**Lecturer: Prof. Henry Muccini (henry.muccini@univaq.it)**

**Dashboard Monitoraggio Ambientale**

|  |  |
| --- | --- |
| **Date** | 04/02/2018 |
| **Deliverable** | Documento Finale – D3 |
| **Team (Name)** | 404 Not Found |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| D’Ercole Dario | *244615* | [dario.dercole@student.univaq.it](mailto:dario.dercole@student.univaq.it) |
| Carestia Alessandro | *244236* | [alessandro.carestia@student.univaq.it](mailto:alessandro.carestia@student.univaq.it) |
| Pennacchia Francesco | *247848* | [francesco.pennacchia@student.univaq.it](mailto:francesco.pennacchia@student.univaq.it) |
| Avallone Andrea | *179324* | [andrea.avallone@student.univaq.it](mailto:andrea.avallone@student.univaq.it) |

Table of Contents of this deliverable

List of Challenging/Risky Requirements or Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **Challenging Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** |
| Formato memorizzazione del dato; | 14/11/2018 | 29/01/2019 | Inizialmente si era pensato all’utilizzo di una scala di valori, scala con la quale il dato di ogni singolo sensore veniva elaborato per poter essere salvato in un formato più leggero. Approfondendo lo studio sulle problematiche di salvataggio dei dati si è deciso di utilizzare il dato nella maniera più simile alla realtà, questo per evitare la perdita di informazioni dovute ad una eventuale conversione ad una metrica diversa |
| Gestione comportamento dei sensori (in caso di dati anomali); | 20/11/2018 | 30/11/2018  (modificato il 29/01/2019) | Prevediamo che il Sistema potrà fornire un feedback al sensore sul suo stato così da regolarne la frequenza di invio dei dati |
| visualizzazione del giusto livello di dettaglio da parte di un gestore | 30/11/2018 | 03/12/2018 (modificato il 29/01/2019) | È giusto che un gestore veda i dati di tutto il sistema e quindi di ogni singolo sensore? La risposta che ci siamo dati a seguito di diverse discussioni è no. Ha senso che veda solo se nel suo livello inferiore vi sono anomalie. Quindi viene implementato un sistema di segnali di allerta che comunicheranno ad ogni gestore la presenza di anomalie e/o problematiche. Di seguito verranno fornite ulteriori informazioni sul come viene gestito il livello di dettaglio. |
| elaborazione del dato in base al livello di gestione | 30/11/2018 | 03/12/2018 (modificato il 29/01/2019) | Il dato lo dobbiamo aggregare per poterlo vedere anche nei livelli superiori oppure abbiamo bisogno di un semplice alert di eventuali anomalie? L’utilizzo di un campo alert permette di alleggerire molto il compito del client rispetto alla soluzione descritta nella prima documentazione. |
| Rischio tutti i sensori aumentano la loro frequenza di invio | 30/11/2018 |  | Questo è il rischio principale che il nostro sistema si potrebbe trovare ad affrontare. Si può verificare solo nel caso in cui tutti i sensori stiano rilevando dati al di sopra della soglia stabilità, quindi a tutti i sensori viene richiesta una maggiore frequenza di inoltro del dato.  Essendo però questa una problematica limite si è deciso di non approfondire la ricerca di soluzioni in questo momento. |

A. Requirements Collection

*In this section, you should describe both the application* ***features/functional*** *requirements as well as the* ***non functional*** *ones. You shall also document* ***constraints*** *and* **rules***, if they apply.*

A.0 Detailed Scenarios

**Scenari:**

* **Rilevazione di anomalie:**

Il sistema analizza i dati relativi ad un segnale e ne controlla tipo di parametro, relativo valore e lo stato di funzionamento del sensore:

1. Se non risulta nessuna anomalia, allora il sistema conclude l’analisi senza eseguire altre azioni;
2. Se il sensore riporta uno stato anomalo nel suo funzionamento e/o un valore ambientale anomalo, viene inoltrata una query al database che aggiornerà il valore di controllo. Questo alert verrà poi propagato nei record del database relativi alle zone che circoscrivono l’area contenente il sensore (quindi il record relativo all’edificio e al distretto).

* **Rilevazione dei pericoli:**

Il sistema scansiona ad intervalli regolari i record relativi agli ultimi segnali ricevuti dai sensori di una stessa area. Se almeno uno dei sensori riporta un valore ambientale anomalo, viene calcolato lo stato di pericolo per quell’area utilizzando un algoritmo che tiene conto della severità delle singole anomalie e della combinazione di tipi di parametro fuori soglia.

Se l’algoritmo riporta un possibile pericolo, ad esempio un incendio, viene inoltrata una query al database che aggiornerà il valore di controllo (“alert”) ad un valore che corrisponde al tipo di pericolo nell’area corrispondente. Questo alert verrà poi propagato nei record del database relativi alle zone che circoscrivono l’area interessata (quindi il record relativo all’edificio e al distretto).

* **Visualizzazione dashboard e modifica livello di dettaglio:**

Il gestore si autentica sul sistema e viene rediretto alla dashboard, che caricherà i primi dati relativi alla zona di sua giurisdizione. Ogni minuto vengono richiesti al server dati aggiornati e il gestore riceverà eventuali notifiche di pericoli individuati nella sua zona in tempo reale, senza attendere la prossima richiesta di dati.

In ogni momento, il gestore potrà selezionare una delle macroaree presenti nella sua zona per caricare i dati relativi a quella macroarea:

* Nel caso di un gestore di città, potrà selezionare uno dei distretti così da poter monitorare gli edifici presenti in quel distretto;
* un gestore di distretto potrà selezionare uno degli edifici presenti, così da monitorare la situazione nelle singole aree dell’edificio;
* un gestore di edificio potrà selezionare una delle aree (stanze, corridoi, atri…), così da monitorare la situazione in quell’area: questo è l’unico caso in cui si avrà accesso ai dati dei singoli sensori.

Da qui avrà l’opzione di risalire al suo livello iniziale. In ogni caso, non si potrà mai scendere di oltre un livello.

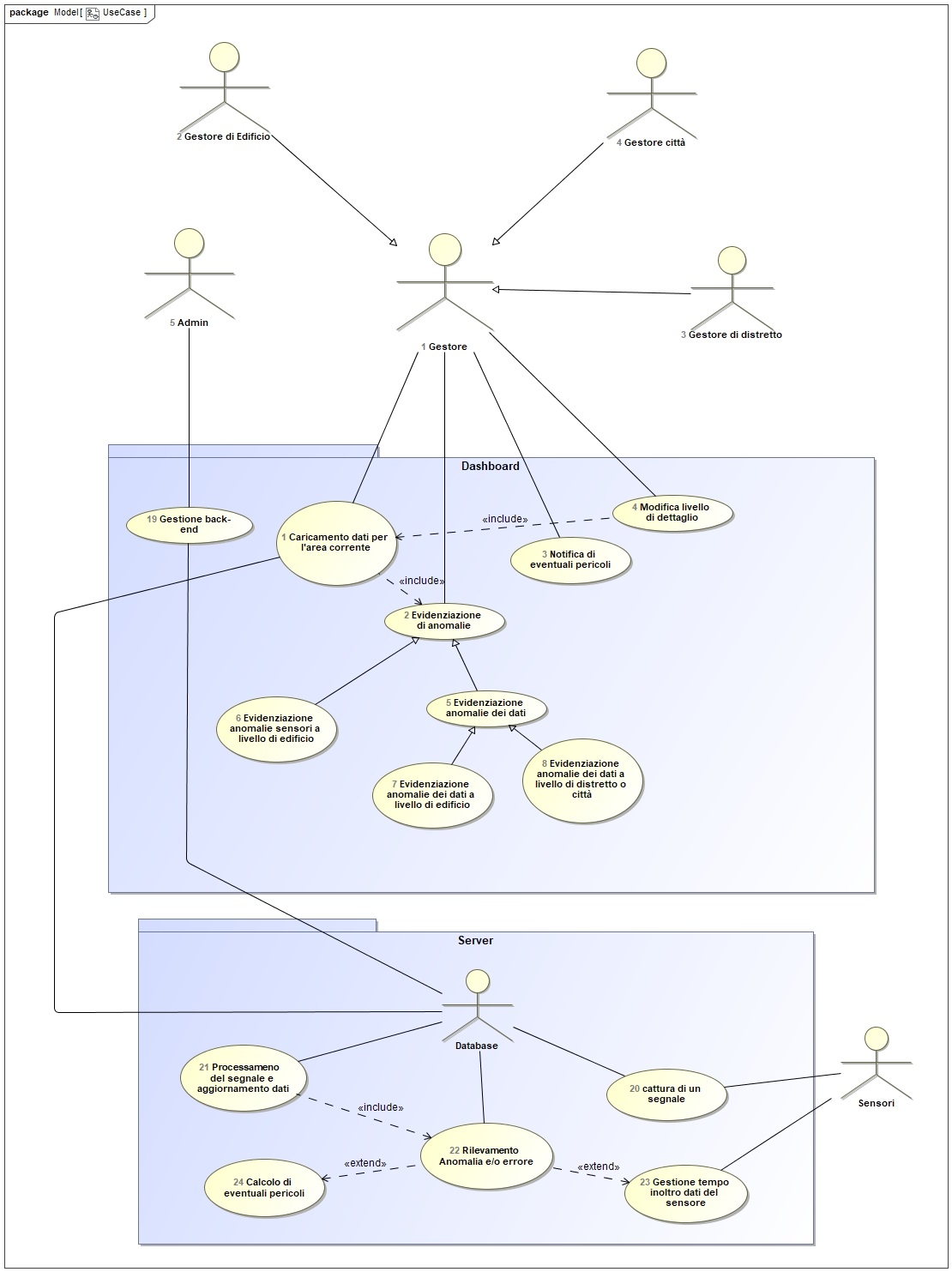
A.1 Functional Requirements

*<List, in bullet points, all the functional requirements your system shall implement>   
<Describe functional requirements and stakeholders through Use Case Diagrams>. <Then, prioritize them, and provide a table-based description of the most important requirements>*

*<Provide a table-based description of the most important requirements, using the Alistair Cockburn Use Case Template>*

1. I sensori devono poter inviare i segnali al server, dove verranno processati;
2. Il server deve poter tradurre il contenuto di un segnale in dati trascrivibili sul database;
3. Il server deve poter memorizzare i dati processati sul database;
4. Il server deve saper rilevare se un sensore sia rimasto silente troppo a lungo;
5. Il server deve poter modificare la frequenza di invio del dato rilevato da un sensore nel caso in cui l’ultimo segnale abbia riportato un valore fuori soglia;
6. Il server deve essere in grado di rilevare potenziali pericoli a partire da una combinazione di anomalie;
7. Il server deve poter notificare a tutti i gestori al momento connessi riguardo potenziali pericoli rilevati;
8. Il server deve fornire l’accesso in back-end agli amministratori di sistema;
9. I gestori devono potersi autenticare all’apertura delle loro dashboard;
10. I gestori devono poter monitorare sulle loro dashboard le zone (intera città, distretto o edificio) di loro competenza;
11. I gestori devono poter scendere di un livello nella gerarchia di zone, ovvero il gestore di città deve poter selezionare un distretto così da monitorare lo stato dei singoli edifici, e così via per i gestori di livello inferiore;
12. Il gestore a livello di edificio deve essere notificato riguardo tutti sensori malfunzionanti, tramite feedback visivo;
13. La dashboard dovrà segnalare al gestore le zone (distretto, edificio o area) in cui sono state rilevati molteplici anomalie, specialmente nei casi in cui il server abbia rilevato un potenziale pericolo.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requisito funzionale** | **Priorità** | **Use Case** | **Sequence Diagram** |
| 4. Il server deve saper rilevare se un sensore sia rimasto silente troppo a lungo; | Media | 22 | 22 |
| 5. Il server deve poter modificare la frequenza di invio del dato rilevato da un sensore nel caso in cui l’ultimo segnale abbia riportato un valore fuori soglia; | Alta | 23 | 23 |
| 6. Il server deve essere in grado di rilevare potenziali pericoli a partire da una combinazione di anomalie; | Alta | 24 | 24 |
| 7.  Il server deve poter notificare a tutti i gestori al momento connessi riguardo potenziali pericoli rilevati; | Alta | 2-3 | 2-3 |
| 10. I gestori devono poter monitorare sulle loro dashboard le zone (intera città, distretto o edificio) di loro competenza; | Media | 1-4 | 1-4 |
| 12. Il gestore a livello di edificio deve essere notificato riguardo tutti sensori malfunzionanti, tramite feedback visivo; | Alta | 6 | 6 |
| 13. La dashboard dovrà segnalare al gestore le zone (distretto, edificio o area) in cui sono state rilevati molteplici anomalie, specialmente nei casi in cui il server abbia rilevato un potenziale pericolo. | Media | 2-3-4 | 2-3-4 |



**Figura 1.**  Use case diagram v3- non si è riscontrata la necessità di apportare modifiche rispetto alla versione 2

Questa è la versione definitiva dell’Use case, rispetto alla prima versione dell’Use Case sono state apportate diverse modifiche, in modo particolare lato server, che inizialmente erano state trascurate. Nella parte riguardante il server sono state effettuate delle importanti aggiunte, inerenti il controllo e l’aggiornamento dei dati; per esempio i casi d’uso 22, 23 e 24 che riguardano il controllo dei dati, oppure il caso d’uso 21 che riguarda l’aggiornamento dei dati.

È stata presa la decisione di inserire i controlli sui dati lato server, per evitare di appesantire il lato client e quindi lasciare tutto il lato computazionale al server.

Per quanto riguarda la Dashboard, si occupa di far ricevere ai gestori i dati di interesse, con eventuali evidenziazioni di anomalie, sul sensore o sul dato e le eventuali notifiche di pericolo. Tutto questo viene gestito dai seguenti casi d’uso, l’1 richiede i dati per l’area di interesse del gestore, il 4 si occupa del livello di dettaglio al quale il gestore può accedere. Per quanto riguarda il livello di dettaglio è stato deciso di inserire questo caso d’uso in quanto non è necessario che i livelli più alti accedono a informazioni non rilevanti per loro, quindi ad ogni gestore vengono riportate le informazioni del livello di appartenenza. I casi d’uso 2,4,5,6,7 e 8 si occupano dell’evidenziare le anomalie sulla Dashboard in base alla gravità.

Per quanto riguarda la gestione Back-End del caso d’uso numero 19 non abbiamo dato una priorità alta, quindi abbiamo deciso di non sviluppare ulteriormente questo caso d’uso.

**Diseguito vengono riportate le descrizioni tramite tabelle di Cockburn dei principali use case:**

|  |  |  |
| --- | --- | --- |
| USE CASE | #1: Caricamento dati per l'area corrente. | |
| Goal in Context | Caricare i dati dei sensori dal server e aggiornare quelli già presenti nella dashboard. | |
| Scope & Level | Interfaccia Gestore - Primary task. | |
| Preconditions | Il gestore sta visualizzando la dashboard | |
| Success End Condition | Il dato nel server relativo al sensore è succesivo (nel tempo) a quello nella dashboard, e viene aggiornato. | |
| Failed End Condition | Mancato aggiornamento dei dati. | |
| Primary,  Secondary Actors | Database  Gestore (generico) | |
| Trigger | Il timer d’aggiornamento dei dati scade | |
| DESCRIPTION | Step | Action | |
|  | 1 | Vengono richiesti i dati per l’area visualizzata al server. | |
|  | 2 | Se siamo a livello di edificio, controlliamo che i dati ricevuti da ogni sensore siano aggiornati:  Se si, il dato sostuisce il suo corrispondente nella dashboard  Altrimenti il dato non viene rimpiazzato ed inoltre se il dato ricevuto dal server risale ad oltre 3 minuti fa, segnala un’anomalia nel sensore responsabile (UC #6). | |
|  | 3 | La dashboard avvia il controllo sulle anomalie dei dati (UC #5). | |
| EXTENSIONS | Step | Branching Action | |
|  | 2a | Se siamo a livello di distretto o città:  A questo livello il sistema si limita a controllare nella risposta del server quali sono le sotto-aree che riportano anomalie nei dati (UC #8) | |

|  |  |  |  |
| --- | --- | --- | --- |
| RELATED INFORMATION | | Caricamento dati per l'area corrente. | |
| Priority: | | Alta. | |
| Frequency | | Al più 50 volte al minuto. | |
| Channels to actors | | GUI | |
| USE CASE # | #2: Evidenziazione Anomalie | | |
| Goal in Context | Riporta in dashboard informazioni visive sulle anomalie | | |
| Scope & Level | Modulo anomalie del client - Primary Task | | |
| Preconditions | Si sta visualizzando la dashboard | | |
| Success End Condition | A livello di edificio:  I dati sopra (o sotto) la soglia vengono evidenziati  Le aree dell’edificio con multiple anomalie nei dati saranno marcate in modo molto evidente  Le aree dell’edificio in cui sono presenti sensori difettosi saranno contrassegnate più o meno marcatamente in base al numero di sensori rimanenti rispetto ogni parametro monitorato.  A livello di distretto e città:  Gli edifici e i distretti in cui sono presenti anomalie nei dati verranno evidenziati con colore che varia in base alla molteplicità delle anomalie rilevate nei livelli inferiori (con gradienti che vanno dal verde al rosso, in ordine crescente di anomalie). | | |
| Failed End Condition | Le anomalie rilevate potrebbero non essere tutte evidenziate correttamente, fino alla prossima attivazione di questo UC. | | |
| Primary,  Secondary Actors | Gestore (generico)  Database | | |
| Trigger | Ricezione nuovi dati | | |
| DESCRIPTION | Step | | Action |
|  | 1 | | Il modulo anomalie controlla nella risposta del server se sono presenti dati riguardanti anomalie |
|  | 2 | | Se presenti, legge le informazioni e modifica i campi della dashboard per riflettere il tipo e la severità dell’anomalia |

|  |  |
| --- | --- |
| RELATED INFORMATION | Evidenziazione Anomalie |
| Priority: | Media priorità |
| Frequency | Fino a 50 / minuto |
| Channels to actors | GUI |

|  |  |  |
| --- | --- | --- |
| USE CASE # | #3: Notifica di eventuali pericoli | |
| Goal in Context | Dare un’allarme al gestore quando si riceve un broadcast di pericolo dal server che riguarda l’area di interesse | |
| Scope & Level | Modulo di notifica dei pericoli (Primary Task) | |
| Preconditions | Il gestore sta visualizzando i dati sulla sua dashboard | |
| Success End Condition | Il gestore viene notificato di un possibile pericolo | |
| Failed End Condition | Il pericolo non viene notificato oppure viene notificato un allarme errato | |
| Primary,  Secondary Actors | Server  Gestore | |
| Trigger | La dashboard riceve una notifica di pericolo in broadcast dal server | |
| DESCRIPTION | Step | Action |
|  | 1 | La dashboad controlla le informazioni contenute nella notifica |
|  | 2 | Se la notifica riguarda la zona attualmente monitorata oppure una una sotto-zona, il sistema procede nell’identificazione del pericolo |
|  | 3 | Una volta appurato il tipo di pericolo e la zona o sotto-zona interessata, il sistema farà apparire in dashboard un’allarme che punta alla entry relativa alla zona in cui è stato rilevato il pericolo. |

|  |  |
| --- | --- |
| RELATED INFORMATION | Notifica di eventuali pericoli |
| Priority: | Alta |
| Frequency | Caso peggiore: 750 < n < 1500 al minuto |
| Channels to actors | GUI |

|  |  |  |
| --- | --- | --- |
| USE CASE # | #4: Modifica livello di dettaglio | |
| Goal in Context | Possibilità di accedere ad un livello più dettagliato di informazioni. In base alla tipologia di gestore. | |
| Scope & Level | Dashboard gestore – (primary task) | |
| Preconditions | Il gestore sta visualizzando la sua Dashboard. | |
| Success End Condition | Vengono visualizzate le informazioni relative all’area selezionata. | |
| Failed End Condition | Viene visualizzato un messaggio di errore e il gestore continua a visualizzare i dati dell’area corrente. | |
| Primary,  Secondary Actors | Gestore, Admin (?)  Database | |
| Trigger | Il gestore selezione una voce dalla lista di aree che sono di sua competenza. | |
| DESCRIPTION | Step | Action |
|  | 1 | Il gestore seleziona l’area di interesse (Città, Distretto, Edificio). |
|  | 2 | Il sistema recupera i dati relativi alle sotto-aree dell’area selezionata. |
|  | 3 | Il sistema visualizza i dati sulla Dashboard. |
| EXTENSIONS | Step | Branching Action |
|  | 1a | Se il gestore è già sceso di un livello:  La dashboard non permetterà di fare una selezione |

|  |  |
| --- | --- |
| RELATED INFORMATION | Modifica livello di dettaglio. |
| Priority: | Medio |
| Frequency | Variabile (l’attivazione è interattiva, non automatica) |
| Channels to actors | Interattivo |

|  |  |  |
| --- | --- | --- |
| USE CASE # | #6: Evidenziazione anomalie nei sensori a livello di edificio | |
| Goal in Context | Evidenziare il/i dato/i proveniente/i da un sensore che presenta problemi | |
| Scope & Level | Modulo di calcolo delle anomalie | |
| Preconditions | Il gestore sta visualizzando i dati relativi ad una certa area | |
| Success End Condition | I dati relativi a sensori che riportano problemi vengono evidenziati in modo distinto. | |
| Failed End Condition | I dati relativi a sensori che riportano problemi potrebbero non essere tutti evidenziati fino al prossimo avvio di questo caso d’uso | |
| Primary,  Secondary Actors | Database  Gestore di edificio, Gestore di distretto (dopo essere sceso di un livello) | |
| Trigger | Completato il caricamento dei dati in dashboard | |
| DESCRIPTION | Step | Action |
|  | 1 | Se si sta visualizzando la dashboard relativa ad una singola stanza o area dell’edificio:  Il Sistema recupera dalla risposta del server i dati relativi al funzionamento dei sensori presenti nella stanza attualmente monitorata nonché il timestamp dell’ultimo segnale ricevuto |
|  | 2 | Il sistema controlla quanti sensori per ogni tipo di dato monitorato sono ancora in funzione nella stanza; un sensore non è in funzione se e solo se l’ultimo segnale risale ad oltre 3 minuti fa OPPURE il sensore ha riportato di essere malfunzionante |
|  | 3 | Evidenzia il parametro (temperatura,…) in grigio se c’è almeno un sensore di backup, altrimenti evidenzia in nero. |
| EXTENSIONS | Step | Branching Action |
|  | 1a | Se si sta visualizzando la dashboard relativa ad un intero edificio:  Il Sistema recupera dalla risposta del server i dati relativi al funzionamento dei sensori presenti in tutte le stanze e aree dell’edificio attualmente monitorato nonché il timestamp dell’ultimo segnale ricevuto da ognuno di essi. |
|  | 2a | Il sistema controlla quanti sensori per ogni stanza sono in funzione |
|  | 3a | Evidenzia il box relativo ad una stanza in grigio se almeno un sensore è malfunzionante, evidenzia in nero se almeno il 50% dei sensori nella stanza sono malfunzionanti |

|  |  |
| --- | --- |
| RELATED INFORMATION | Evidenziazione anomalie nei sensori a livello di edificio |
| Priority: | Media priorità |
| Frequency | Almeno 50 / minuto |

|  |  |  |
| --- | --- | --- |
| USE CASE # | #22: Rilevamento anomalia e/o errore. | |
| Goal in Context | In seguito all’aggiornamento dei dati (di un sensore), vengono controllati lo stato di funzionamento del sensore e la variabile ambientale misurata (confronto col massimale). | |
| Scope & Level | Rilevatore anomalia e/o errore ,Primary Task. | |
| Preconditions | Il server ha catturato e processato e il segnale inviato da un sensore, e ne ha aggiornato i dati. | |
| Success End Condition | In base all’esito del controllo, l’alert del sensore e dei livelli superiori (area, edificio e distretto) che lo contengono, vengono opportunamente modificati o lasciati invariati. | |
| Failed End Condition | Azione sbagliata sull’alert del sensore, che si trascina anche ai livelli superiori. | |
| Primary,  Secondary Actors | Server.  Sensore | |
| Trigger | Aggiornamento dati relativi ad un sensore. | |
| DESCRIPTION | Step | Action |
|  | 1 | Vengono aggiornati i dati relativi ad un sensore. |
|  | 2 | Vengono controllati la variabile ambientale misurata e lo stato di funzionamento. |
|  | 3 | Uno dei due valori o entrambi portano alla modifica dell’alert del sensore e ai livelli superiori. |
| EXTENSIONS | Step | Branching Action |
|  | 3a | I valori controllati non sono tali provocare anomalia o errore.  L’alert non viene modificato. |

|  |  |
| --- | --- |
| RELATED INFORMATION | #22: Rilevamento anomalia e/o errore. |
| Priority: | Alta |
| Frequency | Almeno 150.000/minuto (in media) |
| Channels to actors | Database |
| OPEN ISSUES | Range di valori per l’alert. |
| Superordinates | Processamento del segnale e aggiornamento dati. |

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE # | #23: Gestione tempo inoltro dati del sensore | | |
| Goal in Context | Aumentare/Diminuire la frequenza con cui il sensore invia i dati | | |
| Scope & Level | Modulo di calcolo delle anomalie | | |
| Preconditions | Il server deve aver già ricevuto alcuni dati dal sensore, nei quali ha riscontrato delle anomalie. | | |
| Success End Condition | La frequenza di inoltro del dato rilevato dal sensore viene aumentata quando si riscontrano anomalie e diminuita quando l’anomalia non viene più riscontrata. | | |
| Failed End Condition | Non è possibile modificare la frequenza | | |
| Primary,  Secondary Actors | Server  Sensore | | |
| Trigger | Maggiore quantità di dati ricevuti | | |
| DESCRIPTION | Step | | Action |
|  | 1 | | Il sistema ha riscontrato la presenza di anomalie nei dati ricevuti / il sistema riscontra che l’anomalia non è più presente |
|  | 2 | | Il sistema imposta un arco di tempo minore per l’inoltro del dato oppure quando non più presente l’anomalia il sistema ripristina l’arco di tempo ad 1 minuto |
| RELATED INFORMATION | | Gestione tempo inoltro dati del sensore | |
| Priority: | | Media priorità | |
| Frequency | | Al Massimo 150000/ minuto ( caso in cui tutti I sensori hanno un anomalia | |
| Subordinates | | <optional, depending on tools, links to sub.use cases> | |

|  |  |  |
| --- | --- | --- |
| USE CASE # | #24: Calcolo di eventuali Pericoli | |
| Goal in Context | Vengono riscontrate le presenze di eventuali pericoli | |
| Scope & Level | Modulo di calcolo dei pericoli | |
| Preconditions | Il server deve aver già ricevuto alcuni dati dal sensore, nei quali ha riscontrato delle anomalie  Nella stessa stanza sono state riscontrate diverse anomalie da diversi tipi di sensori | |
| Success End Condition | Viene riportato un allarme di tipo grave per presenza di pericoli | |
| Failed End Condition | Non è possibile calcolare il pericolo | |
| Primary,  Secondary Actors | Server | |
| Trigger | Maggiore quantità di dati ricevuti | |
| DESCRIPTION | Step | Action |
|  | 1 | Il sistema ha riscontrato la presenza di anomalie nei dati ricevuti da sensori di vario genere presenti nella stessa area |
|  | 2 | Il sistema, utilizzando le opportune regole di calcolo, determina quale sia il pericolo |
|  | 3 | Il sistema segnala il pericolo a tutti i livelli di gestione |

|  |  |
| --- | --- |
| RELATED INFORMATION | Calcolo di eventuali Pericoli |
| Priority: | Media priorità |
| Frequency | Nel Peggiore dei casi (pericoli in tutte le aree)  750<n <1500 |

A1.1 GUI Requirements (da riempire a partire dalla Versione 2)

La GUI deve:

* Essere intuitiva;
* Permettere che le anomalie siano subito evidenziate una volta effettuato l’accesso;
* Fornire, all’avvio dell’applicazione, una schermata per il login, e notificare un eventuale errore nell’inserimento delle credenziali;
* Mostrare i dettagli della zona di competenza associata al gestore una volta eseguito l’accesso. Un gestore di distretto, per esempio, inizialmente avrà una vista sullo stato degli edifici contenuti;
* Aggiornare automaticamente i dati mostrati ogni minuto (30 secondi se sono presenti pericoli) dopo l’ultimo aggiornamento;
* Dare la possibilità di aggiornare i dati mostrati manualmente (mediante un apposito tasto);
* Mostrare le zone e i sensori mediante rettangoli in cui verrà riportato: il proprio ID ridotto (si ottiene levando l’ID delle zone contenenti l’elemento visualizzato); tipo di pericolo (solo nelle aree); in più solo nei sensori si avranno tipo, stato di funzionamento (on/off) e ultimo valore misurato con unità di misura;
* Evidenziare le anomalie colorando l’interno dei rettangoli con i colori verde se non sono presenti anomalie, rosso se vi sono presenti pericoli ed infine il giallo se vi sono anomalie di lieve entità;
* Mostrare le zone ed i sensori in ordine alfabetico, questo in una situazione di normalità, in caso di presenza di anomalie o pericoli in alto vengono riportate le zone interessate.
* Permettere a ogni gestore, premendo su una zona, di visualizzare i dati in essa contenuta e di poter tornare alla vista precedente mediante un apposito tasto;
* Impedire ad un gestore di ogni livello (ponendo il livello città come il più alto) di scendere oltre un livello di dettaglio rispetto a quello iniziale;
* Segnalare eventuali pericoli con segnali acustici e/o visivi;

A1.2 Business Logic Requirements (da riempire a partire dalla Versione 2)

Il sistema dovrà:

* Per ogni segnale ricevuto da un sensore, ricavare ID, variabile ambientale misurata e stato di funzionamento;
* Aggiornare i dati di un sensore con i valori ricavati dal segnale;
* Calcolare la presenza di anomalie di sensore ad ogni singola ricezione di segnale;
* Verificare la presenza di anomalie o pericoli in una zona in funzione dello stato degli elementi in essa contenuti;
* Ogni 3 minuti fare un controllo su ogni sensore per verificare che l’ultimo segnale sia stato inviato non più di 3 minuti prima;
* Impostare lo stato di funzionamento di un sensore a off se da esso non sono stati ricevuti segnali negli ultimi 3 minuti;
* Per ogni richiesta di aggiornamento dati da parte di una dashboard, fornire solo i dati degli elementi da essa visualizzati;

Questo garantirà la congruenza dei dati, aumenterà la velocità di gestione nel lato client, causerà un lavoro aggiuntivo nel lato server.

A1.3 DB Requirements (da riempire a partire dalla Versione 2)

I requisiti a livello di database sono mutati durante lo studio del sistema che si sta sviluppando, essendo arrivati alla conclusione solo in questo deliverable che avremmo utilizzato un DB relazionale.  
I requisiti del DB sono:

* Struttura intuitiva del modello ER;
* Il dato deve avere una struttura ben definita;
* La presenza di una struttura ben definita permetterà una buona gestione delle query;
* Deve poter essere generale per futuri ampliamenti del sistema;

A.2 Non Functional Requirements

*<List and describe here the* ***most important*** *non functional requirements.>***Requisiti non funzionali:**

* Il sistema dovrà supportare la memorizzazione ed il processamento di un minimo di 150.000 segnali al minuto;
* Il sistema dovrà poter gestire richieste provenienti da al più 50 client;
* Il sistema notificherà che un sensore è andato silente entro 3 minuti dal suo ultimo segnale ricevuto;
* Il sistema notificherà un eventuale pericolo in dashboard entro 1 minuto dalla sua rilevazione;
* Le dashboard dovranno essere utilizzabili senza la necessità di una formazione specifica;
* Il sistema dovrà essere modulare, in modo da permettere future espansioni a costi contenuti;
* Il sistema dovrà utilizzare un protocollo di comunicazione sicuro per la ricezione dei segnali, per renderne difficile la manipolazione da parte di malintenzionati;
* Le dashboard verranno visualizzate su istanze del web browser installato sul dispositivo;
* In caso di blackout, il sistema una volta pienamente in funzione fornirà i dati aggiornati, inclusi le informazioni su quali sensori eventualmente sono ancora senza corrente, entro 4 minuti;

A.3 Excluded Requirements

* *<List and describe here the* ***most important******excluded*** *functional requirements.>***Installazione dei sensori:** se ne occupa un’altra organizzazione, noi dobbiamo solamente assicurarci che i sensori di cui l’ID non sia ancora registrato in DB vengano aggiunti automaticamente alla ricezione del primo segnale a loro corrispondente.
* **Sostituzione / riparazione di sensori guasti o malfunzionanti:** ci è richiesto soltanto di notificare le anomalie sulle dashboard dei gestori.

A.4 Assumptions

*<Briefly document, in this section, the most relevant requirement assumptions/decisions you had to made during your project>*

Questo è l’elenco delle assunzioni a cui fare riferimento per poter leggere il documento:

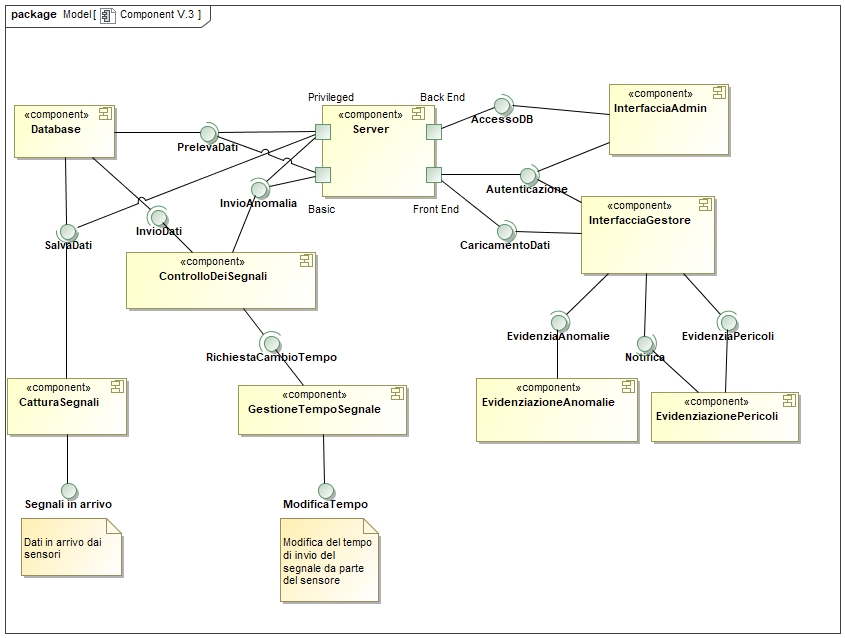
* I sensori avvertiranno di essere guasti per qualsiasi tipo di problema che si presenti, anche se i dati continuano ad essere veritieri.
* I 3 tipi di gestore saranno distinti al momento del login sul sistema e sono così suddivisi: gestore di città, gestore di distretto, gestore di edificio.
* Chi installa i sensori provvederà ad assegnarli gli ID strutturati nel modo seguente: <ID città><ID distretto><ID edificio><ID interno>, in modo da identificare in modo univoco il sensore.
* Il valore che viene fornito dai sensori non verrà approssimato tramite una scala, ma nella sua interezza. Inizialmente si era pensato all’utilizzo di una scala per avere così uno spazio di archiviazione dei dati ridotto. Questo però comportava perdita di informazione e poca precisione nei controlli.
* Il Timestamp dell’ultimo segnale ricevuto viene memorizzato, in modo tale da poterlo confrontare con i nuovi segnali in arrivo.
* Solo il gestore che si trova a livello di edificio può vendere il dato, non abbiamo ritenuto opportuno mostrare il dato nella sua interezza a tutte le tipologie di gestore.
* I sensori non omnidirezionali (come ad esempio i sensori di luminosità) andranno installati tenendo conto degli standard previsti rispetto alla zona in cui appartengono (ad esempio i sensori di luminosità per ambienti esterni devono essere puntati in modo che la luce solare non causi “false” anomalie.).
* Le anomalie vengono evidenziate sulla Dashboard. I colori che evidenziano queste anomalie variano a seconda della gravità dal verde (Nessun anomalia riscontrata) al rosso (Più anomalie riscontrate).
* In caso di pericolo viene inviata una notifica.

***A.5 Prioritization***

<List here all the requirements, in prioritized order>

B. Software Architecture   
*<Report here both the static and the dynamic view of your system design, in terms of a Component Diagram, Class Diagrams and their related Sequence Diagrams >*

C.1The static view of the system: Component Diagram



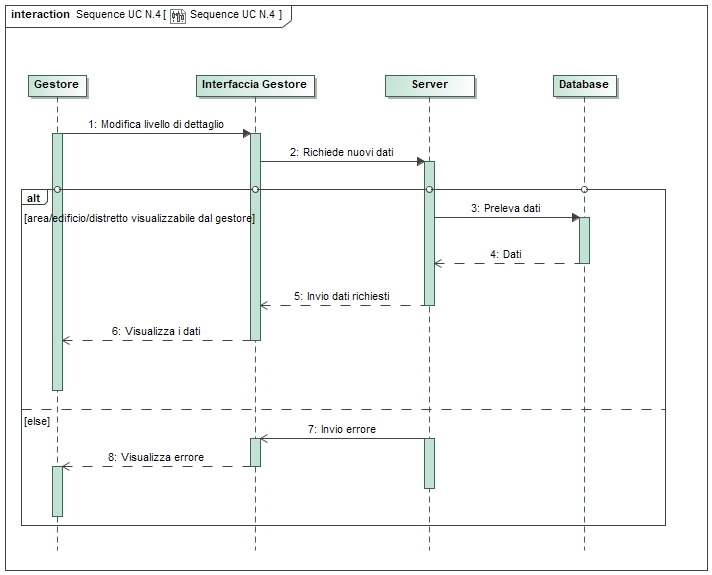
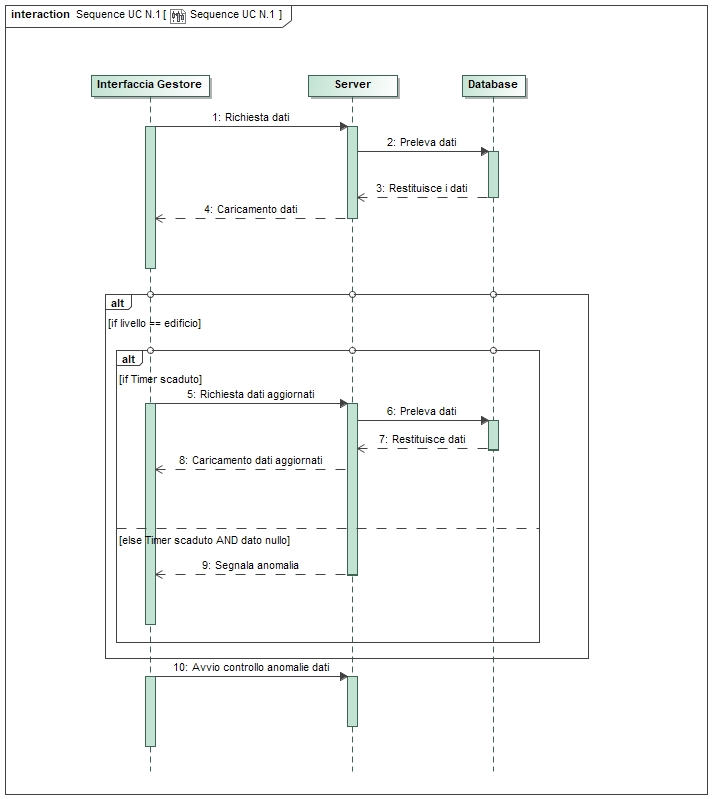
Come si può notare dai vari use case ed il component Diagram sono state apportate molte modifiche rispetto il primo deliverable. Le principali riguardano la gestione del dato direttamente utilizzando funzioni server, così da non fare i vari controlli sui dati in ogni singolo client, alleggerendo così il compito del client, evitanto inoltre il dover permettere la copia dei dati in ogni livello rischiando la Perdita di persistenza dei dati stessi, diminuendo quindi il quantitativo di dati trasmessi e ricevuti dal server.

Nella versione 3 del component l’unica modifica effettuata è stata quella di rinominare il componente “FronteServer” in “Server”, questa scelta è dovuta al fatto che “FronteServer” poteva portare a confusione. Lo scopo di quel componente è quello di separare tutto quello che riguarda il lato server dall’interfaccia gestore e admin, dando così una un’idea più chiara e separata dei componenti. A sinistra del server abbiamo tutti i componenti di cattura e controllo dei segnali gestite dal server, mentre a destra le interfacce gestore e admin.

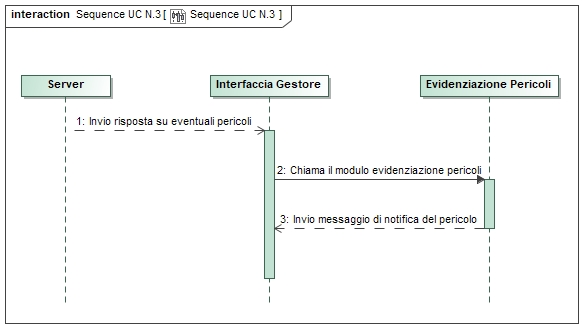
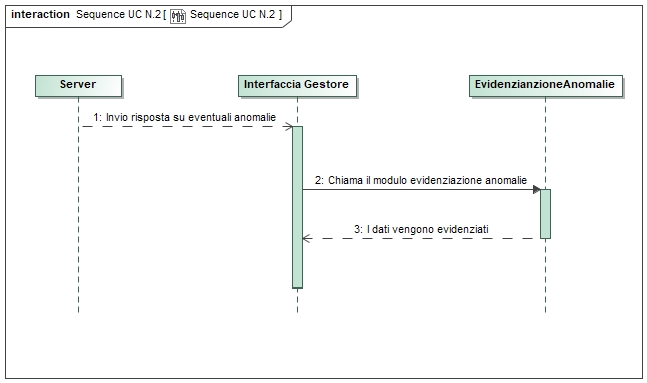
C.2 The dynamic view of the software architecture: Sequence Diagram

In questo documento non verranno descritti tutti gli uses case tramite l’ultilizzo dei sequence diagram, verranno quindi descritti gli uses case che descrivono al meglio i requisiti funzionali a cui è stata data maggiore priorità, cioè quelli descritti nella tabella riportata nel paragrafo A.1.

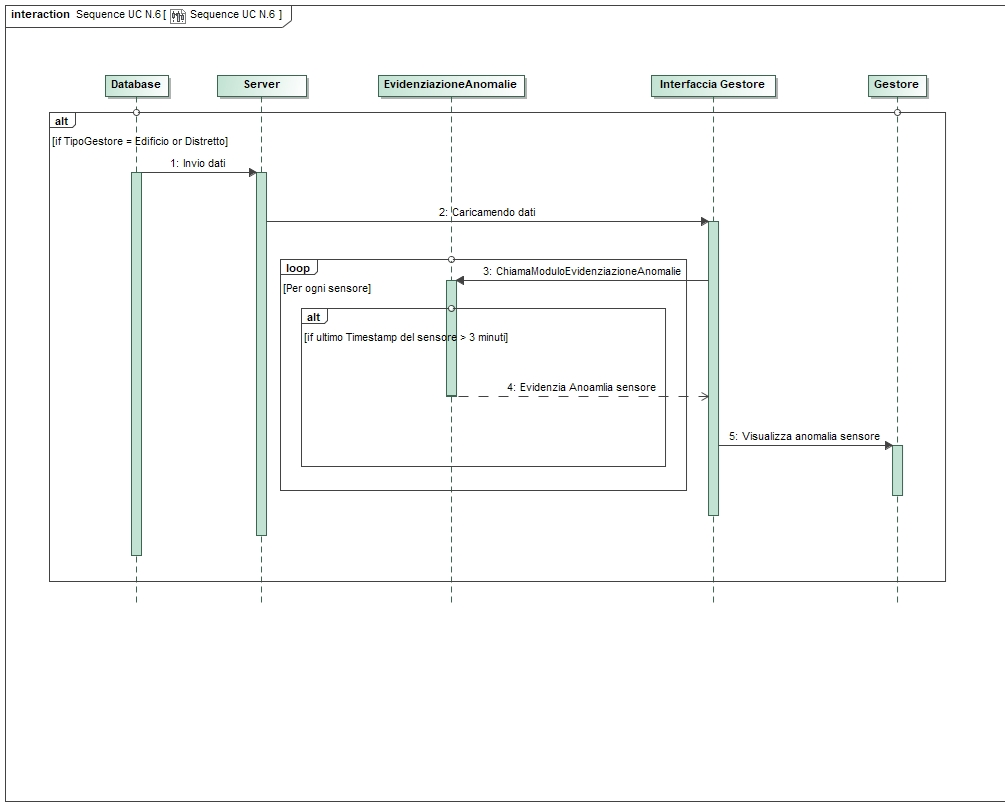
**Requisito funzionale 10** - Di seguito sono riportati i sequence diagram 1 e 4, che descrivono come il sistema risolva il requisito funzionale numero 10



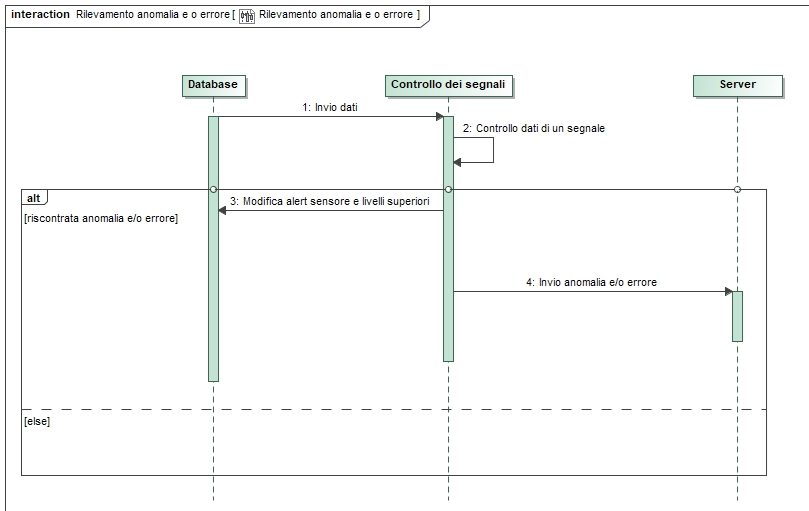
**Requisito funzionale 13 e 7 –** Di seguito i sequence diagram 2-3-4, che descrivono come il sistema implementa il requisito funzionale numero 13. Il sequence diagram 4 non viene riportato per brevità in quanto è stato appena utilizzato per il requisito 10. I seuqence diagram 2 e 3 servono anche per il requisito 7.



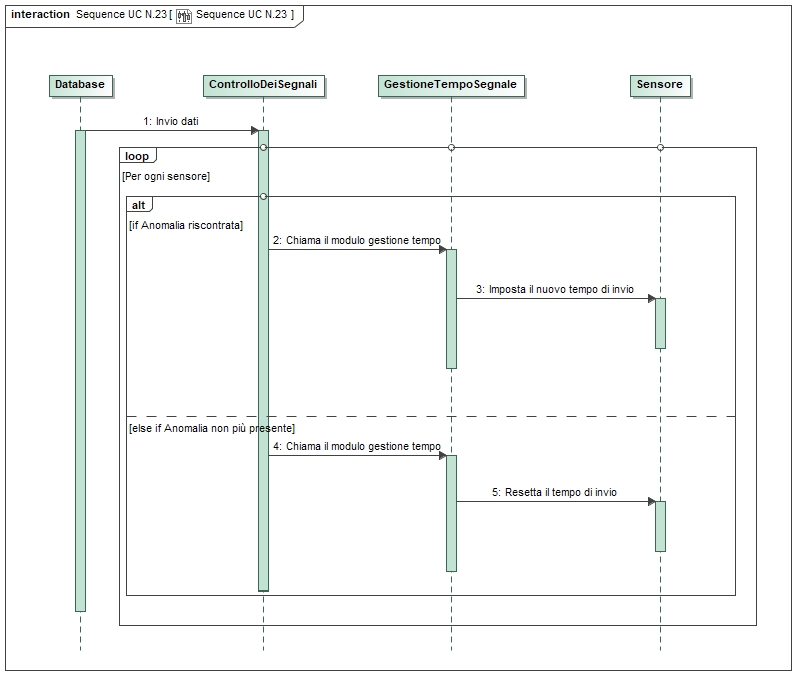
**Requisito funzionale 12 –** Di seguito i sequence diagram 6, che descrivono come il sistema implementa il requisito funzionale numero 12.



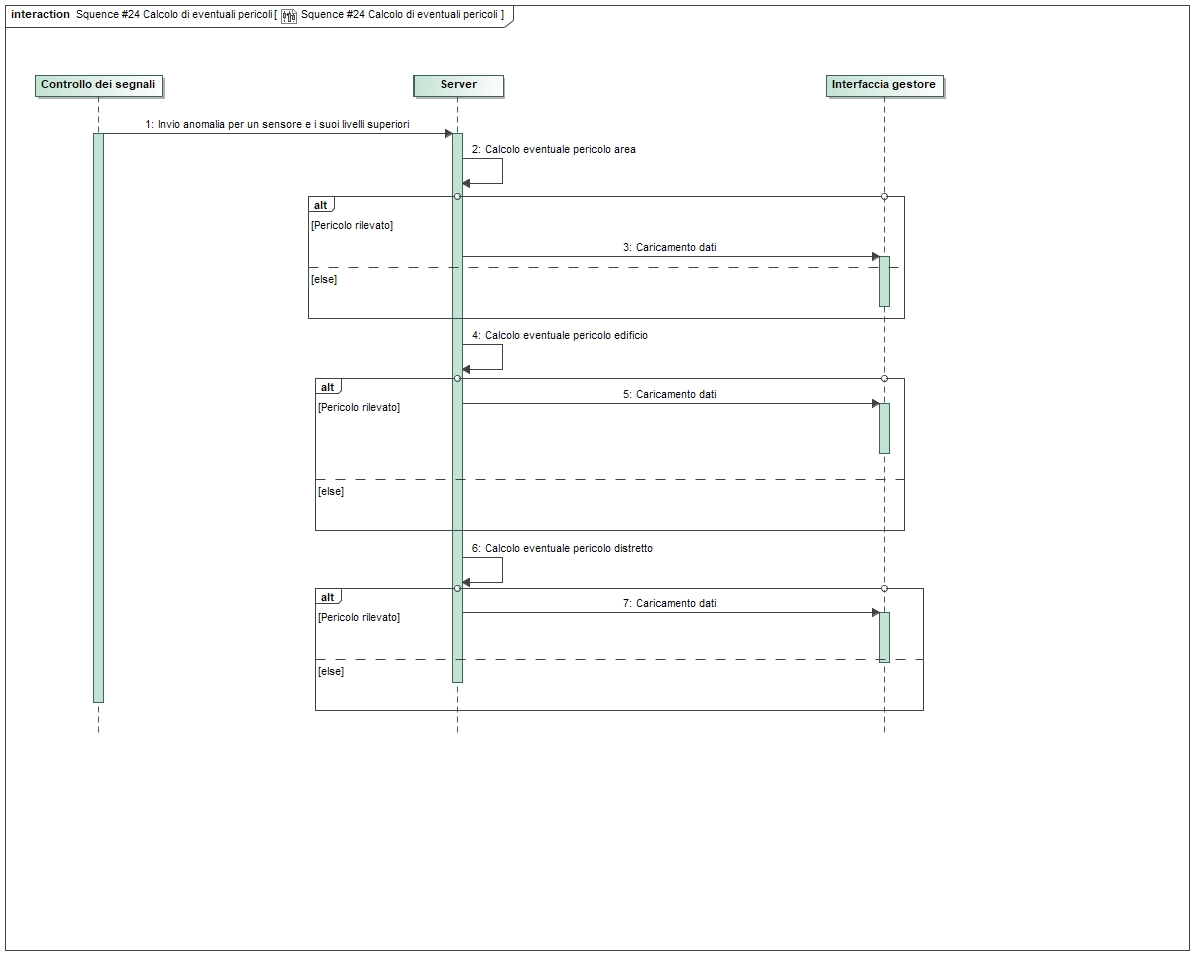
**Requisito funzionale 4 –** Di seguito i sequence diagram 22, che descrivono come il sistema implementa il requisito funzionale numero 4.



**Requisito funzionale 5 –** Di seguito i sequence diagram 23, che descrivono come il sistema implementa il requisito funzionale numero5.



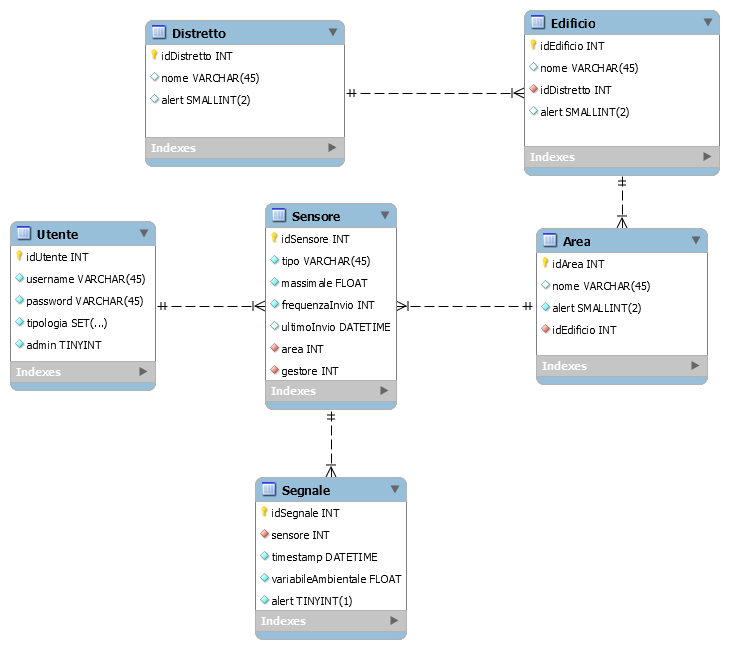
**Requisito funzionale 6 –** Di seguito i sequence diagram 24, che descrivono come il sistema implementa il requisito funzionale numero 6.



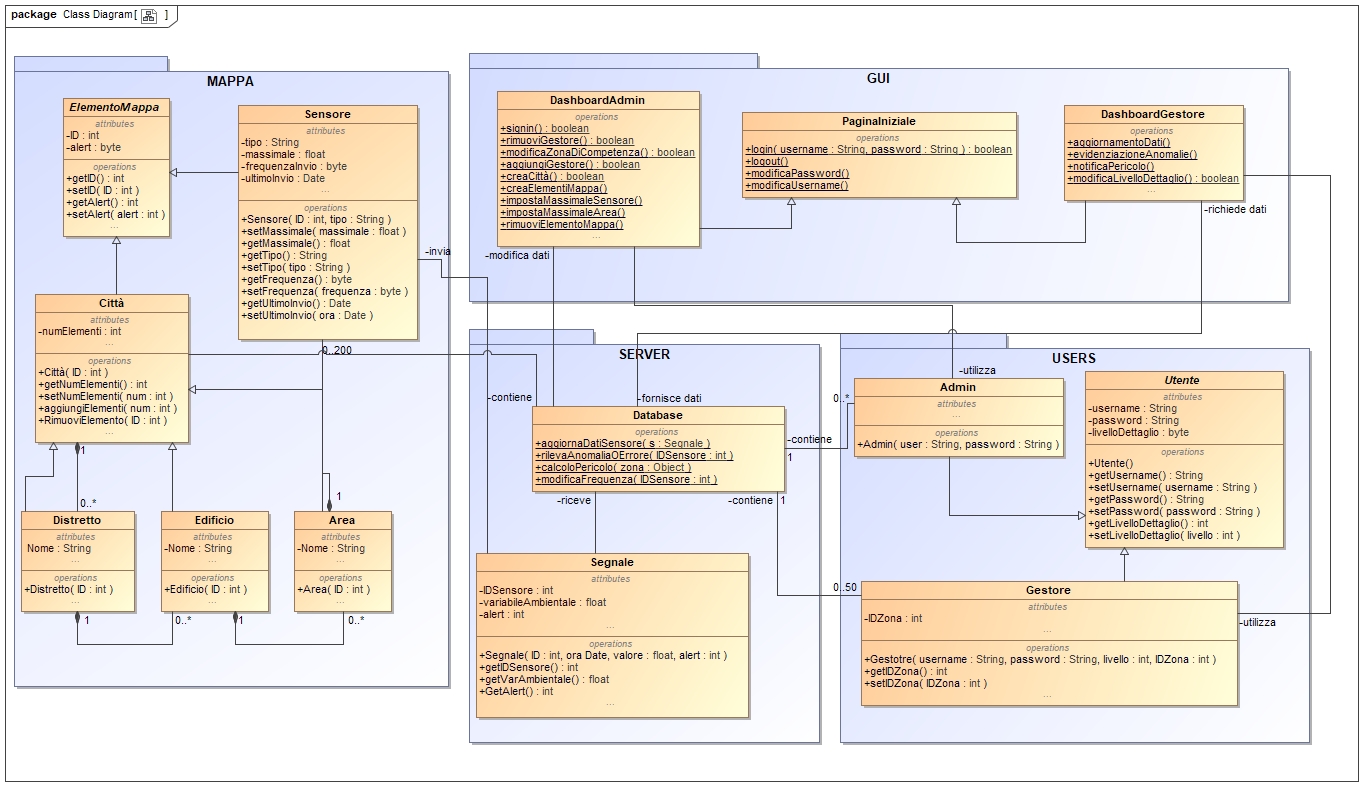
C. ER Design

<Report here the Entity Relationship Diagram of the system DB>

Qui viene riportato il modello ER del nostro DB relazionale.



D. Class Diagram of the implemented System



La prima versione del class diagram(del D2) è stato ricavato mediante un processo di forward engineering sullo use case diagram(per quanto riguarda le classi), i metodi sono stati ricavati maggiormente dal component diagram. La seconda versione è stata ricavata dalle modifiche apportate durante la scrittura del codice.

Questo modello si divide in quattro package:

* **Users**: contiene la classe astratta Utente che generalizza Admin e Gestore. Possiedono gli attributi necessari per l’accesso nel sistema quali username e password, uno che indica il tipo di utente(livelloDettaglio), e uno esclusivo della classe Gestore (IDZona) che indica la zona di competenza (edificio, distretto o città). I metodi forniti sono quelli basilari per restituire e modificare tali attributi, e i costruttori;
* **GUI**: le classi qui presenti non hanno attributi, ma solo metodi statici messi a disposizione degli utenti. PaginaIniziale, usata da tutti i tipi di utenti, permette l’accesso e l’uscita dal sistema e la modifica delle proprie credenziali. DasboardAdmin, utilizzabile esclusivamente dagli utenti admin, mette a disposizione tutto il necessario per la gestione backend (inserimento, cancellazione e modifica di dettagli degli utenti e degli elementi mappati contenuti nel database). DasboardGestore si occupa della visualizzazione dei dati monitorati, la notifica di pericoli e la modifica del livello di dettaglio, ed è utilizzabile solo dagli utenti gestori;
* **Mappa**: contiene la classe astratta ElementoMappa che generalizza Sensore e Città, che a sua volta generalizza Distretto, Edificio e Area. Gli attributi comuni a tutti sono ID per l’identificazione e alert per rappresentare lo stato di allerta (stato di funzionamento nei sensori). Nella classe Sensore vengono aggiunti: tipo (indica l’unità di misura); variabileAmbientale (ultimo valore registrato); massimale (indica il valore soglia); frequenzaInvio (per regolare la frequenza di invio di segnali); ultimoInvio (indica l’ora in cui è stato ricevuto l’ultimo segnale). Nella classe Città si aggiunge il solo numElementi che rappresenta il numero di elementi contenuti. Anche qui, tutti i metodi sono i costruttori e quelli basilari per restituire e modificare tali attributi, e in più ci sono aggiungiElementi e rimuoviElemento della classe Città, che servono a modificare la lista degli elementi contenuti. La classe più grande contiene elementi della classe più piccola, e in questo caso l’ordine di grandezza (crescente) è: Sensore, Area, Edificio, Distretto, Città;
* **Server**: contiene le classi Segnale e Database. Il primo viene generato nel momento in cui il server riceve il segnale inviato da un sensore e inizia il suo processamento. Possiede tre attributi: IDSensore (ID del sensore che ha inviato il segnale), variabileAmbientale (valore misurato), alert (che indica lo stato di funzionamento del sensore); e i soli metodi per restituire i suoi attributi, infatti, i dati del segnale servono solo per aggiornare lo stato del sensore associato, e una volta fatto ciò, vengono eliminati. Database, invece, non ha attributi e definisce i metodi statici per (elenco in ordine cronologico di esecuzione): leggere i dati di un segnale e utilizzarli per aggiornare quelli del sensore inviante (aggiornaDatiSensore); si controlla se i nuovi dati producono anomalie o errori a livello di sensore (RilevaAnomalieOErrore); in funzione del risultato del controllo precedente la frequenza di invio viene modificata (modificaFrequenza); se si trovano anomalie o errori si calcola la presenza di pericoli ai livelli di area, edificio e distretto che contengono il sensore (calcoloPericolo). Una volta compiuta questa serie di azioni, il segnale processato viene eliminato e si passa al successivo.

E. Design Decisions   
<Document here the **5** most important design decisions you had to take. You can use both a textual or a diagrammatic specification.>

1. **Utilizzeremo un server centrale**: si era valutata un’architettura con server locali ed un server centrale, ma si sarebbero create diverse problematiche sull’aggiornamento e la comunicazione dei dati raccolti.   
   Abbiamo quindi optato per la gestione del sistema con un singolo server centrale, questo ci permetterà di evitare problematiche di aggiornamento dei dati. La cosa però più importante è quello di rendere scalabile il sistema.
2. **Realizzeremo una webApp:** si renderà quindi possibile l’accesso da un qualunque dispositivo, verrà implementata utilizzando il framework Ionic. Il client ha quindi la necessità di essere molto leggero, quindi la struttura del sistema e la gestione dei dati deve favorire questa necessità.
3. **Utilizzeremo un Database Relazionale.** Nei precedenti documenti avevamo descritto la decisione di utilizzare una base di dati non relazionale, analizzando in maniera più approfondita la struttura dei dati che abbiamo progettato non si è riscontrata la necessità di dover utilizzare un dabatase non relazionale. Inoltre, in fase di sviluppo prototipo, dove continuavamo a pensare ad un DB non relazionale, abbiamo riscontrato problematiche nell’aggiornamento dei dati, i tempi di aggiornamento erano eccessivi.
4. **Forniremo feedback ai sensori sul loro stato di anomalia (regolazione della frequenza di invio).** Ogni sensore ha un parametro di tempo, questo parametro è normalmente impostato per inviare il dato calcolato minuto per minuto. In caso di dati non congruenti o dati che dimostrano la presenza di problematiche verrà modificata questa frequenza di invio.
5. **Il dato del singolo sensore non verrà visualizzato da tutti i livelli di gestione,** l’unico gestore che potrà visualizzare il dato sarà il gestore di edificio, quindi ai gestori dei livelli superiori (distretto e città) verrà semplicemente comunicato che nel livello immediatamente sottostante vi è un errore/anomalia.Dopo lunghe discussioni si è ragionevolmente arrivati alla conclusione che fornire i dati di tutti i sensori presenti nel sistema a tutti i gestori di qualsiasi livello avrebbe appesantito notevolmente il sistema, creando, inoltre, problemi di usabilità all’utente finale; in più sarebbe stato superfluo far ricevere i dati di un determinato distretto alla dashboard del gestore di un distretto diverso da quello da lui gestito.

F. Explain how the FRs and the NFRs are satisfied by design

*<Report in this section how the architectural and low level design you produced satisfies the FRs and the NFRs>*

G. Effort Recording



***PERT****Make a PERT documenting the tasks and timing you expect to spend on the deliverable. Try to be as precise as possible. Check, after the deliverable deadline, if and how you satisfied (or not) the deadlines.*

***Logging*** *As you are working on the assignment, record what you are doing and how long you spent. As a rule of thumb, you should add a log entry every time you switch tasks. For example, if you do something for two hours straight, that can be one log entry. However, if you do two or three things in half an hour, you must have a log entry for each of them. You do not need to include time for logging, but should include the time spent answering the other parts of this question.*

*For this purpose, please use the* ***LogTemplate.xls*** *file.*

***Categorization*** *When logging the time spent on the project, please create different sub- categories.**Specifically, it is important to clearly distinguish between two main categories: the time spent for “****learning****” (the modeling languages, the tools, etc.) from the time needed for “****doing****” (creating the models, taking the decisions, …). Learning tasks are in fact costs to be paid only once, while doing costs are those that will be repeated through the project.*

*For each category, please define sub-categories. Examples follow. You may add other sub-categories you find useful.*

|  |  |
| --- | --- |
| ***Learning***   * ***Requirements Engineering*** * ***Non functional Requirements*** * ***Use Case Diagrams*** * ***Tool study*** | ***Doing:***   * ***Requirements discovery*** * ***Requirements Modeling (UC diagrams)*** |

***Summary Statistics****Based on the attributes defined above, calculate the summary statistics of the time spent for “learning”, the time spent for “doing”, and the total time.*

***Note: this Deliverable report shall document only the Summary Statistics for the different deliverables (D1, D2, and Final). Detailed information shall be reported in the Excel file.***

***COPY HERE (computed from the spreadsheet): i) the total number of hours spent by the group (that is, hours per task X number of people working on that task), ii) the time spent for LEARNING and for DOING***

Appendix. Prototype   
*<Provide a brief report on your prototype, and especially: information on what you have implemented, how the implementation covers the FR and NFR, how the prototypes demonstrates your project correctness with respect to the FR and NFR. You may add some screenshots to describe what required above. Be ready to show your prototype during the oral examination>*