**“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** | 23/12/2018 |
| **Deliverable** | Deliverable n.3 - Final |
| **Team (Name)** | G-SELL |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| Enrico Monte | *246980* | *enrico.monte@student.univaq.it* |
| Giuseppe Gasbarro | *246886* | *giuseppe.gasbarro@student.univaq.it* |
| Lorenzo De Filippis | *230011* | *lorenzo.defilippis@student.univaq.it* |
| Loreto Cicerone | *244333* | *loreto.cicerone@student.univaq.it* |
| Stefano Florio | *246817* | *stefano.florio@student.univaq.it* |

**Project Guidelines**[do not remove this page]*This page provides the Guidelines to be followed when preparing the report for the Software Engineering course. You have to submit the following information:*

* *This Report*
* *Diagrams (Use Case, Component Diagrams, Sequence Diagrams, Entity Relationships Diagrams)*
* *Effort Recording (Excel file)*

***Important:***

* ***document risky/difficult/complex/highly discussed*** *requirements*
* *document decisions taken by the team*
* ***iterate****: do not spend more than 1-2 full days for each iteration*
* ***prioritize*** *requirements, scenarios, users, etc. etc.*

Project Rules and Evaluation Criteria

***General information:***

* *This homework will cover the 80% of your final grade (20% will come from the oral examination).*
* *The complete and final version of this document shall be* ***not longer than 40 pages*** *(excluding this page and the Appendix).*
* *Groups composed of five students (preferably).*

*I expect the groups to submit their work through GitHub*

***Use the same file to document the various deliverable.   
Document in this file how Deliverable “i+1" improves over Deliverable “i".***

**Project evaluation:**

*Evaluation is not based on “quantity” but on “quality” where quality means:*

* *Completeness of delivered Diagrams*
* *(Semantic and syntactic) Correctness of the delivered Diagrams*
* *Quality of the design decisions taken*
* *Quality of the produced code*

Table of Contents of this deliverable

Sommario

**List of Challenging/Risky Requirements or Tasks………………………………………………………4**

Challenging Task Table………………………………………………………………………………………4

Glossario……………………………………………………………………………………………………………5

**A. Requirements Collection………………………………………………………………………………………6**

A.0 Detailed Scenarios……………………………………………………………………………………….6

A.1 Functional Requirements…………………………………………………………………………….8

A.1.1 GUI Requirements……..…………………………………………………………………………….9

A.1.2 Business Logic Requirements……………..…………………………………………………….9

A.1.3 DB Requirements……………………….…………………………………………………………….10

A.2 Non Functional Requirements…………………………………………………………………..11

A.3 Excluded Requirements…………………………………………………………………………….12

A.4 Assumptions……………………………………………………………………………………………..12

A.5 Prioritization……………………………………………………………………………………………..14

**B. Software Architecture………………………………………………………………………………………..15**

C.1 The static view of the system: Component Diagram………………………………….15

C.2 The dynamic view of the software architecture: Sequence Diagram...………15

**C. ER-Design …………….…………………………………………………………………………………………..17**

**D. Class Diagram of the implemented system………………..…………………….………………..20**

**E. Design Decisions……………………………………..………………..…………………….………………..21**

**F. Explain how the FRs and the NFRs are satisfied by design………………..………………..24**

**G. Effort Recording………………………………………………………………………………………………..27**

PERT……………………………………………………………………………………………………………….27

Summary Statistics…………………………………………………………………………………………27

**H. Appendix Code**…………………………………………………………………………………………………….28

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** |
| Ottimizzare il sistema ffinchè riesca a gestire almeno 150.000 valori rilevati dai sensori al minuto. | 01/01/2019 | 14/01/2019 | Abbiamo creato uno script in java che una volta avviato esegue l’inserimento di tutti i campi nel database, compresi 150.000 sensori. Inoltre per aggiornare il valore rilevato da ogni sensore, abbiamo creato un eseguibile il quale crea un thread per ogni sensore; ogni thread, a seconda del tipo di sensore, calcolerà un nuovo valore rilevato ad intervalli predefiniti. |
| Stilare un algoritmo per il calcolo della priorità delle anomalie preciso e puntuale. | 08/01/2019 | 17/01/2019 | Abbiamo realizzato un algoritmo in java preciso puntuale ed affidabile che agisce sui sensori della stanza interessata eseguendo il calcolo della priorità ogni 10 secondi (tmin), per avere sempre una situazione aggiornata. |
| Scegliere un metodo per implementare un’interfaccia grafica dell’applicazione semplice ed intuitiva. | 09/01/2019 | 11/01/2019 | Abbiamo scelto di utilizzare javaFX in quanto oltre ad avere una grande libreria di funzionalità, utilizza un proprio linguaggio “javaFX script” orientato alla programmazione grafica che rende lo sviluppo di app grafiche agevolato.  Inoltre per creare le interfacce abbiamo usato scene builder che consente di evitare la creazione manuale di un file FXML. |
| Gestire la fault tolerance del database. | 15/01/2019 | 17/01/2019 | Abbiamo creato uno script eseguibile che pianifica ed esegue il dump del database ogni 5 minuti, in modo da non perdere eventuali dati in caso di guasti o interruzioni, e poter eseguire il ripristino del database quando necessario. |
| Risolvere il problema degli accessi paralleli nel Sistema. | 14/12/2018 | 16/01/2019 | Per gestire il problema degli accessi paralleli all’interno del Sistema abbiamo fatto in modo di limitare il tempo in cui le connessioni MySql vivono, aprendole e chiudendole solo quando si interroga il DB invece di lasciarle aperte per tutta la durata dell’applicazione.  Il limite di connessioni massime al database di default in MySql è impostato a 151. Lo abbiamo alzato a 200. Quindi possono avvenire 200 connessioni simultanee.  Immaginando di avere 50 gestori attivi, le richieste al DB non saranno tutte concorrenti quindi il database non avrà problemi. |
| Risolvere il problema delle modifiche sincronizzate | 01/12/2018 | 15/01/2019 | Abbiamo deciso di sfruttare un DB centralizzato online. Il dump del database è stato caricato su un host DB (db4free.net) che permette l’accesso in remoto(protetto da password). In questo modo si avrà il db sempre disponibile e si sfrutterà il server dell’host avendo una situazione sempre aggiornata. |

**Glossario**

* SENSORE = Dispositivo con un codice univoco che misura determinate **variabili ambientali** (temperatura, umidità, pressione, luminosità, fumo) ed invia i valori rilevati al sistema.
* MAX VALUE = **Soglia** **massima** per una variabile ambientale rilevabile da un sensore, in una determinata aerea; al superamento del MAX VALUE viene attivato un segnale di allerta.
* MIN VALUE = **Soglia minima** per una variabile ambientale rilevabile da un sensore, in una determinata aerea; quando il valore scende al di sotto del MIN VALUE viene attivato un segnale di allerta.
* ANOMALIA = Un’anomalia si verifica quando una o più variabili ambientali rilevate da un sensore non rientrano nel **range** [MIN VALUE,MAX VALUE] oppure uno o più sensori non funzionano.
* MALFUNZIONAMENTO = Si verifica quando uno o più sensori non risultano funzionanti
* GESTORE = Monitora una determinata area. Il gestore può essere:
  + **GestoreCittà** (supervisione di zone di una città)
  + **GestoreZona** (supervisione di edifici di una zona)
  + **GestoreEdificio** (supervisione di piani e stanze di un edificio).
* AREA = Area geografica nella quale sono posizionati sensori;

può essere: **Città, Zona, Edificio, Piano, Stanza** (corridoio, atrio, aula, parcheggio, etc..).

* STATO AREA = Valori medi delle variabili ambientali per l’area in questione e relativo livello di allerta.

1. Requirements Collection

A.0 Detailed Scenarios

|  |  |
| --- | --- |
| Scenario Name | Alta\_Temperatura\_Rilevata |
| Partecipating Actors | GestoreCittà: Bob; GestoreZona: Max; GestoreEdificio: John |
| Flow of Events | * Nella stanza C1.10 dell’edificio Coppito 2 il sensore di temperatura presente AXC001 rileva il valore “27°”, che supera di 5 gradi il max value “22°” impostato per quella stanza. * Bob visualizza sulla dashboard che la zona “Coppito”, appartenente alla città ad esso assegnata (L’Aquila), è colorata in rosso. * Bob capisce dal colore rosso che c’è un anomalia/malfunzionamento di alta priorità in quella zona. * Cliccando sulla zona interessata Bob visualizza la media delle variabili ambientali per quella zona. * Max a sua volta visualizza sulla dashboard che l’edificio “Coppito 2”, appartenente alla zona ad esso assegnata (Coppito), è colorata in rosso. * Cliccando sull’edificio interessato Max visualizza la media delle variabili ambientali per quell’edificio. * John monitorando la dashboard si accorge che la stanza C1.10. al piano terra di Coppito2 è colorata in rosso. * John clicca sulla stanza e visualizza una schermata contenente la media di tutti i valori per quella stanza e tutti i sensori contenuti con i relativi valori rilevati. * John può notare che il sensore AXC001 riporta una temperatura di 27° e che presenta la dicitura “anomalia si”, così capisce che la temperatura è troppo alta. Quindi provvede ad accendere i climatizzatori per rinfrescare l’aria. |

|  |  |
| --- | --- |
| Scenario Name | Pericolo\_Incendio |
| Partecipating Actors | GestoreEdificio: John |
| Flow of Events | * Nella stanza A1.6 dell’edificio Coppito 0, il sensore di fumo FM001 ha un rilevamento positivo (presenza di fumo). * John visualizza sulla dashboard che la stanza A1.6 di Coppito 0 (edificio ad esso assegnato), è colorata in viola (situazione di pericolo). * John inoltre visualizza un bottone cliccabile, nominato “visualizza pericoli”, in alto a destra sulla dashboard che lo informa di un pericolo rilevato. * John clicca sull’icona e visualizza una nuova interfaccia filtrata con la stanza in pericolo. (Se più stanze sono in pericolo, visualizzerà l’elenco di queste ultime). * John clicca sulla stanza A1.6 e e visualizza una schermata contenente la media di tutte le variabili ambientali per quella stanza e tutti i sensori contenuti con i relativi valori rilevati. * John quindi nota che il sensore di fumo FM001 ha come parametro “1” (fumo rilevato), e presenta la dicitura “anomalia si”. * John, allarmato, clicca sul pulsante di emergenza in alto a destra e visualizza i numeri telefonici delle autorità. * Nel minor tempo possibile chiama i vigili del fuoco fornendo le informazioni dell’edificio e della stanza in pericolo. |

|  |  |
| --- | --- |
| Scenario Name | Settaggio\_MaxValue\_temperatura |
| Partecipating Actors | GestoreEdificio: John |
| Flow of Events | * L’aula Magna dell’edificio Coppito 0, è un aula importante perchè ospita conferenze di vario tipo.   John sa che è un’aula importante e sa che ospitando molte persone la temperatura sarà più alta.   * John seleziona la stanza “Turing” e clicca sul bottone “Settaggio Valori”. * John modifica il campo relativo al max value della temperatura, che di default è impostato a 22°, inserendo come valore “24” e salva. Tutto ciò per evitare che durante delle conferenze la dashboard notifichi un falso allarme di temperature alta dovuto alla presenza di un numero elevato di persone. |

A.1 Functional Requirements

* I **sensori** devono essere in grado di inviare i dati rilevati al sistema, ad intervalli regolari.
* Il **gestore città** deve essere in grado di monitorare la città ad esso assegnata, visualizzando lo stato di tutte le zone.
* Il **gestore zona** deve essere in grado di monitorare la zona ad esso assegnata, visualizzando lo stato di tutti gli edifici.
* Il **gestore edificio** deve essere in grado di monitorare l’edificio ad esso assegnato, visualizzando lo stato di tutte le stanze e il loro piano; per ogni stanza può visualizzare i singoli sensori, con le relative variabili ambientali rilevate.
* Il **gestore edificio** può modificare i MIN VALUE e MAX VALUE relativi ad una stanza.
* Il **gestore edificio** in caso di pericolo deve essere in grado di allertare le autorità competenti.
* Il **gestore edificio** deve essere in grado di filtrare le stanze visualizzando solo quelle “in pericolo”.
* La **dashboard** deve notificare al gestore anomalie e malfunzionamenti relativi ai sensori, fornendo dei livelli di allerta in base all’entità del problema.
* La **dashboard** deve notificare al gestore eventuali situazioni di pericolo.
* La **dashboard** deve fornire una visualizzazione delle aree diversa in base al tipo di gestore in servizio.
* L’**Admin** deve poter inserire o modificare un gestore, e assegnare determinate aree a determinati gestori.

A1.1 GUI Requirements

* **Visualizzazione stato edificio:** Il gestore edificio deve essere in grado di visualizzare e monitorare, attraverso una dashboard, lo stato dell’edificio ad esso assegnato, visualizzando per ogni stanza appartenente ad esso, il relativo grado di allerta(colore) e il piano.
* **Visualizzazione stato zona:** Il gestore zona deve essere in grado di visualizzare e monitorare, attraverso una dashboard, lo stato della zona ad esso assegnata, visualizzando per ogni edificio appartenente ad essa, il relativo livello di allerta(colore).
* **Visualizzazione stato città:** Il gestore città deve essere in grado di visualizzare e monitorare, attraverso una dashboard, lo stato della città ad esso assegnata, visualizzando per ogni zona appartenente ad essa, il relativo livello di allerta(colore).
* **Visualizzazione sensori**: Il gestore edificio selezionando una stanza deve essere in grado di visualizzare tutti i sensori presenti in quella stanza con: codice, tipo, le relative variabili ambientali rilevate, e il loro stato; inoltre deve poter capire quale sensore è causa di un’anomalia o malfunzionamento.
* **Visualizzazione valori medi**: Il gestore deve essere in grado di visualizzare per ogni area ad esso assegnata i valori medi di tutte le variabili ambientali rilevate(temperatura, pressione, luminosità, umidità).
* **Visualizzazione pericolo**: In caso di situazioni di pericolo il gestore deve essere in grado di visualizzare una schermata contenente solo le stanze “in pericolo” e focalizzarsi su di esse.
* **Presentazione della GUI [(collegamento al ppt)](GUI%20Requirements.pptx)**

A1.2 Business Logic Requirements

* **Invio dati**: Il sistema deve ricevere i dati dai sensori ad intervalli regolari ed essere in grado di elaborarli rilevando anomalie o malfunzionamenti. Questi dati sono composti da: identificativo del sensore, identificativo della stanza nella quale è inserito, stato del sensore, valore ambientale rilevato.
* **Calcolo priorità e pericoli**: Il sistema deve effettuare, ad intervalli di 10 secondi(tmin), il calcolo della priorità di ogni stanza, in base ai dati inviati dai sensori in essa inseriti, e notificare al gestore il risultato in modo ben visibile, soprattutto in caso di rilevamento di pericoli.
* **Gestione sensori**: Il sistema deve permettere al gestore edificio di modificare i MIN VALUE e MAX VALUE relativi ad una stanza, per abbassare o alzare la soglia dei rilevamenti di anomalie.
* **Gestione utenza:** Il sistema deve permettere all’admin di inserire o modificare un gestore, di recuperare la password utente, e assegnare determinate aree a determinati gestori.
* **Allerta sicurezza**: Il sistema, in caso di pericolo, deve far si che il gestore edificio possa, a sua discrezione, allertare le autorità competenti ed evitare incidenti.

A1.3 DB Requirements

* **Salvataggio dati sensori:** Il sistema deve memorizzare, ad intervalli regolari, nel database tutti i dati relativi ad un sensore: identificativo del sensore, identificativo della stanza nella quale è posizionato, segnale di stato, intervallo di invio valore (cambia in base alle anomalie riscontrate).
* **Salvataggio dati valori**: Il sistema deve ricevere e memorizzare un valore per ogni sensore presente nel DB ad intervalli specifici, e salvare il relativo timestamp di ricezione.
* **Database centralizzato**: Per permettere le modifiche sincronizzate e l’accesso in parallelo da parte di più gestori, il database deve essere centralizzato, ovvero memorizzato su un server dove i vari gestori possono accedervi dalle loro macchine.
* **Fault tolerance**: Il database deve permettere funzioni di backup per creare un dump della situazione attuale e permettere l’eventuale ripristino di quest’ultimo. Così da evitare la perdita di dati.
* **Connessioni**: Il database deve permettere l’accesso in parallelo di più utenti (almeno 50 gestori), quindi deve gestire un numero massimo di connessioni pari e/o superiore a 50. Inoltre deve fare in modo di restituire reattivamente i risultati delle query.

.

A.2 Non Functional Requirements

**Performance**

* Il sistema deve essere in grado di gestire fino a 150.000 sensori per città, quindi deve supportare la ricezione di almeno 150.000 messaggi al minuto.
* Il sistema deve permettere l’accesso in parallelo di almeno 50 gestori (modifiche sincronizzate).
* La dashboard deve avere un tempo di refresh inferiore al “min t” per permettere ai gestori di avere una situazione aggiornata della propria area di competenza.
* **Scalability:** Il sistema deve essere in grado di supportare eventuali future espansioni che potrebbero aumentare i valori di Performance.

**Availability**

* Il sistema deve essere accessibile e disponibile 24/24h e 7/7gg; un mancato funzionamento dello stesso potrebbe causare gravi incidenti in quanto non si riuscirebbe a monitorare e localizzare un’anomalia o un malfunzionamento.

**Safety**

* Il sistema deve essere sempre in grado di rilevare un’anomalia o un malfunzionamento, e di calcolare la priorità in maniera precisa e puntuale, senza errori, in modo da notificare l’allerta al gestore. Il gestore edificio può risolvere autonomamente il problema (sostituire il sensore, accendere i climatizzatori, etc..) oppure può contattare le autorità competenti, evitando di mettere a rischio la vita delle persone che occupano un’area.

Inoltre, come ulteriore sicurezza, tutti i diversi tipi di gestore vedono un’anomalia o un malfunzionamento nella propria area con il proprio livello di astrazione.

**Security**

* Il sistema deve essere installato solo sulle macchine presenti nell’ufficio dei gestori (in modo da prevenire che persone esterne possano provare ad entrare nel sistema tramite altri pc e cambiare un valore dei sensori).
* L’accesso da parte di fonti sconosciute alle macchine è tenuto sicuro tramite un login. Se il login fallisce per 5 volte di seguito il sistema blocca l’accesso a quella macchina per un minuto per evitare attacchi bruteforce.

A.3 Excluded Requirements

* Il sistema non deve fornire un servizio per l’interrogazione dei sensori.
* Il sistema non deve fornire un servizio per la gestione di apparecchiature, come condizionatori, impianti riscaldanti etc.., che possano modificare lo stato delle variabili ambientali.

A.4 Assumptions

* Un gestore sarà responsabile di una sola area (città, zona, edificio).
* In un singolo edificio area vi possono essere da 10 a 200 sensori.
* Per ogni stanza deve essere presente almeno un sensore di ogni tipologia (esclusa la tipologia “fumo”).
* I sensori di fumo inviano solo segnali di stato: 0 (funzionante) 1 (non funzionante); e segnali di rilevamento positivo/negativo: 0 (assenza di fumo), 1 (presenza di fumo). L’invio avviene ad intervalli di 1 minuto. Questi sensori di fumo vengono posti soltanto in stanze importanti (aule frequentate da molte persone, aule conferenze, aule che contengono dispositivi tecnologici ecc) per non eccedere con i costi e per avere un feedback immediato in caso di incendio.
* Intervalli di invio valori ambientati (t) :
* Temperatura: t = 60s
* Umidità: t = 90s
* Pressione: t = 120s
* Luminosità: t = 300s
* Fumo t = 60s

Nel momento in cui viene rilevata un’anomalia, “t” si riduce in base alla priorità di quest’ultima:

* + Bassa priorità = t/2
  + Media priorità = t/4
  + Alta priorità = t/6
* LIVELLI DI ALLERTA =
  + Verde: situazione normale (no anomalie/malfunzionamenti)
  + Giallo: bassa priorità
  + Arancione: media priorità
  + Rosso: alta priorità
  + Pericolo: situazione critica che può mettere in pericolo la vita delle persone

BASSA:

**MALFUNZIONAMENTI** = Se solo un sensore per stanza non funziona ma ce n’è almeno un altro dello stesso tipo che funziona.

**ANOMALIE** = Se un solo un tipo di sensore supera i valori sotto elencati:

* temperatura = +- 2 gradi centigradi rispetto a min e max temp
* pressione = +- 50 hpascal rispetto a min e max pres
* umidità =+- 10% rispetto a min e max um
* luminosità =+- 100 lux rispetto a min e max lum

MEDIA:

**MALFUNZIONAMENTI** = Se più sensori in una stanza non funzionano ma ce n’è almeno un altro dello stesso tipo che funziona.

**ANOMALIE**= Se piu tipi di sensori superano i valori sotto elencati:

* temperatura = +- 2 gradi centigradi rispetto a min e max temp
* pressione = +- 50 hpascal rispetto a min e max pres
* umidità =+- 10% rispetto a min e max um
* luminosità =+- 100 lux rispetto a min e max lum

ALTA:

**MALFUNZIONAMENTI** = Se i malfunzionamenti dei sensori fanno in modo che anche solo una variabile ambientale non abbia sensori attivi che rilevano valori.

**ANOMALIE**= Se anche un solo sensore di un tipo qualsiasi supera i valori sottoelencati:

* temperatura = +- 5 gradi centigradi rispetto a min e max temp
* pressione = +- 100 hpascal rispetto a min e max pres
* umidità =+- 20% rispetto a min e max um
* luminosità =+- 250 lux rispetto a min e max lum

PERICOLO:

**MALFUNZIONAMENTI** = Se i malfunzionamenti dei sensori fanno in modo che in una stanza non ci sono sensori attivi.

**ANOMALIE**= Se anche un solo sensore di un tipo qualsiasi supera i valori sottoelencati:

* temperatura = +- 10 gradi centigradi rispetto a min e max temp
* pressione = +- 250 hpascal rispetto a min e max pres
* umidità =+- 30% rispetto a min e max um
* luminosità =+- 500 lux rispetto a min e max lum

***A.5 Prioritization***

* **Salvataggio dati sensori (5)**
* **Calcolo priorità e pericoli (5)**
* **Visualizzazione stato edificio(5)**
* **Visualizzazione sensori (5)**
* **Visualizzazione pericolo(5)**
* **Invio dati(5)**
* **Salvataggio dati valori(5)**
* **Gestione sensori (5)**
* **Visualizzazione stato zona(4)**
* **Visualizzazione stato città(4)**
* **Allerta sicurezza (4)**
* **Fault tolerance(3)**
* **Visualizzazione valori medi(3)**
* **Database centralizzato(3)**
* **Connessione(2)**
* **Gestione utenza (1)**

B. Software Architecture

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

**CD1-Sistema Completo**

La parte fondamentale è il componente CalcoloPriorità, il quale, assieme a CalcoloPericolo, permettono di assegnare ad ogni area visibile nella Dashboard il giusto livello di allerta, analizzando i dati rilevati dai sensori. La Dashboard fornisce una View adatta al tipo di Gestore connesso.

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

**SD1-Aggiornamento dati**

Il ControllerEdificio richiede un aggiornamento dei dati da visualizzare.

Il CheckPriorityRoom richiede i dati dei sensori dal Database, viene fatto il calcolo della priorità e segnalati eventuali pericoli.

**SD2-Settaggio Valori**

Il GestoreEdificio può settare il MIN VALUE e MAX VALUE per i sensori dell’edificio a lui assegnato.

Tali valori saranno poi inviati al Database tramite il component “SettaggioValori”.

**SD3-Visualizza Edifici**

Il GestoreZona, quando entra nel sistema, richiede implicitamente alla Dashboard la visualizzazione degli edifici a lui assegnati.

Il ControllerZona utilizza l’ ”AggiornamentoDati” per ogni edificio appartenente alla zona di competenza del GestoreZona e restituisce alla Dashboard gli edifici con le relative priorità.

**SD4-Visualizza Stanze**

Il GestoreEdificio, quando entra nel sistema, richiede implicitamente alla Dashboard la visualizzazione delle stanze a lui assegnati.

Il ControllerEdificio utilizza l’ ”AggiornamentoDati” e restituisce le stanze con le relative priorità e pericoli.

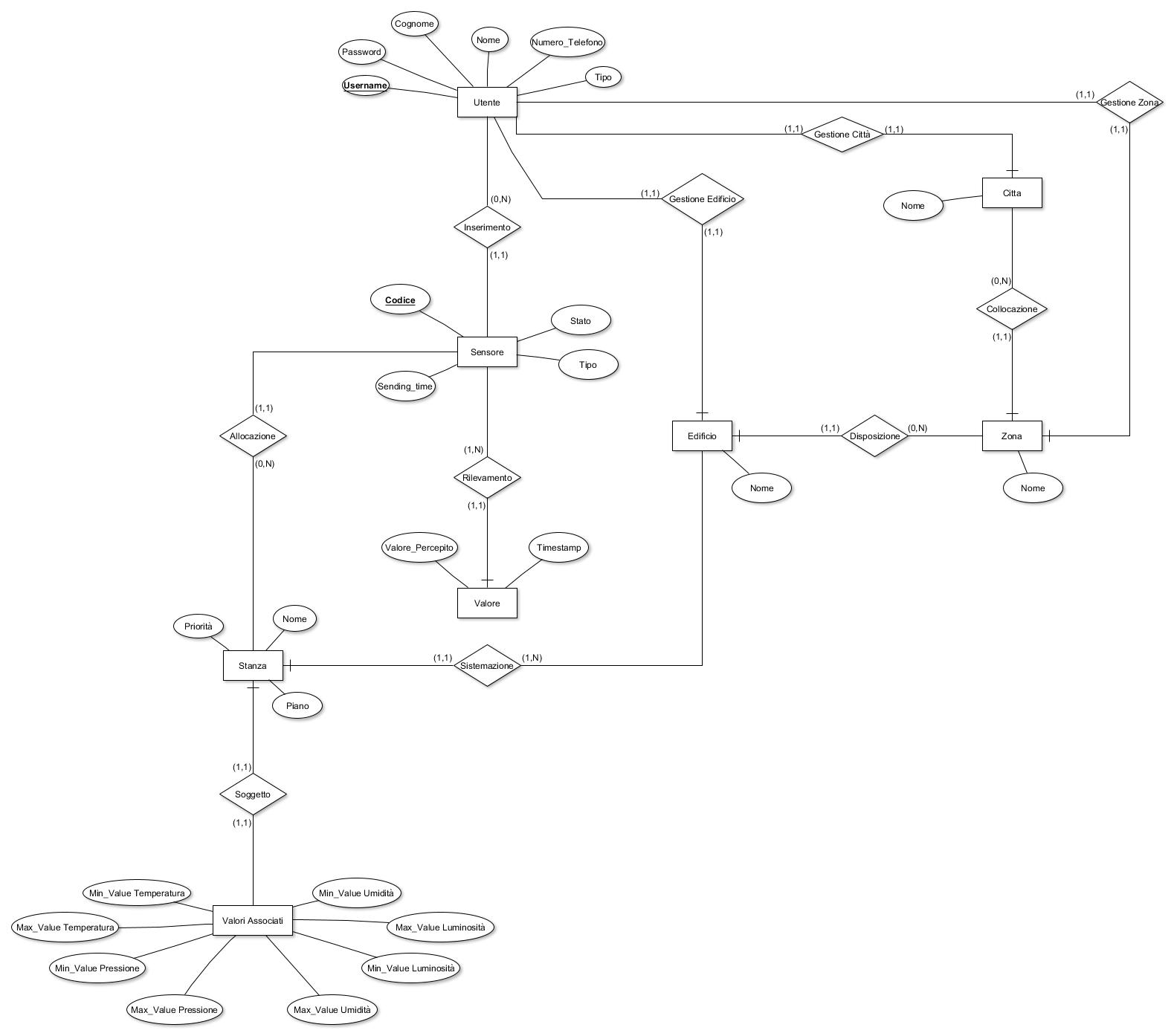
Il GestoreEdificio può contattare le autorità competenti richiedendo alla Dashboard i numeri di quest’ultime.

**SD5-Visualizza Zone**

Il GestoreCittà, quando entra nel sistema, richiede implicitamente alla Dashboard la visualizzazione delle zone a lui assegnate.

La Dashboard richiede al ControllerCItta i dati delle zone appartenenti alla città e per ogni zona vengono calcolate le priorità degli edifici associati utilizzando “AggiornamentoDati”.

C. ER Design



Documentazione delle entità:

|  |  |  |
| --- | --- | --- |
| Entità | Descrizione | Attributi |
| Utente | ***Utente, inteso come Amministratore e/o Gestore di edificio, zona o città, che si interfaccia con il sistema.*** | **Username** (Primary Key), Nome, Cognome, Numero\_Telefono, Tipo, Password. |
| Sensore | ***Dispositivo in grado di trasformare la grandezza in ingresso in un particolare valore di altra natura (valori interi, valori decimali).*** | **Codice** (Primary key), Stato, Tipo, sending\_time. |
| Città | ***Macro-area contente al suo interno una o più zone.*** | Nome. |
| Zona | ***Area sensibilmente più ristretta di una città.*** | Nome. |
| Edificio | ***Costruzione contenente al suo interno un certo numero di piani e stanze.*** | Nome. |
| Stanza | ***Si intende un ambiente destinato ad ospitare al suo interno un certo numero di sensori.*** | Nome, piano. |
| Valore | ***Particolare dato, tipicamente di natura decimale e/o intera, registrato da un sensore.*** | Valore\_Percepito, TimeStamp (Giorno, Ora, Minuti e Secondi in cui un certo valore è stato registrato). |
| Valori Associati | ***Valori minimali e massimali associati ad una certa stanza, tipicamente di natura decimale/intera. Si presuppone infatti che ogni gestore di edifico abbia la possibilità di impostare valori minimali e massimali accettabili per una certa stanza. Si avranno valori minimali e massimali per ogni variabile ambientale.*** | Min\_Value Temperatura, Max\_Value Temperatura, Min\_Value Pressione, Max\_Value Pressione, Min\_Value Umidità, Max\_Value Umidità,  Min\_Value Luminosità,  Max\_Value Luminosità. |

Documentazione delle Relazioni:

|  |  |  |
| --- | --- | --- |
| Relazione | Descrizione | Entità Coinvolte |
| Gestione Città | ***Associa un gestore ad una certa Città.*** | Utente-Città |
| Gestione Zona | ***Associa un gestore ad una certa Zona****.* | Utente-Zona |
| Gestione Edificio | **Associa un gestore ad un certo Edificio.** | Utente-Edificio |
| Collocazione | ***Specifica la distribuzione di un certo numero di Zone, relative ad una singola città.*** | Città-Zona |
| Disposizione | ***Specifica la collocazione di un certo numero di edifici all’interno di una zona.*** | Zona-Edificio. |
| Rilevamento | ***Associazione di un certo valore di natura decimale/intera registrato da un certo Sensore.*** | Sensore-Valore |
| Distribuzione | ***Specifica la collocazione di un certo numero di stanze all’interno di un singolo edificio.*** | Edificio-Stanza |
| Allocazione | ***Specifica la distribuzione di Sensori all’interno di una stanza.*** | Stanza-Sensore |
| Soggetto | ***Specifica la relazione che intercorre tra una stanza ed i corrispondenti valori minimali e massimali (relativi a variabili ambientali) associati ad essa.*** | Stanza-Valori Associati |

D. Class Diagram of the implemented System

**CLD1-Model Diagram**

Descrive le entità del sistema e i metodi che queste mettono a disposizione per l’accesso ai dati dei singoli oggetti. Abbiamo modellato la città come una lista di zone, la zona come una lista di edifici e un edificio come una lista di stanze.

**CLD2-DAO e Priority Controller**

Questo class diagram contiene i package ‘DAO’ e ‘Priority Controller’; il primo contiene tutta una serie di metodi i quali si occupano di interrogare il database; il secondo sfrutta queste interrogazioni per calcolare la priorità di ogni area che è attualmente monitorata da un gestore. Molto importanti le classi dei Thread, le quali rendono possibili l’aggiornamento ogni 10 secondi delle view.

**CLD3-MVC**

Descrive in che modo il sistema implementi il modello MVC; ogni view è collegata ad un controller il quale la popola a seconda dei dati del model; questi controller permettono all’utente di interagire con le view per visualizzare i dati desiderati ed evenuali anomalie o malfunzionamenti.

E. Design Decisions

1. **Calcolo Priorità e Pericoli**

Questa componente si pone a metà tra il database e la dashboard ed ha il compito di analizzare i dati rilevati dai sensori per etichettarli con **il giusto livello di priorità**; al suo interno si trova anche la componente responsabile per l’**identificazione dei pericoli**, ovvero di quelle anomalie che potrebbero mettere a rischio la **vita** delle persone. Attraverso un algoritmo ben studiato e sicuro, il sistema sarà in grado di riconoscere una situazione pericolosa ed inviare immediatamente al responsabile dell’area un’**allerta di pericolo** ben visibile.

1. **Rilevatori di Fumo**

Per aumentare il **livello di safety** del sistema abbiamo deciso, dopo un’attenta analisi, di inserire dei sensori in grado di rilevare immediatamente la presenza di un incendio all’interno della nostra rete di sensori; per ottenere un ottimo compromesso tra costi e utilità, abbiamo deciso di inserire questo tipo di sensore solo nelle **stanze più importanti di un edificio (denotate dall’attributo “priorità=high” nel DB)**, *come ad esempio un’aula magna di una scuola*. I sensori invieranno *‘0’* in caso di stato normale, *‘1’* in caso di fumo rilevato; l’algoritmo per il calcolo dei pericoli riconoscerà immediatamente il dato e manderà il segnale di pericolo al gestore associato all’edificio in questione.

1. **Model-View-Controller Pattern Design**

Il sistema che stiamo sviluppando è basato sulle **interfacce;** questo perché ad ogni tipo di utente che logga nel sistema verrà proposta un interfaccia **ad-hoc al suo ruolo**. Ad esempio ad un *GestoreZona* verranno proposte le informazioni riguardanti esclusivamente gli edifici della zona a lui associata, ad un livello più alto rispetto ad un *GestoreCittà* , al quale verrà proposto un livello più dettagliato del singolo edificio di sua competenza.

Per queste motivazioni abbiamo adottato il pattern **Model-View-Controller** che ci consente, oltre di soddisfare le necessità sopra descritte, di **separare** molto distintamente le logiche all’interno del nostro sistema.

In particolare nella parte **Model** avremo i dati relativi ad *utenti*, *aree*, *settaggi*, *sensori* e *dati rilevati*. La parte **View** conterrà le interfacce di: *login*, *visualizzazione zone*, *visualizzazione edifici*, *visualizzazione piani e stanze*, *settaggio valori*, *gestione aree ed utenza*. Queste interfacce sono **prive** di logica, dato che quest’ultima si trova nella parte **Controller** del sistema; infatti questa parte contiene la *logica per la visualizzazione delle interfacce*, che le aggiorna alla ricezione di input da parte dell’utente, la logica per il *calcolo delle priorità e dei pericoli*, per le *notifiche tra gestori* e per *l’aggiornamento dei valori settabili*.

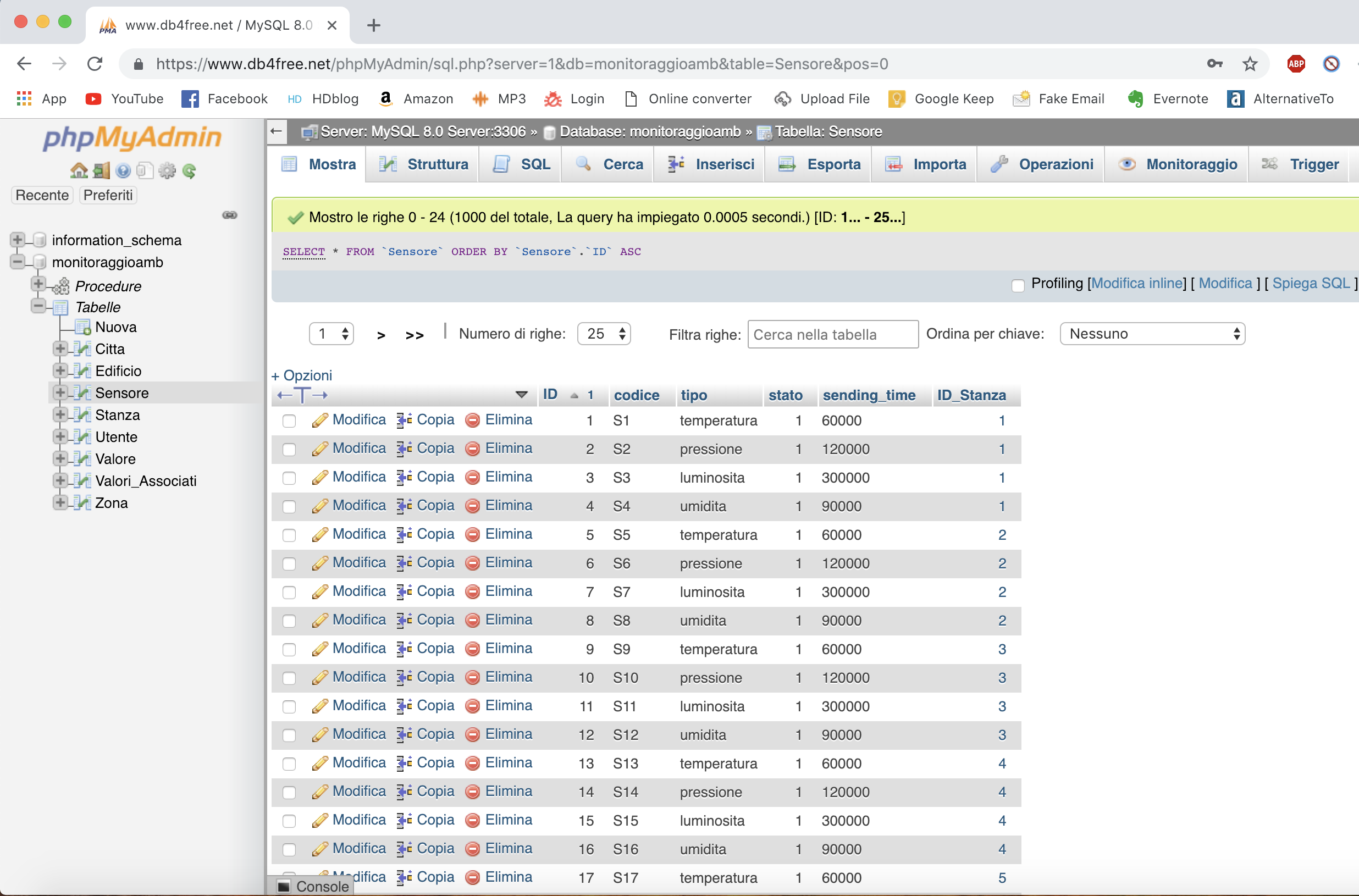
1. **Database relazionale**

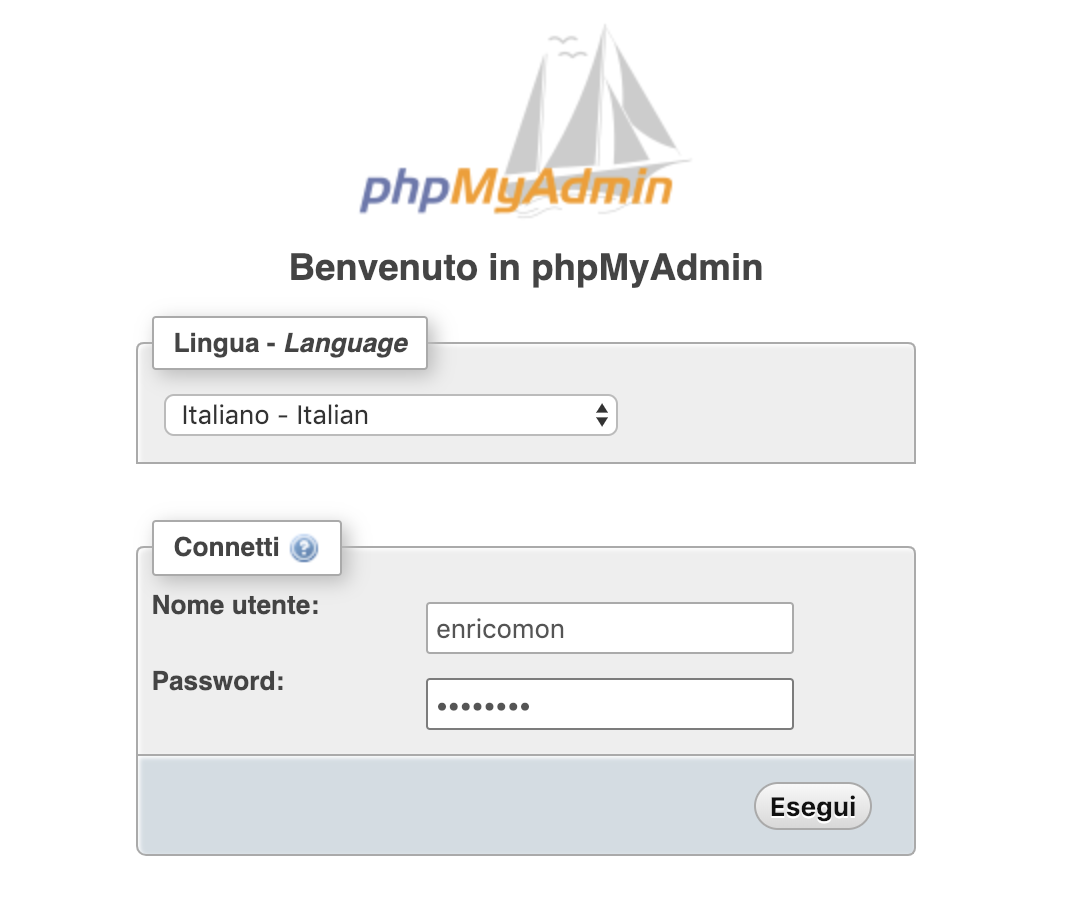
Dopo una lunga ed approfondita ricerca, abbiamo identificato in **MySQL 8.0**, database relazionale, il giusto protagonista per lo storage dei *dati rilevati dai sensori*, delle *informazioni degli utenti e delle aree*, dell’*assegnamento tra gestori ed aree* e dei *valori minimali e massimali* delle stanze.

Attualmente **MySQL 8.0**, con 50 utenti collegati, è in grado di sostenere un massimo di 600.000 richieste al secondo, valore ampiamente valido per i nostri scopi. Abbiamo eseguito test in locale con 150.000 sensori funzionanti e il database regge grazie all’utilizzo dei Thread.

1. **Database Centralizzato**

Il team ha deciso di sfruttare un database **centralizzato** per permettere la *modifica sincronizzata di tutti i dati nel database*. Il database è stato creato come sopra descritto in locale utilizzando MySql Workbench. Tramite uno script Java vi abbiamo inserito tutti i dati delle aree dell’utenza e dei sensori. Il **dump** del database successivamente è stato caricato su un host DB (db4free.net) che funge da **server** per l’applicazione. Infatti questo host permette *l’accesso in remoto ad applicazioni esterne (il server è protetto da password).* Eseguendo l’applicazione, i sensori vanno ad inviare i valori rilevati direttamente al database online e il calcolo della priorità agisce sui dati presenti in esso. Inoltre, accedendo al pannello di controllo del server sql è possibile gestire in modo intuitivo tutti i dati tramite la piattaforma “php my admin” messa a disposizione





1. **Linguaggio Java**

Dopo un’attenta valutazione il team ha deciso di sviluppare l’applicativo nel **linguaggio Java**. Innanzitutto perché è il linguaggio più conosciuto dai membri del team e con il quale avrebbe potuto esprimersi al meglio. In secondo luogo è uno dei linguaggi più popolari e diffusi, quindi è vantaggioso anche dal punto di vista della **leggibilità, futura espansione e scalabilità**. Inoltre essendo molto versatile permette di sviluppare applicazioni per qualsiasi piattaforma (**portabilità**) e in modo più intuitivo rispetto ad altri linguaggi che non sono ad alto livello e che non hanno un elevato livello di astrazione dal linguaggio macchina.

Come ultima caratteristica che ci ha aiutato a scegliere abbiamo considerato che java è un linguaggio *orientato agli oggetti*, quindi adatta nel nostro caso, in cui tramite le classi si possono meglio gestire e distinguere le varie componenti del sistema e le relative relazioni e funzioni.

1. **JavaFX**

Per quanto riguarda lo sviluppo della grafica del sistema abbiamo scelto JavaFX in quanto si sposa con il linguaggio di sviluppo java essendo basato su quest’ultimo, ed in combinazione con esso permette di creare applicazioni all’avanguardia.

JavaFX oltre ad una grande libreria di funzionalità grafiche, utilizza un vero e proprio linguaggio chiamato “java FX script” orientato alla programmazione grafica e rende lo sviluppo di applicazioni grafiche particolarmente agevolato.

Inoltre per la realizzazione delle interfacce abbiamo usato il tool “Scene Builder” che evita la creazione manuale di un file FXML e permette costruirle con semplici drag and drop. Inoltre supporta i css e integra un visualizzatore FXML il quale ci consente di visualizzare un’anteprima durante la creazione del progetto. Infine permette anche di aprire file FXML esistenti e modificarli senza bisogno di scrivere codice.

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

**GUI**

* **Visualizzazione stato città/zona/edificio:** quando un gestore logga nel sistema, il controller Login\_Page\_Controller controlla la giustezza di username e password inseriti; se sono corretti, identifica l’area associata al gestore loggato e la dashboard viene popolata con le sottoaree etichettate da colore(che ne implica lo stato) e nome.
* **Visualizzazione sensori**: un gestore edificio nell’interfaccia ha la possibilità di selezionare una stanza; a quel punto la tabella sulla destra viene popolata dai sensori contenuti nella stanza assieme a stato e valori rilevati.
* **Visualizzazione valori medi**: le aree che un gestore può monitorare sono etichettate da un colore, che varia in base al tipo di allerta, e da dei valori medi, calcolati dinamicamente dal modulo CalcoloPriorità tramite una funzione di average.
* **Visualizzazione pericoli**: i pericoli sono ben visibili in tutti i tipi di interfaccia: infatti quando c’è un pericolo, l’area si etichetta di viola; inoltre per i gestori edificio è presente il bottone “Visualizza pericoli” il quale filtra solo le stanze in pericolo.

**BUSINESS LOGIC**

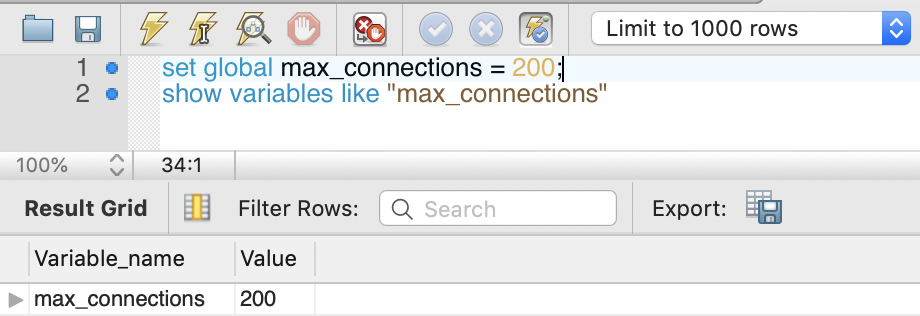
* **Calcolo priorità e pericoli**: questa funzionalità è soddisfatta dal modulo CalcoloPriorità, il quale contiene un algoritmo che calcola le allerte e i pericoli con una affidabilità del 100% (assumendo la veridicità dei dati rilevati dai sensori).
* **Invio Dati:** funzionalità offerta da un applicativo che simula l’invio dei dati rilevati da sensore al database, popolandolo con valori randomici; tuttavia verranno rilevati molto spesso pericoli in quanto valori privi di fondamento.
* **Gestione sensori:** il gestore edificio può cliccare sul bottone “Settaggio Valori” per modificare i MIN e MAX VALUE per ogni tipo di variabile ambientale per stanza.
* **Gestione Utenza:** l’interfaccia “Admin\_Page” permette all’Admin di eseguire queste funzionalità.
* **Allerta sicurezza**: un gestore edificio può visionare i contatti per allertare le autorità cliccando il bottone “Allerta Autorità”.

**DATABASE**

* **Salvataggio dati sensori:** tramite il dump.sql allegato verranno salvati i sensori nel database.
* **Salvataggio dati valori:** lo script “InvioDati.sql” si occupa di popolare il database ad intervalli regolari con nuovi valori rilevati.
* **Fault tolerance:** per soddisfare questo requisito abbiamo creato uno script batch che una volta avviato sfrutta l’utility “MySqlDump”; questo script crea un task pianificato nella macchina sul quale viene avviato che esegue ad intervalli di 5 minuti il dump del database e lo salva nella cartella specificata con il nome “monitoraggio.sql”. Il file di backup ad ogni creazione viene sovrascritto, in modo da avere solo l’ultimo dato aggiornato e non influire sulla memoria di archiviazione. Tutto questo per evitare di perdere eventuali dati in caso di guasti o interruzioni, e poter eseguire il ripristino del database quando necessario.

Nota: lo script, a casa dell’estensione .bat, funziona solo su piattaforma windows.

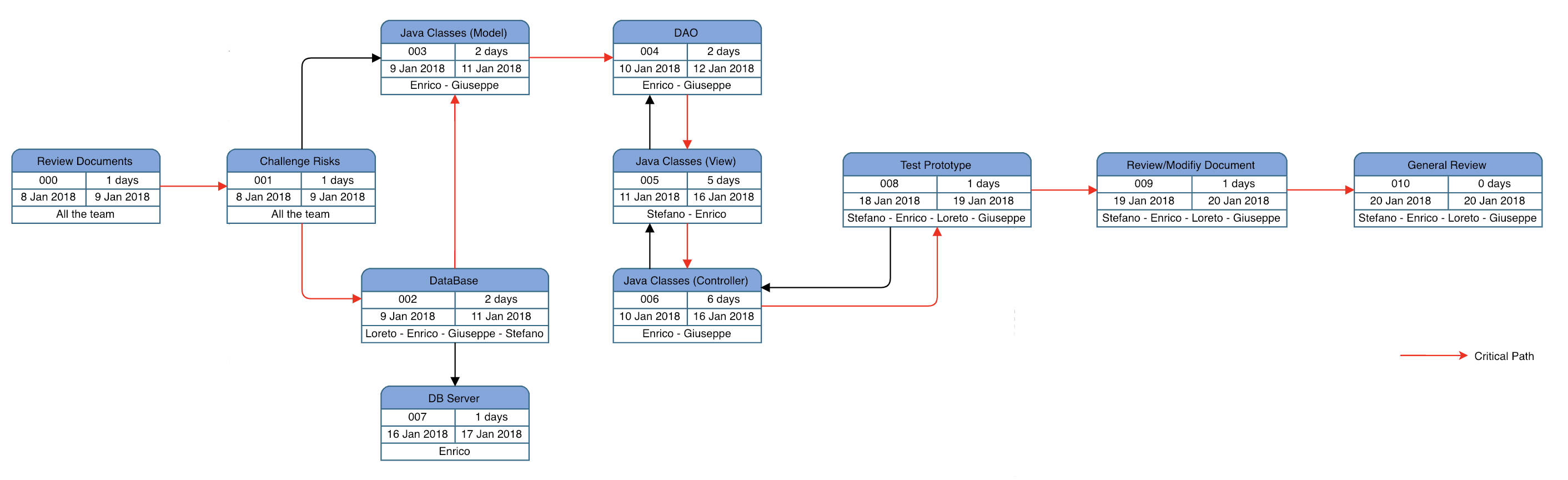
Per i dettagli dello script realizzato cliccare [qui](ScriptDump.pptx).

* **Database Centralizzato**: per soddisfare questo requisito, il team ha scelto, come sopra descritto, di utilizzare un sito di hosting sql, chiamato db4free.net, che permette di inserire un database online, nel loro server, renderlo accessibile da applicazione esterne in remoto. Questo host è reso sicuro da una password per accedere al pannello di controllo. E rende particolarmente semplice la gestione dei dati in quanto integra e mette a disposizione la piattaforma “PHP My Admin”. Questo permette le modifiche sincronizzate al database e mette a disposizione dei gestori tutti i dati aggiornati.
* **Connessioni**: Abbiamo fatto in modo di limitare il tempo di vita delle connessioni MySql, aprendole solo quando si interroga il DB, e chiudendole appena non risultassero più utili per non lasciarle aperte durante tutta la durata dell’applicazione. Il limite di connessioni massime al database di default in MySql è impostato a 151. Abbiamo aumentato il limite di default di 150 connessioni massime di MySql a 200. Quindi possono avvenire fino 200 connessioni simultanee. 

**NON FUNZIONALI**

* **Performance:** abbiamo effettuato test con 150.000 sensori e il database riesce a supportare l’invio dei loro segnali.
* **Availability:** Il database centralizzato caricato su un host online soddisfa questo requisito rendendolo sempre accessibile.
* **Safety:** la safety è soddisfatta dalla prontezza con cui il modulo CalcoloPriorità riconosce un pericolo e lo rende visibile al gestore immediatamente.
* **Security:** il sistema è protetto da login; dopo 5 tentativi falliti, bisognerà aspettare 1 minuto prima di poter digitare nuovamente: questo terrà lontano eventuali tentativi di bruteforce.

G. Effort Recording

*****PERT***

***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.*

***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***

***Ore totali: 209h;***

***Learning: 36h;***

***Doing: 173h;***

Appendix. Code

**ALGORITMO CALCOLO DELLE PRIORITA’**

**

Per il calcolo delle priorità abbiamo suddiviso il calcolo per i malfunzionamenti da quello per le anomalie.

Per questi calcoli passiamo all’algoritmo **i sensori delle stanze associate al gestore** e viene assegnato un numero ad ogni sensore, ovvero 0 se non ci sono problemi, 1 se bassa priorità, 2 se media priorità, 3 se alta priorità, 4 se pericolo.

Per i malfunzionamenti viene assegnato:

- 0 se non ci sono malfunzionamenti;

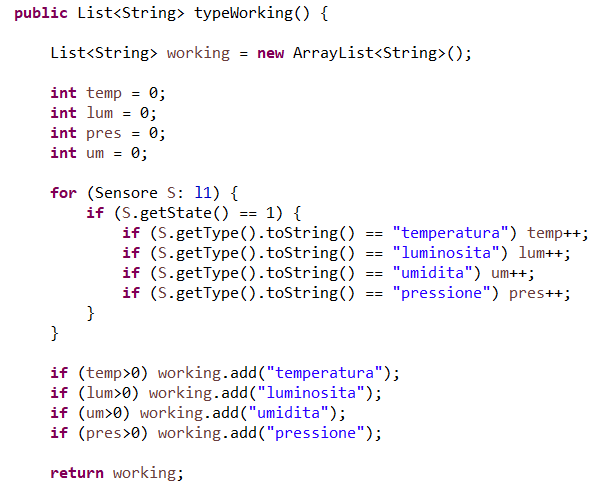
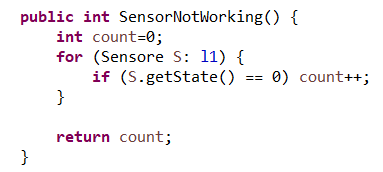
- 1 se c’è un solo sensore malfunzionante ma ce n’è almeno uno dello stesso tipo funzionante;

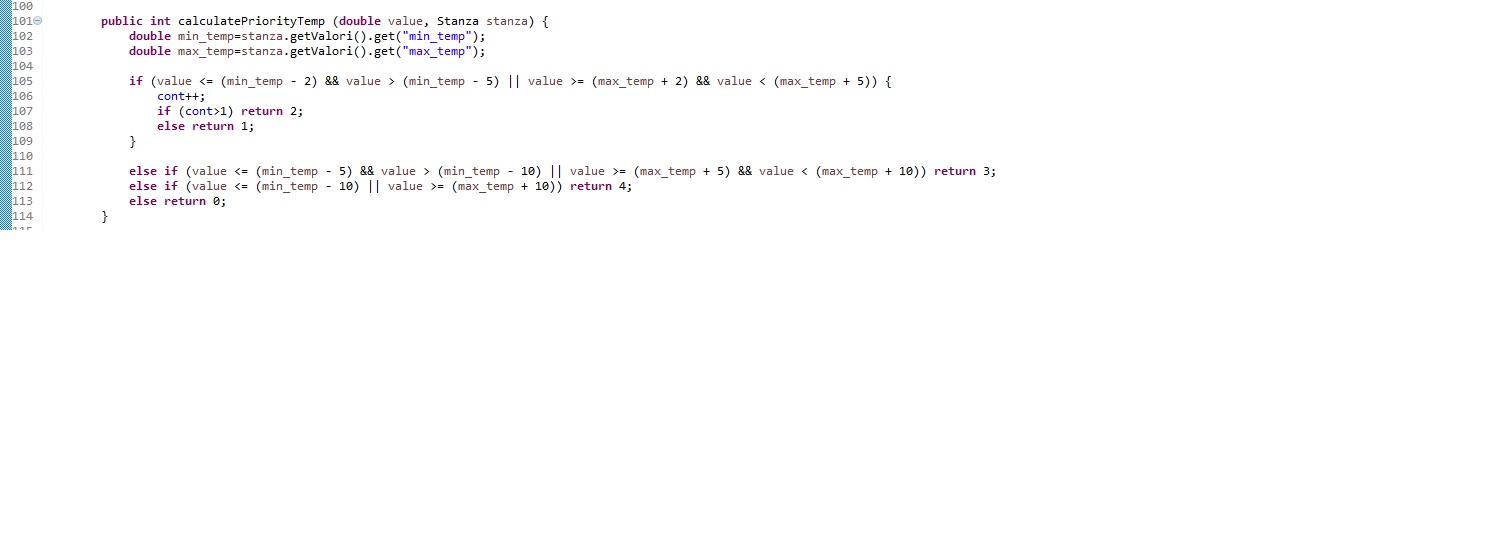
- 2 se più sensori non funzionano ma ce ne sono altri dello stesso tipo funzionanti;

- 3 se per almeno una variabile ambientale non c’è nessun sensore che la può rilevare;

- 4 se più variabili ambientali non possono essere rilevate.

Per calcolare il numero di malfunzionamenti o la mancata presenza di alcune variabili ambientali usiamo gli algoritmi “sensorNotWorking()”e “typeWorking()”.



Per le anomalie, invece, da come si può vedere in allegato, prendiamo come esempio il calcolo per i sensori di temperatura.

Assegnamo, quindi:

- 0 se non ci sono anomalie;

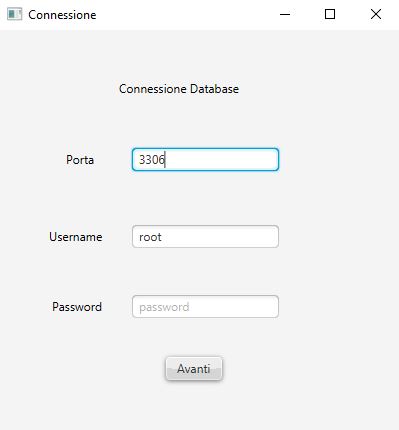
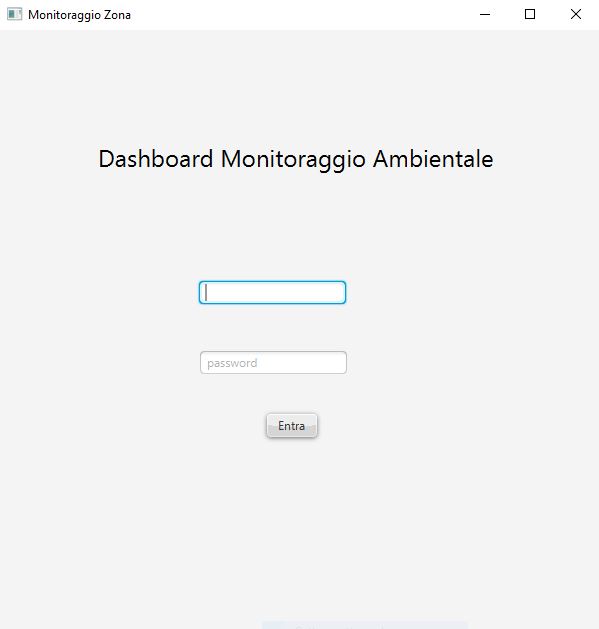
- 1 se la temperatura minima o massima consentita viene superata al massimo di ±2;

- 2 se più di un sensore supera la temperatura minima o massima consentita di ±2;

- 3 se la temperatura minima o massima consentita viene superata al massimo di ±5;

- 4 se la temperatura minima o massima consentita viene superata al massimo di ±10.

**AVVIO DEL PROTOTIPO**

1. Importare il file **“Dump.sql”** nel programma di gestione Database; esso conterrà i dati relativi all’utenza, le aree e i sensori, ed è già popolato con valori rilevati di prova;
2. Avviare il file **“monitoraggio.jar”;**
3. All’avvio verrà mostrata questa schermata:
4. Inserire nei campi appositi porta, username e password per la connessione al database installato localmente e premere su **“Avanti”;**
5. A questo punto si aprirà la schermata di **Login**;
6. Inserire nella schermata di Login l’username e la password come segue:

- per entrare nella dashboard come Gestore Edificio:

Username:”**utente1**” e come password:”**utente**”

- per entrare nella dashboard come Gestore Zona: Username:”**utente21**” e come password:”**utente**”

- per entrare nella dashboard come Gestore Città: Username:”**utentecitta**” e come password:”**utente**”

- per vuole entrare nella dashboard come Amministratore:

Username:”**admin**” e come password:”**admin**”

La Dashboard visualizzata è descritta in maniera dettagliata in A1.2:[GUI Requirements.](GUI%20Requirements.pptx)

Per testare l’applicazione abbiamo usato lo script java “InvioDati.jar” che si occupa di simulare l’inserimento dei valori ambientali rilevati dai sensori nel database utilizzando dei thread che vengono eseguiti in un tempo che varia a seconda della priorità a lui associata (come descritto in A.4: ASSUMPTIONS).