

**“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	20/01/2019
Deliverable	Deliverable Finale - 3
Team (Name)	MM-GTE

Team Members		
Name & Surname	Matriculation Number	E-mail address
Miriana Pompilio	248492	miriana.pompilio@student.univaq.it
Marzia Piccirilli	246972	marzia.piccirilli@student.univaq.it
Erika Biondi	247042	erika.biondi@student.univaq.it
Giulia Scoccia	249503	giulia.scoccia@student.univaq.it
Thomas Moretti	247286	thomas.moretti@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*

Sommario

<u>Introduzione</u>	4
<u>List of Challenging/Risky Requirements or Tasks</u>	5
<u>A. Requirements Collection</u>	6
<u>A.0 Detailed Scenarios</u>	6
<u>A.1 Functional Requirements</u>	7
<u>Use Case</u>	8
<u>Tabelle Di Cockburn</u>	8
<u>A.1.1 GUI Requirements</u>	11
<u>A.1.2 Business Logic Requirements</u>	13
<u>A.1.3 DB Requirements</u>	13
<u>A.2 Non Functional Requirements</u>	13
<u>B. Software Architectures</u>	15
<u>B.1 Component Diagram</u>	15
<u>B.2 Sequence Diagram</u>	17
<u>C. E-R Diagram</u>	21
<u>D. Class Diagram</u>	23
<u>E. Design Decisions</u>	24
<u>F. Explain how the FRs and the NFRs are satisfied by design</u>	31
<u>G. Effort Recording</u>	35
<u>Pert</u>	35
<u>Logging</u>	36
<u>Categorization</u>	37
<u>Appendix. Code</u>	38

Introduzione

Si vuole realizzare un sistema in grado di fornire informazioni di monitoraggio ambientale all'interno di un ospedale.

Il monitoraggio è garantito tramite l'installazione di sensori (di temperatura, umidità etc.) all'interno delle aree di interesse.

L'intera struttura ospedaliera sarà composta da un totale di sette edifici, due dei quali saranno riservati per la mensa e l'obitorio che saranno ad unico piano; mentre i restanti cinque avranno caratteristiche comuni, quali:

- Ogni edificio avrà 3 piani;
- Ogni piano ospiterà 3 reparti più le zone comuni (corridoio, bagno e sala di attesa);
- L'unica eccezione sarà sollevata per il piano terra dell'edificio Uno in quanto, all'interno dei propri uffici amministrativi, sarà prevista una zona dedicata ai server(server di amministrazione e server di supporto).

Dato che il sistema è gestito da un unico server , potrà essere espandibile e quindi il numero di edifici potrà variare e il sistema potrà essere usato per un qualsiasi altro contesto, poiché non abbiamo imposto vincoli al sistema che lo facciamo ricondurre soltanto all'ambiente ospedaliero. In un qualsiasi altro contesto, basterà collegare i gestori ai server e DB che abbiamo istanziato.

Il sistema avrà una **dashboard** per ogni area, consultabile dal responsabile area, una per ogni edificio, consultabile dal responsabile di edificio, ed infine, si avrà una **dashboard** principale consultabile a livello amministrativo.

Le informazioni saranno gestite a livello locale in un database che conterrà i dati del nostro sistema. Ogni dashboard di area visualizzerà soltanto i dati relativi alla sua parte di database. Analogamente ogni dashboard di edificio avrà la sua parte riservata.

Soltanto la dashboard dell'amministrazione avrà accesso a tutto il database e quindi a tutti i dati del sistema.

Sarà previsto, inoltre, un altro database di supporto che servirà ad evitare , in caso , la perdita dei dati.

Ogni reparto costituisce un'area dell'ospedale e sarà provvista di **sensori**, i quali invieranno le proprie informazioni relative all'ambiente, al Database.

Inoltre, parallelamente, trasmetteranno i propri dati al database di supporto (backup).

Il sistema sarà in funzione 7 giorni su 7, h24. I dati, quindi, saranno sempre consultabili dai corrispondenti responsabili in ogni momento e saranno aggiornati e gestiti a seconda della loro priorità.

List of Challenging/Risky

Challenging Task	Date the task is identified	Date the challenge is resolved	Explanation on how the challenge has been managed
Simulazione invio dati da parte dei sensori	11/01/2019	14/01/2019	Abbiamo simulato l'invio dei dati , attraverso Socket Client/Server.
Cosa prototipare	08/01/2019	08/01/2019	Visto la mancanza di esperienza e tempo, abbiamo dato priorità all'invio dei dati e a un'interfaccia basilare della dashboard di amministrazione sulla quale mostriamo il riconoscimento da parte dei responsabili e la visualizzazione di un edificio, un'area e una zona.
Come distinguere i responsabili di area e di edificio.	11/01/2019	11/01/2019	Abbiamo modificato il DB , in modo da riuscire a differenziare i tipi di responsabile.
Salvataggio dati nel DB non appena vengono inviati	14/01/2019	15/01/2019	Implementazione della classe Connessione , che permette tramite query la connessione al DB e l'inserimento dei dati in esso.

A. Requirements Collection

A.0 Detailed Scenarios

Qui di seguito sono descritti 3 possibili scenari che possono verificarsi nel nostro sistema.

- 1) Tutti i valori sono nella norma e compaiono in verde sulle dashboard.
- 2) Problemi di lieve entità : sulle dashboard compaiono alcuni valori in arancione.
- 3) Situazione di emergenza : sulle dashboard compaiono valori in rosso.

Si presume che nel momento in cui si verificano i tre scenari sopra descritti , Carla, Fabio e Miriana rispettivamente responsabile di area , edificio e amministrazione , abbiamo fatto il riconoscimento nel sistema e quindi siano loro a consultare i dati sulle dashboard in quel momento.

- 1) Tutti i valori compaiono in verde, Carla può fare zoom-in e zoom-out nelle varie stanze della sua area di appartenenza, e consultare i dati , analogamente Fabio e Miriana , possono farlo sulle rispettive dashboard di edificio e amministrazione , nelle quali hanno ovviamente una visione più ampia dei dati.
- 2) Supponiamo, ad esempio, valori in arancione nella stanza 3, area 2, edificio 4. Carla clicca sull'icona della stanza che comparirà in arancione facendo quindi zoom in sulla stanza, per individuare meglio dove si trova l'errore e da quale sensore proviene ed avvisa Fabio che provvederà ad attuare il protocollo necessario per il problema riscontrato(ad esempio avvisare qualcuno di competenza che vada sul posto a controllare). Fabio, si occuperà a sua volta di mettersi in contatto con Carla per accertarsi che l'errore sia stato notato e quindi segnalato all'amministrazione. L'errore verrà “risolto” non appena il valore tornerà nella norma o in caso di malfunzionamento del sensore, non appena esso riprenderà a funzionare, e quindi ricomparirà il valore in verde sulla Dashboard. Nel caso in cui ci sia un cambio di turno e qualche errore non sia stato ancora risolto, i responsabili provvederanno ad avvisare i colleghi successivi degli eventuali errori segnalati, ma non ancora risolti.

- 3) Supponiamo che tanti valori nella stanza 4 , area 1 , edificio 5 appaiano in rosso sulla Dashboard e quindi ci sia un'emergenza. In quel caso Carla fa scattare un allarme (ad esempio premendo un pulsante) che attiverà la procedura richiesta per quel tipo di situazione. Sarà cura inoltre, di Fabio, nel caso sia lui per primo a notare l'emergenza, o semplicemente per accertamento, di contattare Carla in modo tale da far scattare l'allarme nel primo caso o semplicemente per informarsi se l'allarme è scattato nel secondo caso. Questo tipo di procedura vale sia per le aree ad alta che a bassa priorità. Inoltre si presume che questo tipo di emergenze vadano risolte nel minor tempo possibile da parte degli organi di competenza.

A.1 Functional Requirements

Il nostro sistema prevede otto attori:

- Sensore
- Dashboard (area , edificio e amministrazione)
- Responsabile: che a sua volta comprende tre tipologie diverse ovvero il responsabile dell'area, il responsabile di edificio e il responsabile dell'amministrazione.
- Amministratore

Ogni attore può svolgere, sul nostro sistema, le azioni che gli competono.

AZIONI (REQUISITI FUNZIONALI) DIVISI PER ATTORE:

SENSORE: svolge l'azione di inviare i dati che esso rileva al database.

DASHBOARD DI AREA: prende i dati, forniti dal database, relativi ai sensori che si trovano all'interno di essa.

DASHBOARD DI EDIFICIO: prende i dati ,relativi alle aree al suo interno, dal database.

DASHBOARD AMMINISTRAZIONE: permette di consultare i dati relativi a tutto il sistema, ovvero tutti i dati all'interno del database.

RESPONSABILE: sono coloro che possono consultare e visionare i dati sulla dashboard. Tale azione è permessa dopo che il responsabile effettua un login sul sistema attraverso un codice di riconoscimento.

AMMINISTRATORE: è l'unico che ha la piena gestione sul sistema e può operare su di esso in caso di malfunzionamento a livello software, inoltre anch'esso ha accesso alla consultazione dei dati.

Use Case

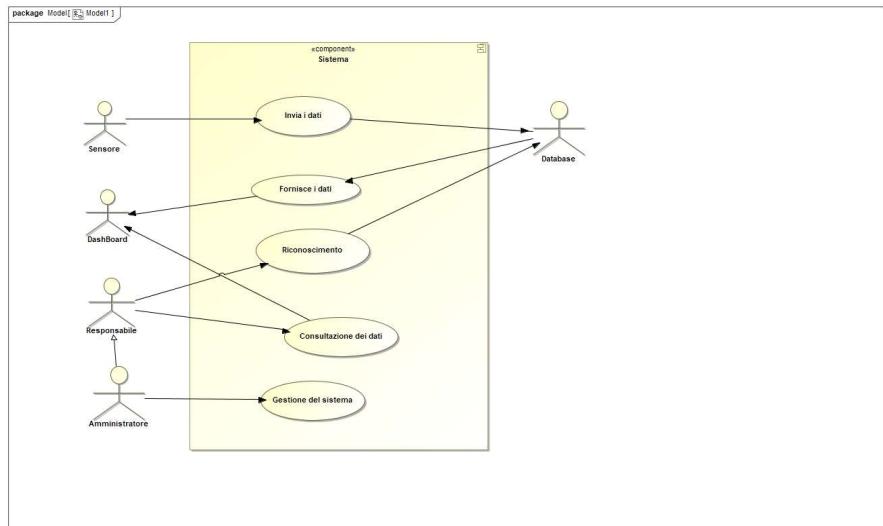


Tabelle di Cockburn

USE CASE #	Invia i dati	
Goal in Context	Invia i dati al database	
Scope & Level	Sistema principale, attività principale	
Preconditions	Il Sistema e tutti i sensori sono attivi e funzionanti.	
Success End Condition	In segnale arriva con successo ed è di colore dell'area corrispondente, rosso o arancione nel database	
Failed End Condition	Il dato non arriva al database	
Primary	Sensore	
Trigger	Il sensore rileva ed entra in funzione lo UC	
DESCRIPTION	Step	Action
	1	Rilevamento del corrispondente valore nell'area in cui esso è collocato
	2	Salvataggio momentaneo dei valori prelevati su di esso
	3	Invio dei dati al Database
EXTENSIONS	Step	Branching Action
SUB-VARIATIONS		Branching Action

SE course – Deliverables | 2018-2019

RELATED INFORMATION	Invia i dati
Priority:	Questo UC è di vitale importanza, in quanto non potremmo avere dati salvati su database se questo non funzionasse.
Performance	Il nostro UC deve funzionare 24 ore su 24
Frequency	Invio dei dati ogni minuto / in caso di errore ogni 30 secondi
Channels to actors	I dati sono salvati in un database
OPEN ISSUES	Malfunzionamento (mancato invio del dato)
Due Date	Ogni minuto o ogni trenta secondi, in base alla priorità

USE CASE #	Consultazione dei dati	
Goal in Context	Visualizzazione dei dati sulla dashboard	
Scope & Level	Sistema principale, attività principale	
Preconditions	Dashboard acceso e funzionante.	
Success End Condition	Dashboard accesa con la visualizzazione di tutti i dati	
Failed End Condition	Dashboard spenta o con la visualizzazione non corretta di tutti i dati	
Primary	Responsabile	
Trigger	Consultazione e visualizzazione dei dati	
DESCRIPTION	Step	Action
	1	Arrivo dei dati sulla dashboard
	2	Consultazione dati
EXTENSIONS	Step	Branching Action
SUB-VARIATIONS		Branching Action

SE course – Deliverables | 2018-2019

RELATED INFORMATION	Consultazione dei dati relativi all'area	
Priority:	Questo UC è molto importante per il monitoraggio dell'ambiente	
Performance	Il nostro UC deve funzionare 24 ore su 24	
Frequency	Aggiornamento continuo della schermata	
Channels to actors	I dati sono salvati in un database consultati al responsabile	
USE CASE #	Riconoscimento	
Goal in Context	Registrazione dell'operatore sulla propria area di controllo	
Scope & Level	Sistema principale	
Preconditions	Dashboard acceso e funzionanti	
Success End Condition	Sistemi accessi e 'codice utente' riconosciuto	
Failed End Condition	Codice utente non riconosciuto o monitoraggio non attivo	
Primary,	Responsabile	
Trigger	Consultazione e visualizzazione dei dati sulla dashboard	
DESCRIPTION	Step	Action
	1	Inserimento del codice dell'operatore
	2	Riconoscimento del codice
EXTENSIONS	Step	Branching Action
SUB-VARIATIONS		Branching Action

RELATED INFORMATION	Riconoscimento
Priority:	Questo UC è importante per l'assegnamento delle responsabilità
Performance	Il nostro UC deve funzionare 24 ore su 24

A1.1 GUI Requirements

Il progetto sarà fornito di tre tipi di GUI diverse, ovvero di tre interfacce utente su tre livelli differenti.

A livello più “basso” abbiamo una dashboard dedicata ai responsabili di area, che sarà suddivisa così: in alto a destra avremo l'autenticazione per il responsabile, di seguito, avremo le stanze situate all'interno dell'area di riferimento, affiancate da tutti i valori (di temperatura, umidità e pressione) ad esse corrispondenti.

VERDE sarà l'icona con nessuna anomalia al suo interno.

ROSSO sarà l'icona che segnala un'anomalia all'interno di essa;

ARANCIONE sarà l'icona chiusa che segnala un'anomalia all'interno di essa (anomali differente da quella segnalata di rosso);

Sarà possibile tornare alla dashboard di area attraverso un pulsante “indietro” situato in alto a destra.

Indietro	UFFICI					
	T1	T2	U1	U2	P1	P2
STANZA 1	22	22	42	42	16	16
STANZA 2	20	20	45	45	22	22
STANZA 3	22	22	47	47	25	25
STANZA 4	29	29	43	43	25	25
STANZA 5	20	20	50	50	22	22
BAGNO	21	21	48	48	22	22
SALA DI ATTESA	22	22	51	51	23	23
CORRIDOIO	22	16	70	50	22	22

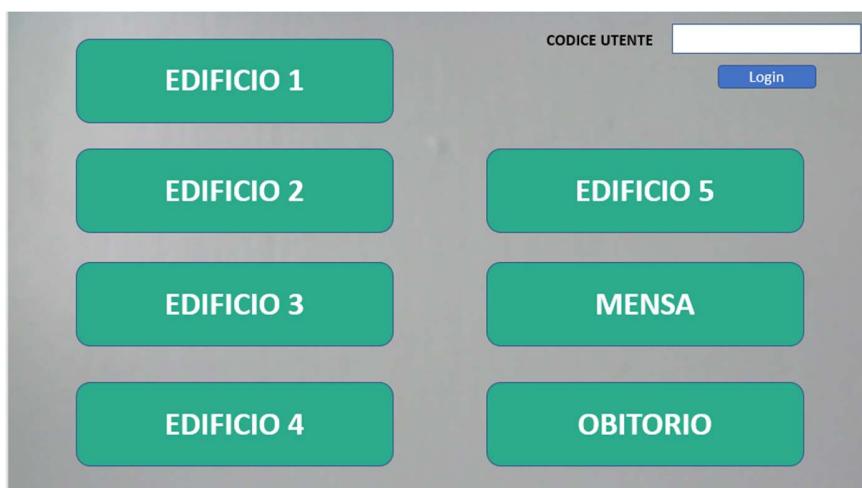
SE course – Deliverables

2018-2019

Nel livello “intermedio”, dedicato alla dashboard di edificio, avremo un’interfaccia basilare che ci permetterà, come nelle altre, di fare l’identificazione utente in alto a destra, di seguito sarà riportato il nome dell’edificio e di conseguenza i piani inerenti ad esso, che saranno selezionabili. Non sarà possibile selezionare più piani contemporaneamente. Sarà possibile tornare alla dashboard amministrativa attraverso un pulsante “indietro” situato in alto a destra.



Infine, nel livello più alto troveremo la dashboard dedicata agli amministratori; oltre il riconoscimento utente, nella schermata avremmo tutti gli edifici controllati da quest’area amministrativa, avendo la possibilità di selezionarli, uno per volta, per entrare nell’edificio interessato e quindi consultare la dashboard di edificio.



A1.2 Business Logic Requirements

L'elaborazione dei dati all'interno del sistema, avverrà nel seguente modo:

Tutti i dati relativi al nostro sistema saranno salvati nel DB. Prima di essere visualizzati, dovranno essere presi dal DB e controllati dalla componente computazionale che sarà implementata, per poi essere restituiti secondo la loro validità e quindi visualizzati nel modo giusto. Per dati si intendono tutti i valori salvati all'interno del nostro DB, quindi anche le informazioni relative agli utenti, che a loro volta saranno controllate prima di validare l'accesso al sistema da parte di un utente. Per quanto riguarda gli altri vincoli riguardanti i dati, ad esempio l'unicità del codice dell'utente, saranno previsti già all'interno della struttura del nostro DB.

A1.3 DB Requirements

È stato realizzato un database MYSQL: un database di tipo relazionale, cioè che organizza i dati in maniera tabellare e usa il linguaggio SQL per operare su essi.

Per ogni transazione, il DBMS (Database Management System) deve garantire le seguenti proprietà:

- ATOMICITA': il DBMS deve garantire che ogni transazione venga eseguita come un tutt'uno. Ovvero una transazione è un'unità di elaborazione.
- CONSISTENZA: ogni transazione lascia il DB in uno stato consistente. Il DBMS garantisce che nessuno dei vincoli di integrità venga violato.
- ISOLAMENTO: una transazione viene eseguita indipendentemente dalle altre. Se più transazioni eseguono in concorrenza, il DBMS garantisce che l'effetto netto è equivalente a quello di una qualche esecuzione sequenziale delle stesse.
- DURABILITÀ: gli effetti di una transazione che ha terminato correttamente la sua esecuzione devono essere persistenti nel tempo. Il DBMS deve proteggere il DB a fronte di guasti.

A.2 Non Functional Requirements

Il sistema deve essere facile da usare, la dashboard deve avere un'interfaccia semplice e si interagisce con essa attraverso un click per aprire la sezione interessata al controllo.

(USABILITY)

Il sistema deve funzionare H24, 7 giorni su 7, per poter permettere in ogni momento il monitoraggio dell'ambiente. **(RELIABILITY)**

- Il sistema deve essere operativo e funzionante al 99%, con un margine di errore che potrebbe presentarsi in caso di un temporaneo rallentamento della rete. (**AVAILABILITY**)
- I dati forniti devono essere accurati e precisi, in termini di misurazione. (**ACCURACY**)

Il sistema deve essere perfettamente sincronizzato con i sensori in modo da far comparire sulla dashboard i dati aggiornati, non appena il sensore manda il segnale di aggiornamento. (**PERFORMANCE**)

- Il sistema può soddisfare l'accesso di 50 o più utenti e un numero di transazioni pari a 150000 simultaneamente. (**CAPACITY**)

E' previsto un server di "supporto" in modo da garantire il corretto funzionamento anche in caso uno dei server di edificio abbia qualche guasto, ed è prevista inoltre una base di dati di "backup" che abbia memorizzati tutti i dati del nostro sistema, in modo da non avere perdita di dati qualora ci fosse qualche problema ai database principali di ogni edificio. (**FAULT TOLERANCE**)

Per quanto riguarda la gestione dei dati sarà suddivisa nel seguente modo.

Per le aree ad alta priorità:

In caso di dati errati: Aggiornamenti dell'invio dei dati ogni 30 secondi.

Tolleranza:
Temperatura: +/- 1°;
Pressione: +/- 5%;
Umidità: +/- 10%;

I

n caso di mancanza di un segnale, oppure, due valori fuori norma di tipi diversi, o un valore ambientale fuori norma, il numero corrispondente (o la casella) avrà colore **ARANCIONE**.

In caso di mancanza di due o più segnali, o, due valori fuori norma dello stesso tipo, oppure tre o più valori fuori norma di tipi diversi, il numero corrispondente (o la casella) avrà colore **ROSSO**.

(Per quanto riguarda quest'ultimo caso sarà previsto immediatamente l'invio di un messaggio d'allarme sul dispositivo portatile in quel momento riconosciuto).

Per le aree a bassa priorità:

In caso di dati errati: Aggiornamenti dell'invio dei dati ogni minuto.

Tolleranza:
Temperatura: +/- 5°;
Pressione: +/- 10%;

Umidità: +/- 10%;

In caso di mancanza di un segnale oppure un valore fuori norma, il numero corrispondente (o la casella) avrà colore **ARANCIONE**.

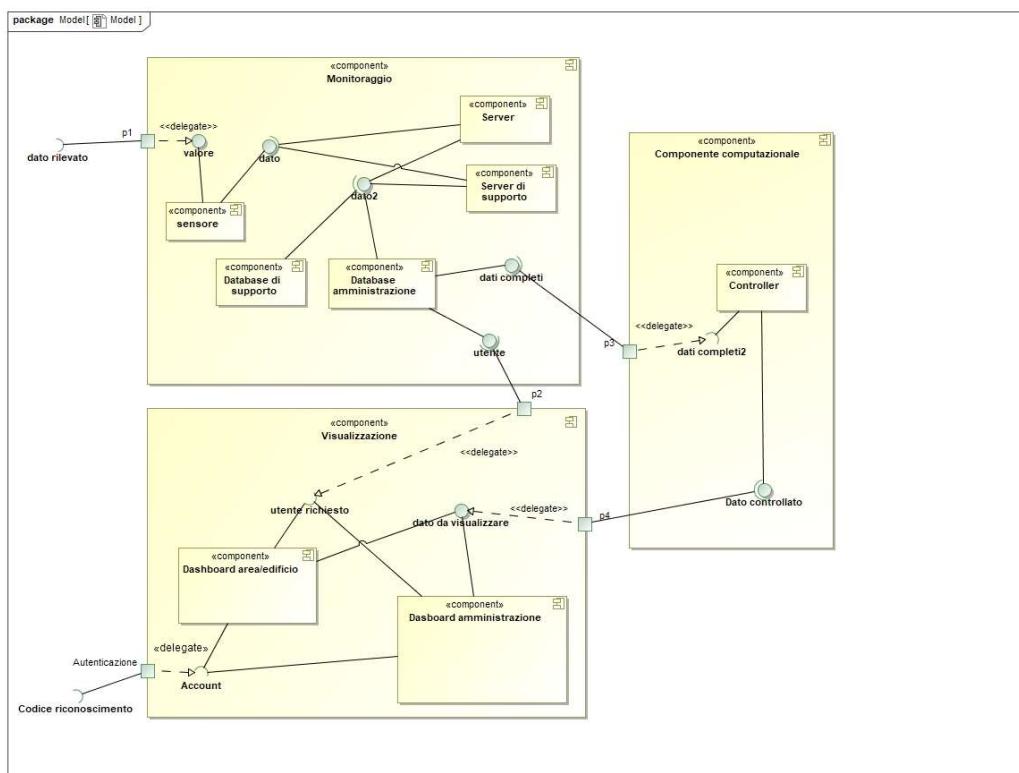
In caso di mancanza di due o più segnali oppure due o più valori fuori norma il numero corrispondente (o la casella) avrà colore **ROSSO**.

In tutti gli altri casi, per entrambe le priorità, il numero corrispondente avrà colore **VERDE**.

Sono adottate misure di sicurezza tali da garantire che nessun utente malintenzionato possa intromettersi nel sistema ad esempio creando falsi allarmi o facendo in modo che qualche messaggio di allarme non arrivi al responsabile. (**SECURITY**)

B. Software Architecture

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



Descrizione Component Diagram

L'architettura prevede l'utilizzo di tre principali componenti:

- **Monitoraggio**, inteso come la parte statica del sistema e sul come i dati, rilevati dai sensori, vengono inviati tramite server;
- **Visualizzazione**, intesa come la parte software dove risiedono i dati rilevati nella precedente componente nei vari database, e su come essi, vengono visualizzati nelle dashboard;
- **Componente computazionale**, intesa come parte di calcolo e di controllo del sistema per quanto riguarda la correttezza dei valori rilevati dai sensori.

Componente Monitoraggio

Questa componente è composta a sua volta, da altre componenti: si parte dal singolo **sensore** che contiene un **valore**, ovvero un **dato rilevato** da esso che “entra” nel sistema tramite porta **p1** (input).

Questo **dato** viene offerto al **Server**, visto come componente che richiederà tali dati (**dato**) forniti, appunto, dal sensore. In caso di malfunzionamento del Server, viene inviato lo stesso valore alla componente **Server di supporto**. In entrambi i casi, i Server, faranno da tramite per l'invio dei dati (**dato2**) alle componenti **Database amministrazione** e **Database di supporto**.

Componente Computazionale

Una volta inviati i dati ai Database, essi offriranno un'interfaccia (**dati completi**) che sarà, tramite porta **p3**, inviata alla componente **Controller**, la quale si aspetta, per l'appunto, dei dati da elaborare (**dati completi2**). Questa componente si occupa di controllare i dati ricevuti, per determinare (tramite colorazione sulla dashboard), la correttezza di questi ultimi. Restituirà quindi, il **dato controllato** che, tramite porta **p4** entrerà nella componente Visualizzazione e potrà essere visualizzato dalle dashboard.

Componente Visualizzazione

In questa componente, è previsto il riconoscimento dell'utente che farà il login nella dashboard tramite **codice di riconoscimento**. Una volta inserite le credenziali, verrà eseguita l'**autenticazione** (anche nome della porta) e l'utente sarà identificato dalla componente **Dashboard amministrazione** o **Dashboard area/edificio**, dopo che queste

ultime avranno consultato la componente Database amministrazione per vedere se quel codice è effettivamente registrato (ciò avviene tramite interfaccia **utente richiesto** che si aspettano le dashboard, l'interfaccia **utente** sarà offerta dal database e manda tramite porta **p2** alla componente Visualizzazione) e quindi potrà essere consentito l'accesso.

Entrambe le componenti, una volta ricevuta la correttezza del dato dal Controller, offrono l'interfaccia **dato da visualizzare**, così sarà possibile consultare i dati del database e avere quindi, una visione completa di tutte le aree appartenenti ai singoli edifici.

Sono state spostate le componenti Database amministrazione e Database di supporto, per avere una visione più chiara di quella che è l'architettura del sistema.

Fondamentalmente, ciò che contengono i database sono più che altro dati statici che verranno analizzati dalla componente Computazionale per poi essere inviati alle Dashboard e in questo modo, visualizzarli. Se entrambe le componenti, Database e Dashboard, fossero nella componente principale Visualizzazione, si sarebbe passati dalla componente Monitoraggio a quella Visualizzazione per poi passare alla componente Computazionale e di nuovo a quella Visualizzazione: questo perché la componente Computazionale prende per l'appunto i dati dalla componente Database.

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

Descrizione Sequence Diagram

Abbiamo diviso i Sequence Diagram, facendo riferimento agli Use Cases.

Invio dati:

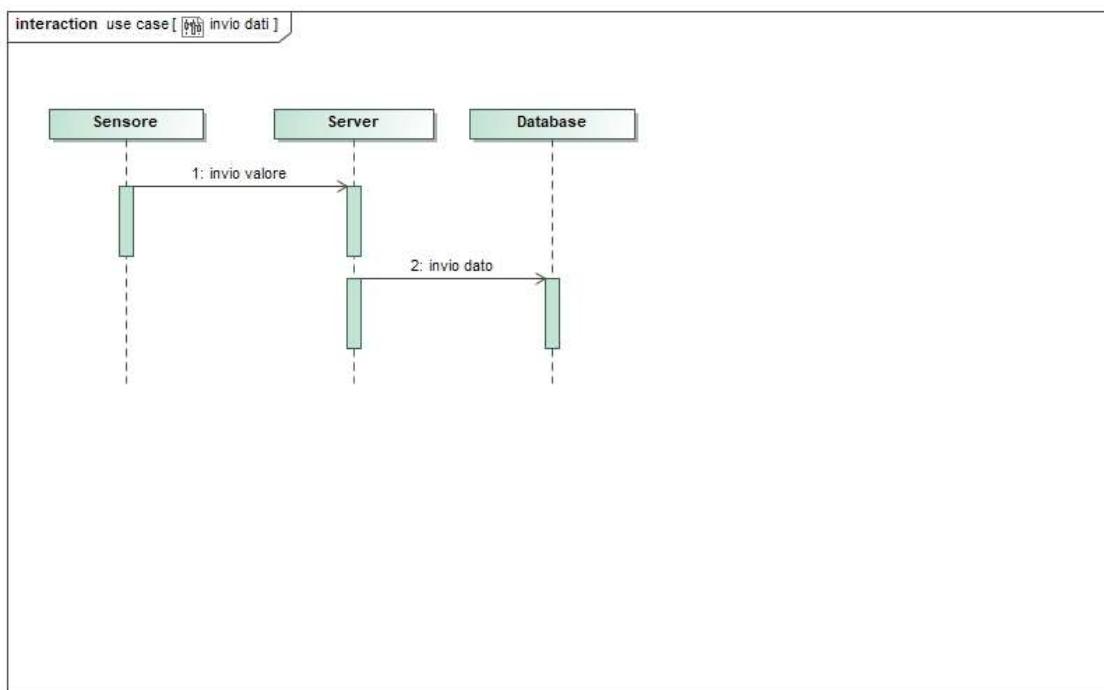
In questo sequence, la lifeline **Sensore** , tramite chiamata asincrona e tramite messaggio **invio valore** , invia il valore del dato alla lifeline **Server**, la quale provvederà all'invio alla lifeline **Database**. Per Server si intende anche il server di supporto che entrerà in funzione soltanto nel caso in cui ci sia un malfunzionamento del Server “standard”.

Controllo/visualizzazione :

La lifeline **Controller**, tramite chiamata sincrona e messaggio **richiesta dato**, chiede i dati al Database, che successivamente, tramite messaggio **invio dato** verranno inviati ad essa, che si occuperà di elaborare i dati tramite **elaborazione dato**. Infine, invierà tramite chiamata asincrona, i dati elaborati alla **Dashboard** mediante messaggio **invio dato controllato**.

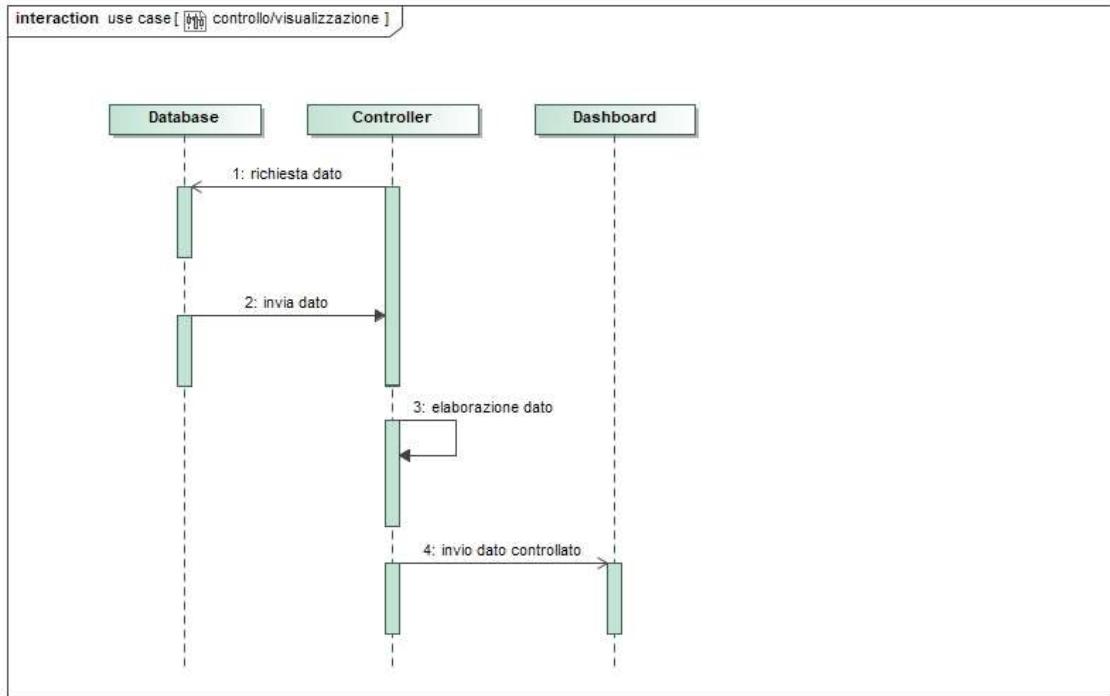
Riconoscimento:

Il **responsabile** (lifeline), tramite codice, si identificherà con il sistema nella Dashboard con una chiamata sincrona. Essa (lifeline), si metterà in comunicazione tramite messaggio **richiesta accesso(codice)** con la lifeline **Controller**, quest'ultimo invierà il codice al Database, che ricercherà al suo interno il codice passato (**ricerca(codice)**), e invierà infine alla dashboard una risposta con il corrispettivo esito (**risposta(esito)**). Quest'ultima si interfacerà con il responsabile tramite messaggio **login(esito)** e, a seconda di esso, ovvero positivo o negativo, avverrà la consultazione della dashboard in base alla propria area di competenza (**consultazione**), oppure verrà mostrato un messaggio di errore per codice non riconosciuto(**messaggio_errore**). Per quanto riguarda gli Use cases Consultazione e Gestione Sistema: il primo è stato inserito all'interno del Sequence Riconoscimento come una possibile alternativa dell'esito del login, invece gestione sistema non abbiamo ritenuto fosse significativo a livello di sequence diagram.



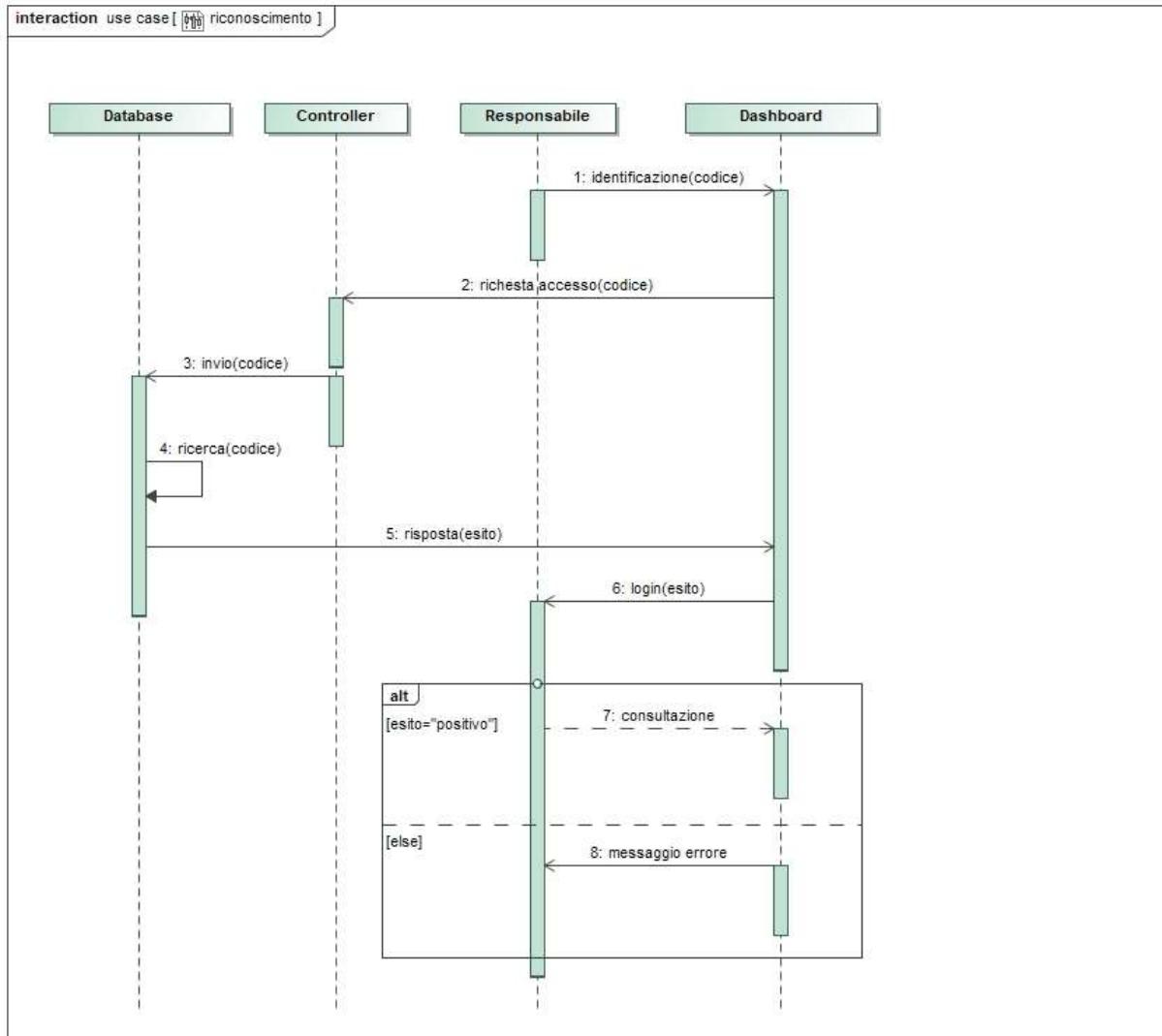
SE course – Deliverables

2018-2019

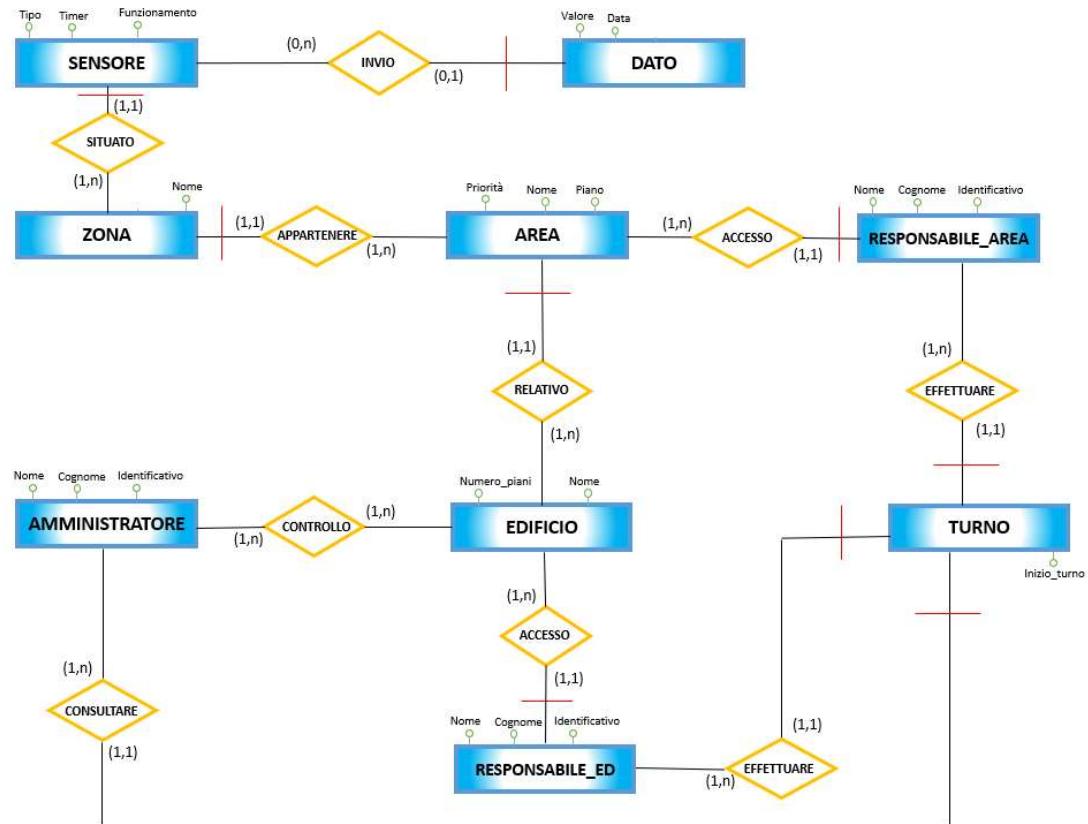


SE course – Deliverables

**2018-
2019**



C. ER Design



TABELLE

SENSORE {ID, ID_ZONA, tipo, timer, funzionamento}
ZONA {ID, ID_AREA, nome}
AREA {ID, ID_EDIFICIO, nome, piano, priorità}
EDIFICIO {ID, nome, numero_piani}
RESPONSABILE_AREA {ID, ID_AREA, ID_EDIFICIO, nome, cognome, identificativo}
DATO {ID, ID_SENSORE, valore, data}
AMMINISTRATORE {ID, nome, cognome, identificativo}
RESPONSABILE_EDIFICO {ID, ID_EDIFICIO, nome, cognome, identificativo}
TURNO {ID, ID_AMMINISTRATORE, ID_RESPONSABILE_EDIFICO,
ID_RESPONSABILE_AREA, inizio_turno}
AMMINISTRAZIONE {ID, ID_AMMINISTRATORE, ID_EDIFICIO}

Ogni **sensore**, caratterizzato da un proprio ID (univoco) e da un timer (che rappresenta ogni quanto i dati verranno inviati), sarà situato in una determinata **zona** appartenente ad un' **area** di un preciso **edificio**.

I **responsabili di area** potranno accedere alle sole aree ad essi designate, i **responsabili di edificio**, invece, oltre all'accesso ai rispettivi edifici, hanno la visione anche su tutte le aree all'interno di essi.

L'**amministratore**, invece, avrà l'accesso a qualsiasi dashboard presente all'interno del sistema : potrà quindi accedere, tramite le proprie credenziali, a qualsiasi postazione e consultare qualsiasi tipo di dato fornito dai sensori.

Ogni volta che un responsabile accede al sistema, verrà registrato l'inizio del **turno**.

LEGENDA ATTRIBUTI

SENSORE	
ATTRIBUTO	DESCRIZIONE
ID	Chiave primaria dell'entità
ID_ZONA	Chiave esterna
TIPO	Tipo di sensore (esempio: temperatura)
TIMER	Ogni quanto inviare aggiornamenti dati (valori rilevati dai sensori)
FUNZIONAMENTO	Se il sensore funziona o meno (1-0)

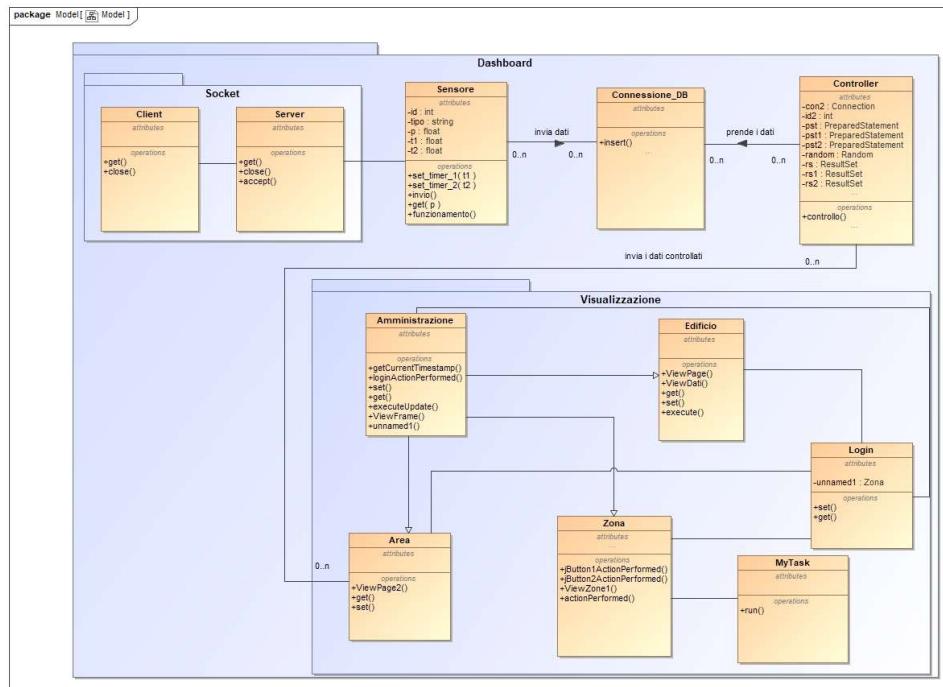
ZONA	
ATTRIBUTO	DESCRIZIONE
ID	Chiave primaria dell'entità
ID_AREA	Chiave esterna
NOME	Nome della zona (esempio: bagno, corridoio , stanza)

AREA		RESPONSABILE AREA	
ATTRIBUTO	DESCRIZIONE	ATTRIBUTO	DESCRIZIONE
ID	Chiave primaria dell'entità	ID	Chiave primaria dell'entità
ID_EDIFICIO	Chiave esterna	ID_AREA	Chiave esterna
NOME	Nome dell'area (esempio: pediatria)	ID_EDIFICIO	Chiave esterna
PIANO	Piano dell'edificio in cui è situata l'area	NOME	Nome del responsabile
PRIORITA	Priorità dell'area : alta o bassa	COGNOME	Cognome del responsabile

EDIFICIO		DATO	
ATTRIBUTO	DESCRIZIONE	ATTRIBUTO	DESCRIZIONE
ID	Chiave primaria dell'entità	ID	Chiave primaria dell'entità
NOME	Nome dell'edificio (esempio: mensa)	ID_SENSORE	Chiave esterna
NUMERO PIANI	Numero di piani che compongono l'edificio	VALORE	Valore del dato (esempio : 30°)
		DATA	Timestamp del rilevamento del dato

AMMINISTRATORE		RESPONSABILE EDIFICIO	
ATTRIBUTO	DESCRIZIONE	ATTRIBUTO	DESCRIZIONE
ID	Chiave primaria dell'entità	ID	Chiave primaria dell'entità
NOME	Nome dell'amministratore	ID_EDIFICIO	Chiave esterna
COGNOME	Cognome dell'amministratore	NOME	Nome del responsabile
IDENTIFICATIVO	Codice identificativo dell'amministratore	COGNOME	Cognome del responsabile
		IDENTIFICATIVO	Codice identificativo del responsabile
TURNO		AMMINISTRAZIONE	
ATTRIBUTO	DESCRIZIONE	ATTRIBUTO	DESCRIZIONE
ID	Chiave primaria dell'entità	ID	Chiave primaria dell'entità
ID_AMMINISTRATORE	Chiave esterna	ID_AMMINISTRATORE	Chiave esterna
ID_RESPONSABILE_AREA	Chiave esterna	ID_RESPONSABILE_EDIFICIO	Chiave esterna
ID_RESPONSABILE_EDIFICIO	Chiave esterna	INIZIO_TURNO	Timestamp dell'inizio del turno dei responsabili e dell'amministratore
		ID_EDIFICIO	Chiave esterna

D. Class Diagram of the implemented System



Descrizione Class Diagram

Abbiamo rappresentato le classi principali che caratterizzano il nostro sistema. Ognuna di esse è caratterizzata dal proprio nome , una lista di attributi e un'altra contenente i metodi. Tutte le classi sono contenute all'interno del Package “Dashboard” che sta a rappresentare l'intero sistema. Inoltre, vi sono anche altri due Packages : “Visualizzazione” e “Socket” , che contengono le classi che li rappresentano. Le classi sono collegate tra loro attraverso associazioni che possono avere una freccia piena o vuota, e cardinalità. Sulle associazioni è descritta “l'azione” compiuta in modo da esplicitare in che modo le classi interagiscono tra di loro e inoltre le cardinalità stanno ad indicare la molteplicità delle azioni.

Esempio freccia piena: **Sensore** → “invia dati” → **Connessione_DB** con Cardinalità 0..n
Questo esempio sta a rappresentare un Sensore che invia da 0 a n dati al Database .

Esempio freccia vuota: **Amministrazione** → **Edificio | Zona | Area**

In questo caso , la Classe Amministrazione è una Superclasse e le classi Edificio/Zona/Area sono delle sottoclassi che estendono quest'ultima.

E. Design Decisions

- **Perché la scelta di un ambiente ospedaliero?**

La prima decisione comune presa è stata quella di contestualizzare l'ambiente di monitoraggio e riportarlo in una realtà ospedaliera, poiché abbiamo pensato di rendere la specifica il più reale possibile e l'ospedale ci è parso un lungo consono al monitoraggio.

- **Com'è suddiviso il sistema?**

Il sistema è suddiviso in tre diversi “livelli” di monitoraggio: area, edificio e amministrazione. In base al livello nel quale ci troviamo , le rispettive dashboard permetteranno la visualizzazione dei dati relativi ad esso.

- **Come trattiamo la sensibilità dei dati e la loro protezione?**

Non trattandosi di dati molto sensibili non saranno previste protezioni specifiche per quanto riguarda la loro visualizzazione, ma essendo un sistema in cui, si dovranno proteggere tali dati in modo da evitare intrusioni che potrebbero causare falsi allarmi, sarà prevista l'installazione di un firewall e un riconoscimento per operare sul sistema tramite login.

- **Come potrà essere effettuato il login?**

Abbiamo concordato che il login sarà situato in alto a destra della dashboard, senza ostruire la visualizzazione dei dati che saranno visibili in ogni momento.

- **Chi può fare l'accesso alla dashboard?**

L'accesso è consentito ai responsabili, e sarà abilitato soltanto se il codice personale inserito è presente nel database, nella tabella relativa agli utenti, che inoltre conterrà i dati anagrafici di ogni utente e la sua zona di competenza (area, edificio, amministrazione).

- **Come gestire il malfunzionamento di un server?**

Abbiamo pensato di mettere un server di supporto sul quale appoggeranno tutte le dashboard in caso di malfunzionamento del server “standard”.

- **Come gestire la perdita di dati?**

Per quanto riguarda il database, ce ne sarà uno di supporto. I sensori, oltre ad inviare i dati al database di amministrazione (e quindi a quello “standard”), invieranno dati anche a quello di supporto (ridondanza volontaria).

- **Cosa contiene il database amministrativo?**

Il database amministrativo contiene tutti i dati inviati da tutti i sensori del nostro sistema. La consultazione dell'intero database è permessa soltanto a livello amministrativo , poiché per quanto riguarda le aree e gli edifici , potranno consultare soltanto le parti del database relativi all'area/edificio interessati.

- **Per quanto restano i dati nel database di amministrazione?**

Ogni 24h verranno salvati i dati del database, in un hard disk da un 1 terabyte, raggruppati in fogli Excel e successivamente verranno cancellati nella base di dati.

- **In caso di valori fuori norma, viene subito visualizzato il colore rosso di allarme?**

No, è stato stabilito un grado di tolleranza per ogni tipo di sensore; facendo riferimento anche al tipo di area (bassa o alta priorità).

- **Come identificare nel sistema, l'importanza di un'area rispetto a un'altra?**

Abbiamo suddiviso le aree per alta e bassa priorità. La differenza consiste sostanzialmente, in una diversa gestione di tolleranze (per le basse sarà maggiore, per le alte minore), la visualizzazione dei valori fuori norma e gli allarmi.

- **Come viene rilevato il funzionamento del sensore?**

Se un sensore rileva i valori allora vuol dire che esso funziona, ma non è garantita la correttezza del valore rilevato. Il suo non funzionamento comporterà una non ricezione dei dati sulla dashboard.

- **Perché implementare un unico database, oltre quello di supporto?**

Abbiamo preso coscienza che un unico database, affiancato da quello di supporto è sufficiente per garantire e supportare la quantità di dati di questo sistema, rendendolo più ottimizzato in quanto ogni sensore invierà i dati direttamente al DB di amministrazione.

- **Perché implementare un unico server, oltre quello di supporto?**

Un server unico sarà sufficiente per effettuare tutte le comunicazioni. In caso di mancato funzionamento verrà utilizzato quello di supporto.

- **Come viene garantita la correttezza e non dei dati?**

Ci sarà una componente ‘Controller’ che si occuperà di elaborare il valore dei dati forniti dal database. Su questi ultimi verrà fatta una verifica, ovvero, se i dati esaminati saranno “fuori norma” o meno, e/o in assenza del loro valore, essi, nella dashboard, saranno visualizzati con colori diversi.

Sono previste in totale **7 strutture**.

I 5 edifici raggruppati hanno le medesime caratteristiche, quali :

- 3 piani ciascuno
- Ogni piano ospita almeno 3 reparti

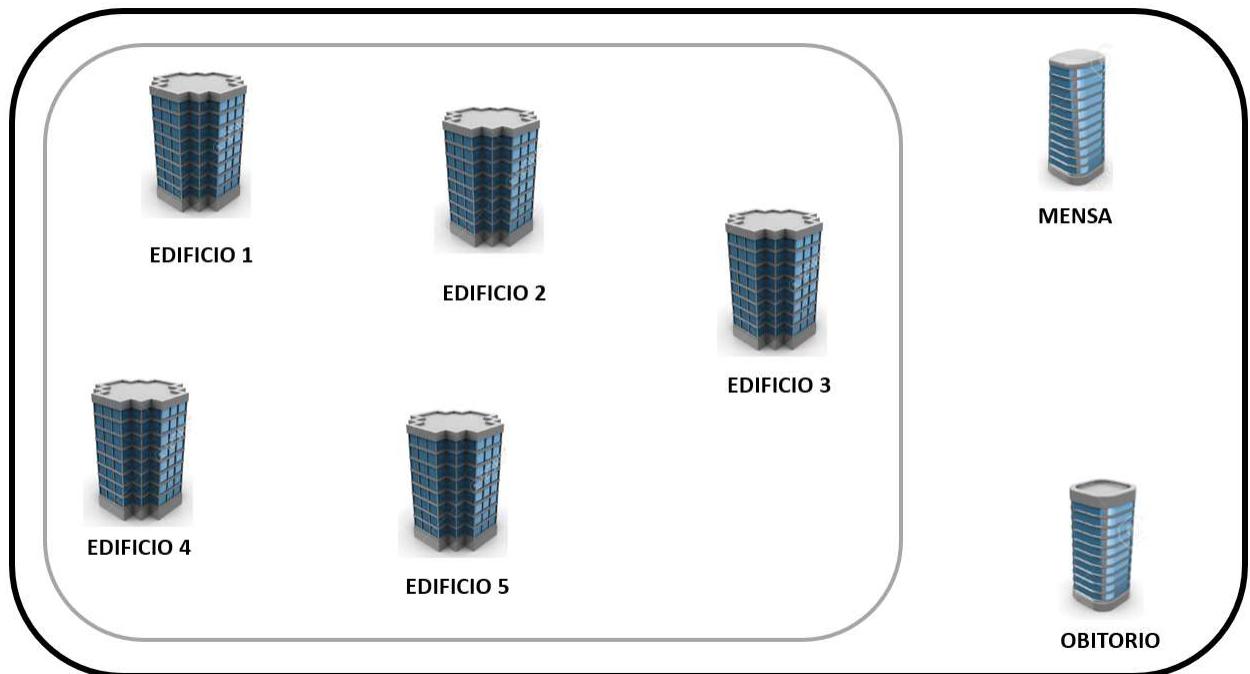
Inoltre, per ogni piano sono previste zone di passaggio (corridoi, scale) e zone comuni (sale di attesa, bagni comuni, etc.).

Dalla descrizione sono esclusi i due edifici

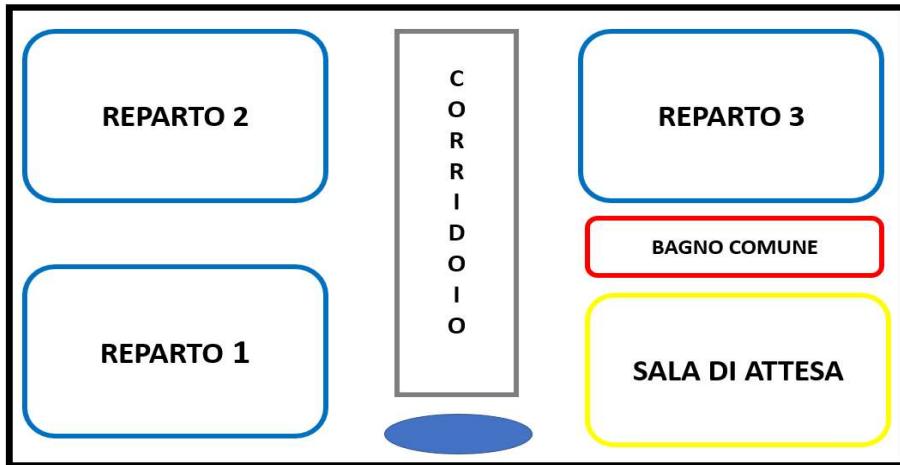
- Mensa
- Obitorio

in quanto considerate strutture a sé stanti caratterizzati da un unico piano (più le zone comuni).

VISTA GENERALE OSPEDALE



VISTA DI UN PIANO DI UN EDIFICIO (ECCETTO IL PIANO TERRA)



Planimetria di un generico piano (escluso il piano terra)

DISPOSIZIONE REPARTI ALL'INTERNO DEGLI EDIFICI

EDIFICIO 1

- **Piano terra :** Uffici amministrativi, punto informazioni , centro prelievi
- **1 piano :** medicina interna, sala operatoria, gastroenterologia
- **2 piano :** magazzino, laboratorio analisi e centro sterilizzazione

EDIFICIO 2

- **Piano terra :** maxillofacciale, oculistica, dermatologia
- **1 piano :** sala operatoria, cardiologia, rianimazione, terapia intensiva
- **2 piano :** magazzino, centro sterilizzazione, neurochirurgia

EDIFICIO 3

- **Piano terra :** pediatria, cardiologia pediatrica, pronto soccorso pediatrico
- **1 piano :** ginecologia, sala parto, terapia intensiva neonatale
- **2 piano :** magazzino, centro sterilizzazione, sala operatoria

EDIFICIO 4

- **Piano terra** : pronto soccorso, punto informazioni, radiologia
- **1 piano** : ortopedia, oncologia, sala operatoria
- **2 piano** : magazzino, centro sterilizzazione, urologia

EDIFICIO 5

- **Piano terra** : geriatria, pneumologia, neurologia
- **1 piano**: sala operatoria, lungodegenza, centro dialisi
- **2 piano**: magazzino, centro sterilizzazione, psichiatria

MENSA

- **Piano terra** : zona cucina e zona consumazione pasti

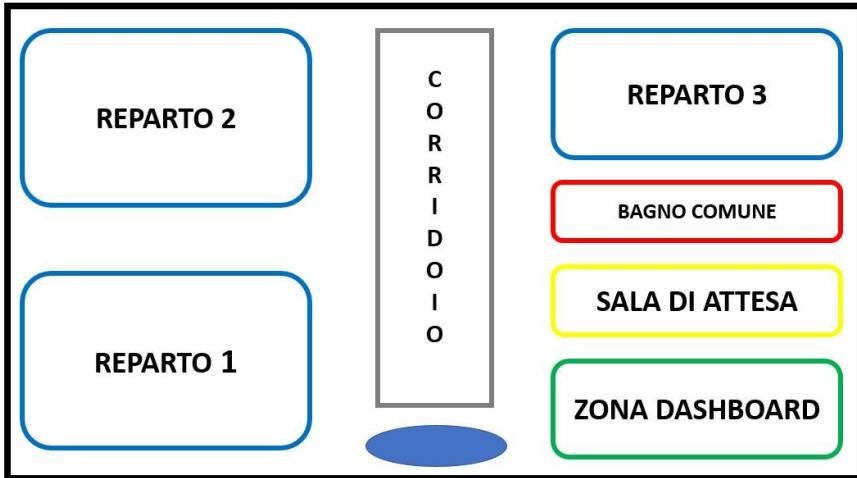
OBITORIO

- **Piano terra** : zone comuni

Per ogni edificio, al piano terra, sarà previsto un ufficio dedicato alla dashboard di edificio.

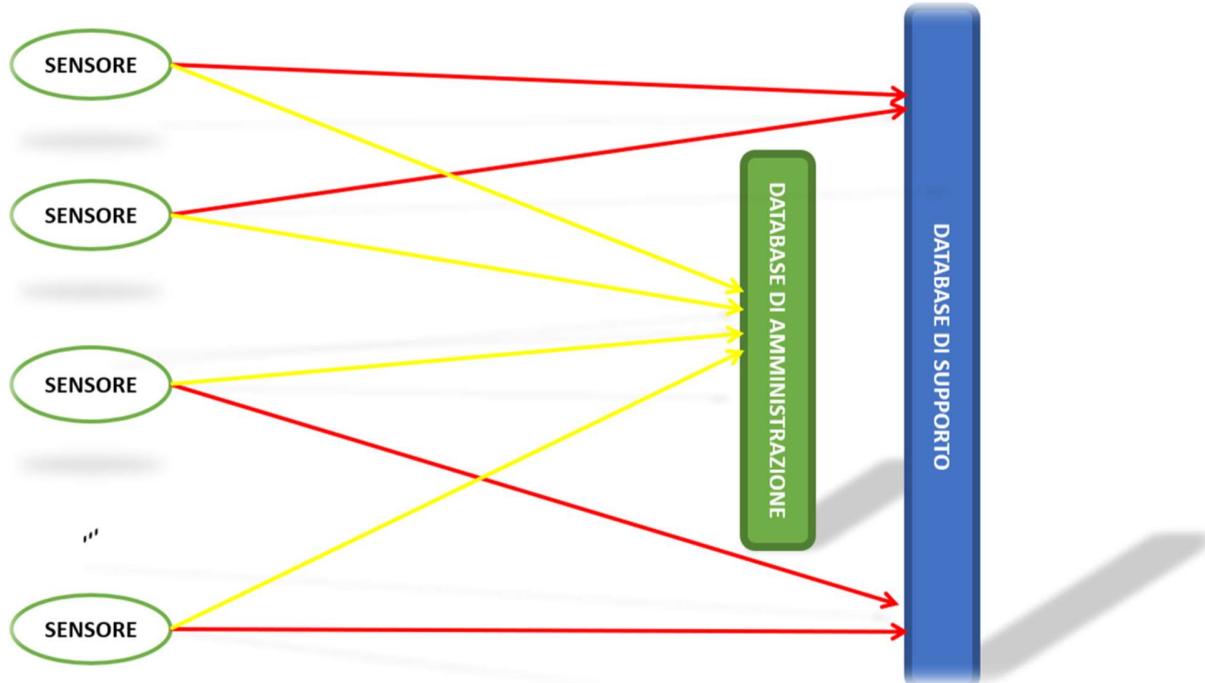
Inoltre, per ogni ufficio sopra citato saranno installati 2 sensori di temperatura.
Va specificato che le dashboard di area e di edificio svolgono la funzione di sola consultazione dei dati e non di invio.

VISTA DEL PIANO TERRA DI OGNI EDIFICIO



Piano terra di ogni edificio.
La zona dashboard ospita la dashboard per la visualizzazione dati relativi all'edificio.

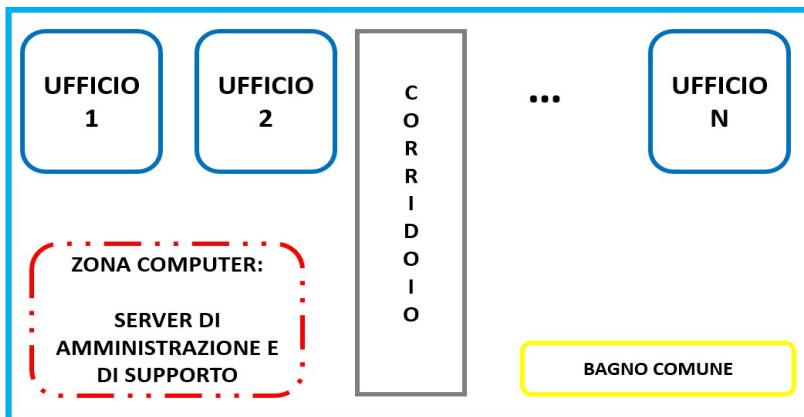
IN DETTAGLIO ..



I sensori distribuiti all'interno di ogni struttura invieranno i dati al database di amministrazione e a quello di supporto.

Ogni area e ogni edificio avranno una propria dashboard, la quale consentirà la visualizzazione dei propri dati (presi dal database).

VISTA REPARTO AMMINISTRAZIONE - EDIFICO N° 1 - PIANO TERRA



Piano terra edificio 1 : reparto uffici amministrativi nel quale sono situati i due server.

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

Andiamo ora a vedere come alcuni requisiti, funzionali e non, identificati dal sistema vengono soddisfatti attraverso le decisioni di design prese.

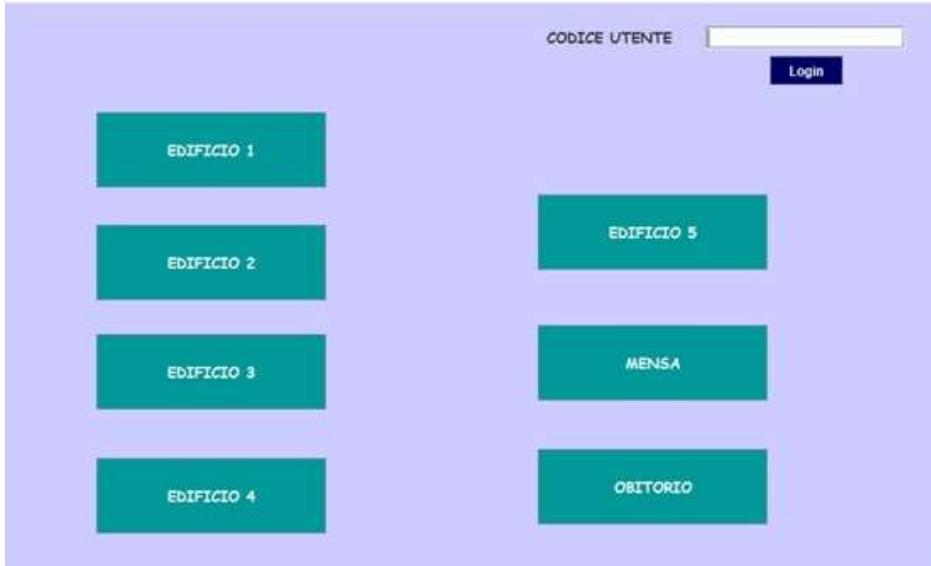
Innanzitutto parlando dei responsabili, otteniamo una differenziazione dei vari tipi di utenti attraverso il database, poiché per ogni tipologia di responsabile è stata creata una differente tabella, collegata univocamente con amministrazione, edificio o area.

In generale il sistema è attivo h24 poiché collegato ad un generatore sempre acceso che, in caso di malfunzionamento, verrà sostituito dal generatore di supporto nei tempi dovuti. Inoltre, non abbiamo previsto nessun vincolo per quanto riguarda la visualizzazione dei dati, che saranno ricevuti dal sistema e visualizzati sulla dashboard in ogni momento. Abbiamo quindi inteso, nel nostro sistema, l'inserimento delle credenziali non come un permesso ad accedere ai dati, bensì un monitoraggio attivo sui turni tenuti dai vari responsabili per tenerne una traccia, infatti a tal proposito abbiamo

SE course – Deliverables | 2018-2019

implementato la tabella **Turno** sul Database, che viene aggiornata ogni qualvolta un responsabile effettua il login.

La dashboard è resa di facile utilizzo grazie all'utilizzo di bottoni, labellati da nomi chiari e che individuano chiaramente la zona, l'area o l'edificio a cui si riferiscono.



Il requisito di **Riconoscimento** viene soddisfatto attraverso la realizzazione di un'apposita spazio in alto a destra sulla dashboard logicamente per la privacy dell'utente il codice inserito sarà protetto da una crittografia MD5 hash e la password non sarà più visibile una volta effettuato l'accesso, al suo posto comparirà un messaggio di **benvenuto**, nel caso di riconoscimento avvenuto positivamente, oppure messaggio di **accesso negato**, in caso contrario.

SE course – Deliverables | 2018-2019



Ed inoltre siamo riusciti ad ottenere in un'area che le stanze vengono visualizzate in **verde**, **arancione** e **rosso**, a seconda che si presenti un errore sulle misurazioni, logicamente poiché i dati che arrivano al nostro sistema sono dati totalmente random, non c'è un'altissima possibilità di avere una o più stanze in un'area con problemi.

SE course – Deliverables | 2018-
2019

UFFICI

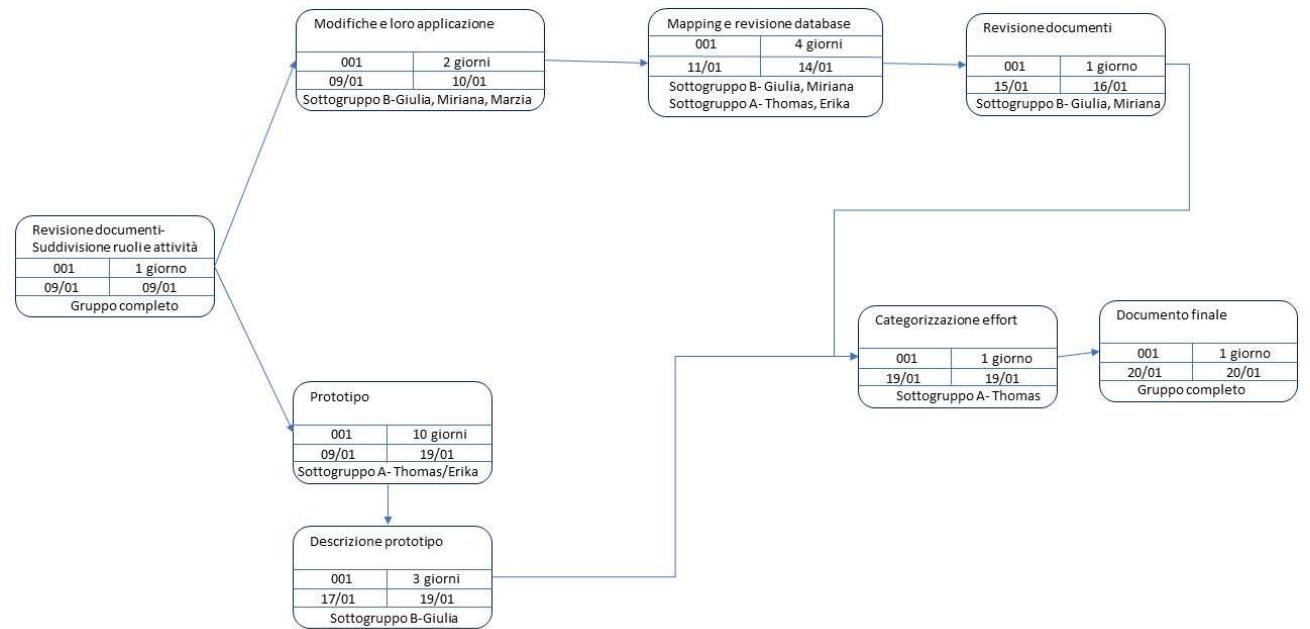
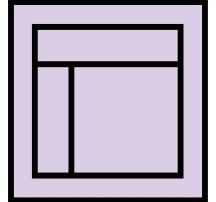
	T1	T2	U1	U2	P1	P2
STANZA 1	-	-	-	-	-	-
STANZA 2	-	-	-	-	-	-
STANZA 3	-	-	-	-	-	-
STANZA 4	-	-	-	-	-	-
STANZA 5	-	-	-	-	-	-
BAGNO	-	-	-	-	-	-
SALA DI ATTESA	-	-	-	-	-	-
CORRIDOIO	-	-	-	-	-	-

UFFICI

	T1	T2	U1	U2	P1	P2
STANZA 1	24°	25°	61%	40%	8%	25%
STANZA 2	28°	29°	50%	42%	9%	20%
STANZA 3	29°	18°	69%	60%	31%	9%
STANZA 4	16°	20°	59%	37%	31%	15%
STANZA 5	17°	22°	60%	60%	15%	21%
BAGNO	22°	22°	44%	69%	30%	16%
SALA DI ATTESA	25°	25°	58%	33%	23%	20%
CORRIDOIO	29°	19°	70%	37%	8%	10%

G. Effort Recording

PERT



Logging

Personal Journal					
Team (number and name): MM-GTE					
Student name: Giulia, Marzia, Thomas, Erika, Miriana Student number: 5 Email: marzia.piccirilli@student.univag.it					
When (Month/Day)	Time spent	Partners <i>to whom report hour were given people hours been working</i>	Brief Description of the performed task	Category	Sub-Category
11 8	3:30 h	5	Comprensione della specifica	Tutti	
11 12	3:30 h	5	Scelta ambito di lavoro e decisione ruoli con PERT dei sotto-gruppi	Tutti	
11 14	3 h	4	Decisioni finali riguardanti il progetto e inizio lavoro	Tutti	
11 16	2h	5	Discussione altre decisioni e continuo del lavoro	Tutti	
11 20	3h	3	Consegna Use Cases, requisiti funzionali, requisiti non funzionali	Sottogruppo A	Thomas, Marzia, Miriana
11 21	3h	2	Documento di design e decisioni per il Component Diagram	Sottogruppo B	Erika, Thomas
11 22	3 h	3	Consegna tabelle di cockburn	Sottogruppo A	Thomas, Marzia, Miriana
11 24	1h	3	Consegna Component Diagram e decisioni per il Sequence Diagram	Sottogruppo B	Erika, Thomas, Giulia
11 25	2h	3	Revisione lavoro sottogruppo A	Sottogruppo A	Thomas, Marzia, Miriana
11 26	2 h	5	Discussione altre decisioni e continuo del lavoro	Tutti	
11 23	3h	5	Revisione di tutto e anticipo categorizzazione effort	Tutti- Sottogruppo B	Erika, Giulia
11 30	7h	4	Compilazione del Template	Tutti	
12 6	2h	5	Discussioni per la seconda parte e sui class diagram	Tutti	
12 12	6h	5	Suddivisione del lavoro, Pert, discussioni e continuo del lavoro	Tutti	
12 17	2:30 h	4	Modifiche progetto	Tutti	
12 18	4:30 h	4	Requisiti GUI- Business Logic e inizio ER Diagram	Tutti	
12 19	3h	4	Requisiti specifici DB, ER Diagram, Class Diagram e inizio Prototipo	Tutti	
12 20	6h	4	Prototipo, completamento requisiti specifici e ER Diagram	Tutti	
12 21	5h	5	Completeramento Prototipo, Design Decision, Mapping, Categorizzazione	Tutti	
1 9 8h		5	Pert, capire le modifiche e applicarle. Revisione documenti e continuo del proto	Tutti	
1 10 7h		5	Discussioni e continuo del lavoro	Tutti	
1 11 6h		4	Aaggiornamento del db e continuo del prototipo	Sottogruppo B- Sottogruppo	Thomas, Erika, Giulia, Miriana
1 14 6h		4	Mapping e prototipo	Sottogruppo B- Sottogruppo	Thomas, Erika, Giulia, Miriana
1 15 8h		4	Prototipo e aggiornamento con revisione dei documenti	Sottoaruppo B- Sottoaruppo	Thomas, Erika, Giulia, Miriana
1 17 8h		3	Prototipo e descrizione	Thomas, Giulia, Marzia	
1 18 8h		2	Prototipo e descrizione	Thomas, Giulia	
1 19 8h		2	Prototipo, descrizione e categorizzazione	Thomas, Giulia	

Categorization

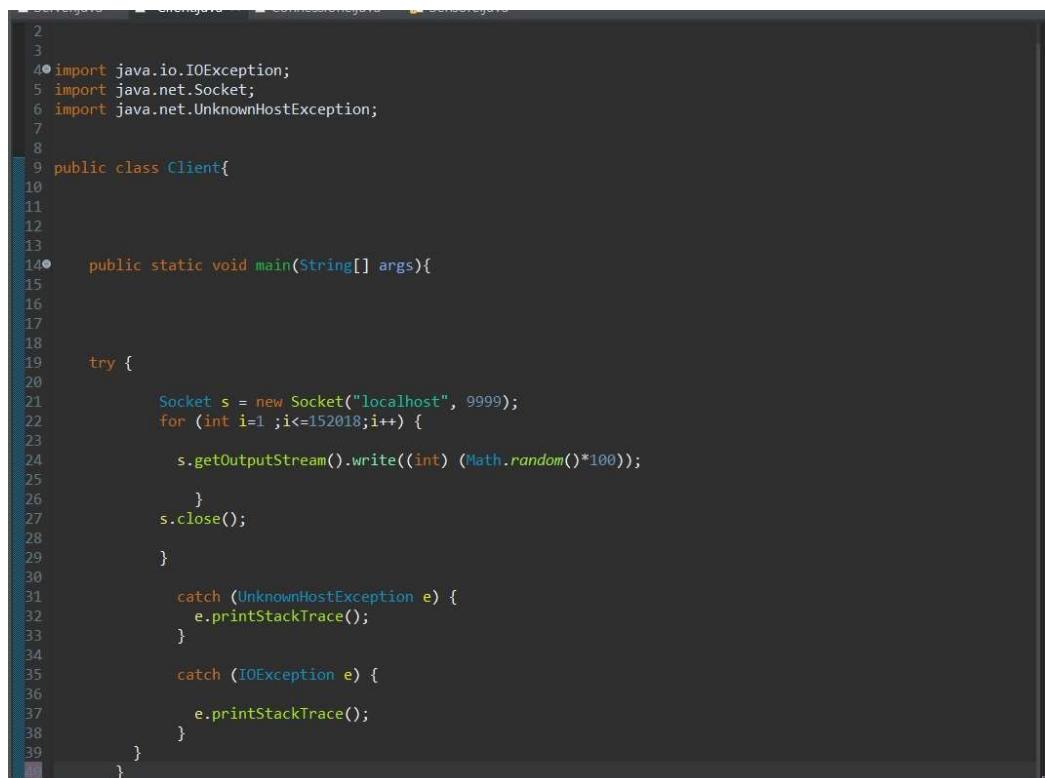
Delivery3	Start-end	Concluso	Componente Gruppo
Organizzazione generale	09-14 Gennaio	si	Tutti
Attività 1.1- Fase decisionale per modifiche da apportare		x	Tutti
Attività 1.2- Modifiche progetto		x	Giulia, Miriana, Marzia
Attività 1.3-Suddivisione del lavoro		x	Tutti
Prototipo v.2	09-19 Gennaio	si	Thomas, Erika, Giulia
Attività 1.1- Fase di studio e realizzazione		x	Thomas, Erika
Attività 1.2- Descrizione prototipo		x	Giulia
Mapping	11-14 Gennaio	si	Giulia, Miriana
Attività 1.1- Fase di studio		x	Giulia, Miriana
Attività 1.2- Documento		x	Giulia, Miriana

Appendix. Code

Ad oggi il nostro sistema non riesce a implementare e a soddisfare tutti i requisiti, funzionali e non.

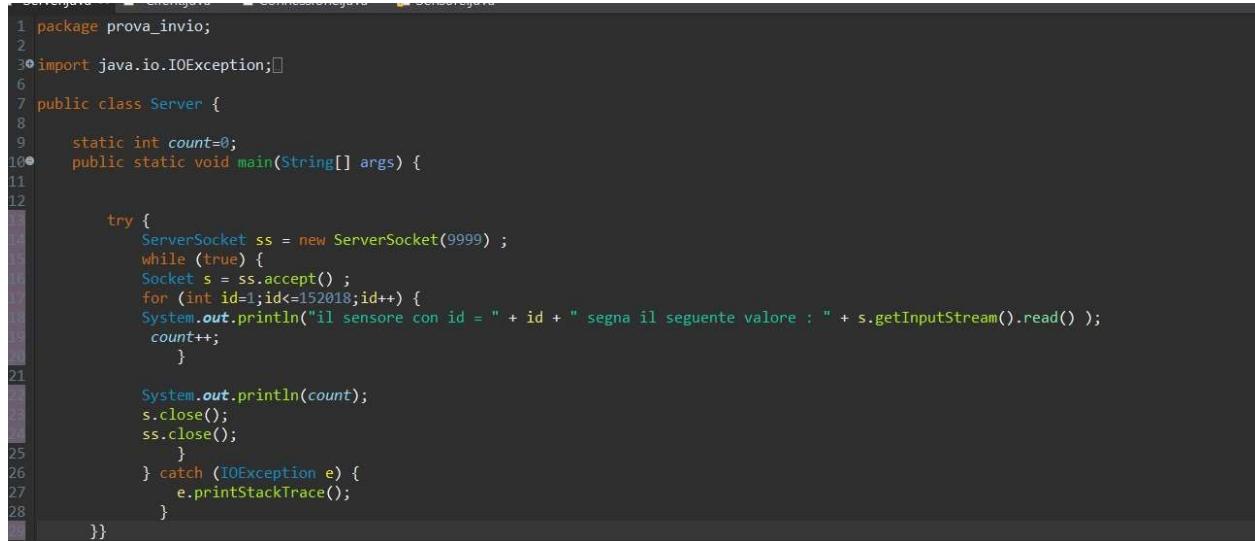
Per avere un risultato nei tempi della consegna, ci siamo concentrati sull'implementazione della dashboard relativa all'**amministrazione**, permettendo una visualizzazione dettagliata sull'**Edificio 1** e sull'area degli **Uffici**.

Come prima cosa, siamo riusciti a simulare l'invio di 150000 sensori al nostro server in tempi ristretti (15 sec.), questo attraverso l'utilizzo di un socket client-server.



```
2
3
4• import java.io.IOException;
5 import java.net.Socket;
6 import java.net.UnknownHostException;
7
8
9 public class Client{
10
11
12
13
14•     public static void main(String[] args){
15
16
17
18
19     try {
20
21         Socket s = new Socket("localhost", 9999);
22         for (int i=1 ;i<=150000;i++) {
23
24             s.getOutputStream().write((int) (Math.random()*100));
25
26         }
27         s.close();
28     }
29
30
31         catch (UnknownHostException e) {
32             e.printStackTrace();
33         }
34
35         catch (IOException e) {
36
37             e.printStackTrace();
38         }
39     }
40 }
```

Immagine 1.1: codice della classe client



```

1 package prova_invio;
2
3 import java.io.IOException;
4
5 public class Server {
6
7     static int count=0;
8
9     public static void main(String[] args) {
10
11
12         try {
13             ServerSocket ss = new ServerSocket(9999) ;
14             while (true) {
15                 Socket s = ss.accept() ;
16                 for (int id=1;id<152018;id++) {
17                     System.out.println("il sensore con id = " + id + " segna il seguente valore : " + s.getInputStream().read() );
18                     count++;
19                 }
20
21                 System.out.println(count);
22                 s.close();
23                 ss.close();
24             }
25         } catch (IOException e) {
26             e.printStackTrace();
27         }
28     }
29 }
```

Immagine 1.2: codice della classe server

Una volta avviata la classe server (immagine 1.2) e mandata in esecuzione la classe client (immagine 1.1), la loro esecuzione genererà in maniera random 150000 dati, il problema che abbiamo riscontrato però è stato l'invio dei 150000 segnali ricevuti dal server al Data Base per permetterne il salvataggio, infatti il tempo per la memorizzazione è risultato enormemente più lungo rispetto ai 60 secondi richiesti dalla specifica.

Siamo però riusciti a permettere l'aggiornamento al minuto, il salvataggio sul Data Base e la visualizzazione sulla dashboard riducendo i dati a quelli relativi ad una sola area di un singolo edificio. Di seguito mostriamo come cambia il codice della classe client e della classe server e di come viene implementata la classe connessione, che appunto permette la connessione tra il sistema e il database:

SE course - Deliverables | 2018-2019

```
1 [REDACTED LINES]
public class Connessione {
    @SuppressWarnings("finally")
    public boolean insert(ArrayList<Object> args) {
        Connection connect = null;
        PreparedStatement preparedStatement = null;
        boolean success=true;
        try{
            connect=(Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/ospedale","root","C4neT0by");
            preparedStatement = connect.prepareStatement("INSERT INTO ospedale.dato(valore,IDsensore) VALUES(?,?)");
            preparedStatement.setFloat(1,(float)args.get(0));
            preparedStatement.setInt(2,(int)args.get(1));
            preparedStatement.executeUpdate();
        }catch(SQLException e){
            success=false;
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Errore");
            alert.setHeaderText("Errore Database");
            alert.setContentText(e.getMessage());
            alert.showAndWait();
        }catch(Exception e){
            success=false;
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Errore");
            alert.setHeaderText("Errore Generico");
            alert.setContentText(e.getMessage());
            alert.showAndWait();
        }finally{
            try{
                if(connect!=null) connect.close();
                if(preparedStatement!=null) preparedStatement.close();
                return success;
            }
            catch(final SQLException e){
                final Alert alert = new Alert(AlertType.INFORMATION);
                alert.setTitle("Errore");
                alert.setHeaderText("Errore Database");
                alert.setContentText(e.getMessage());
                alert.showAndWait();
                return false;
            }
        }
    }
}

public static void main(String[] args){
    njfi.setVisible(true);
    //Timer timer=new Timer();
    //TimerTask task= new MyTask();
    //timer.schedule(task,1000,5000);
    int y=0;
    Thread tl = Thread.currentThread();
    njfi.setVisible(true);
    try {
        Socket s = new Socket("localhost", 9888);
        for (int i=0;i<6;i++) {
            for(int z=0, id=1;z<48;z++,id++) {
                //y= ((int) (Math.random()*100));
                //System.out.println(y);
                Sensore a= new Sensore(id,"umidita", y, 40, 60);
                Controller c= new Controller(a);
                y=c.controllo();
                a.cambio_valore(y);

                a.invio();
            }
            s.close();
            try {
                tl.sleep(55000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

```

package dashboard;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
/**
 *
 * @author those
 */
public class Server {

    public Server() {
        // TODO Auto-generated constructor stub
    }
    public static void main(String[] args) {
        int cont=0;
        try {
            ServerSocket ss = new ServerSocket(9888) ;
            while (true) {
                Socket s = ss.accept() ;
                for (int i=0;i<48;i++) {
                    System.out.println(s.getInputStream().read());
                    cont++;
                }
                System.out.println(cont);
                s.close();
                ss.close();
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

Immagini 1.3, 1.4 e 1.5: classi connessione, client e server.

Ad oggi il nostro è un sistema statico e non dinamico, che quindi permette un corretto funzionamento solo in fase di compilazione e non ancora a run-time, quindi non possiamo garantire una grande performance. Inoltre, considerando il fatto che i dati sono random, non possiamo neanche garantire un'accuratezza.