“Software Engineering”

Course

a.a. 2018-2019

Template version 1.0

Deliverable #2

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

Dashboard Monitoraggio Ambientale

Date	23/12/2018
Deliverable	D2
Team (Name)	Soft_Ing

Team Members		
Name & Surname	**Matriculation Number**	**E-mail address**
Salvatore Salernitano	*242016*	*salvatore.salernitano@student.univaq.it*
Lorenzo Salvi	*242387*	*lorenzo.salvi@student.univaq.it*
Ludovico Di Federico	*243542*	*ludovico.difederico@student.univaq.it*
Marco Poscente	*243591*	*marco.poscente@student.univaq.it*
Francesco Catani	*246186*	*francesco.catani@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

Challenging Task	Date the task is identified	Date the challenge is resolved	Explanation on how the challenge has been managed
Area (geografica) non considerata come attore	12/11/2018	12/11/2018	Dopo un'attenta riflessione, il team ha deciso di non considerare l'area geografica un attore, dato che non possedeva servizi rilevanti (Use case).
Registrazione per i Gestori dei Sensori	13/11/2018	13/11/2018	I Gestori dei Sensori possono registrarsi/accedere al sistema mediante Username e Password
Component Diagram	21/11/2018	23/11/2018	Dopo un colloquio con il docente, ci siamo resi conto che il Component Diagram non deve riportare le funzionalità del sistema sotto un aspetto più approfondito ma deve rappresentare il Sistema ad alto livello e soprattutto deve tener conto dei Requisiti Non Funzionali.
Revisione ed Aggiornamento della Deliverable D2 partendo dalla D1	14/12/2018	14/12/2018	Il team, dopo un colloquio con il docente, ha deciso di riadattare il contenuto della Deliverable D1 nella Deliverable D2, apportando delle modifiche attinenti alla vecchia versione. (1. STORAGE NON NFR ma ASSUNZIONE; 2. DATABASE RIMOSSO COME NFR; 3. RIMOZIONE DELLE DECISIONI DEGLI FR E NFR DAL DECISION DESIGN ed AGGIUNTA DELLA DECISIONE ATTINENTE AL PATTERN MVC)
Ristrutturazione Component Diagram	23/11/18	17/12/18	Il team ha deciso di ristrutturare il Component Diagram in base alle necessità del prototipo
Ristrutturazione Modello ER	15/12/18	20/12/2018	Il team ha deciso di inserire come Foreign Key nella relazione Sensore, il campo Edificio in modo tale da avere maggior tracciabilità dei sensori all'interno di un piano associato all'edificio.

A. Requirements Collection

N.B: LE PARTI EVIDENZIATE IN GIALLO SONO L'AGGIUNTA DELLA DELIVERABLE D2

A.0 Detailed Scenarios

SCENARIO 1: PRIMO AVVIO: Mediante la Interfaccia Login, il *Gestore* dei sensori o l'Amministratore potrà accedere alla sua area riservata e monitorare i dati provenienti dai sensori; *l'Amministratore* del Sistema, potrà visualizzare i vari gestori che sono loggati e registrati all'interno del Sistema e potrà rispondere ai relativi ticket inviati dai gestori dei sensori.

SCENARIO 2: INVIO SEGNALE: Il sensore invia un segnale contenente le sue variabili ambientali e il suo stato di funzionamento (Rosso: Molti valori fuori soglia, Arancione: un solo valore fuori soglia, Verde: tutte i valori sono riportati in maniera corretta), a seconda dei valori fuori soglia, il sensore invierà più frequentemente il segnale. In caso di ALLARME, il sensore verrà evidenziato nella dashboard del Gestore mediante la sezione SENSORI A RISCHIO. Per i valori ambientali dei sensori meno rilevanti, il Gestore dei sensori potrà visualizzare le informazioni mediante la sezione SELEZIONARE AREA.

SCENARIO 3: EREDITARIETA' DEI GESTORI: Il gestore dei sensori (Gestore Città) potrà visualizzare il corretto stato della sua area (segnalato in V*erde*) solo nel caso in cui tutte le aree gestite dal corrispondente Gestore delle Aree, risultano a Norma. Di conseguenza, il rispettivo gestore dell'Area visualizza il corretto stato della sua area (sempre segnalato in *Verde*) solo se risulta a norma l'area gestita dal corrispondente gestore dell'Edificio.

SCENARIO 4: RIPRISTINO: Il sensore effettuerà un backup dei valori ambientali periodicamente ma solo in caso in cui i valori risultano a norma. Tali valori che verranno salvati, potranno essere utilizzati dal gestore dei sensori per poter ripristinare i valori ambientali in caso di dati fuori soglia.

SCENARIO 5: TICKET: Il gestore dei sensori potrà inviare un Ticket mediante SEGNALAZIONE ERRORI nel caso in cui risulta esserci un errore nel sistema software. Il ticket sarà visibile dall'Amministratore del Sistema mediante la sezione MONITORAGGIO SISTEMI.

A.1 Functional Requirements

Per modellare la complessità del nostro Sistema, il team ha optato per la **decomposizione funzionale** che consiste nel suddividere il Sistema in base alle sue funzionalità:

- **Dashboard**: interfaccia utilizzata dai gestori dei sensori e dall'Amministratore del Sistema per avere una panoramica dettagliata dell'intero funzionamento del Sistema Software in base ai loro privilegi;
- **Invio Segnali**: il nostro Sistema deve permettere ai sensori di inre periodicamente tutte le loro informazioni ambientali compreso lo stato di funzionamento;
- **Ripristino parametri sensore**: il Sistema deve garantire al Gestore dei Sensori di ripristinare i parametri dei sensori in caso in cui si riscontrano dei valori fuori soglia;
- **Backup parametri sensori**: I sensori devono essere in grado di effettuare periodicamente ed automaticamente i backup dei loro parametri;
- **Interazione**: l'Amministratore del Sistema e i Gestori dei Sensori devono interagire tra loro (es: mediante utilizzo dei ticket);
- **Registrazione Gestore Sensori**: L'Amministratore del Sistema può inserire un nuovo Gestore dei Sensori assegnandogli delle credenziali d'accesso;

A1.1 Use Case Diagrams

Il team ha deciso di utilizzare **Draw.io** per modellare lo Use Case Diagram:

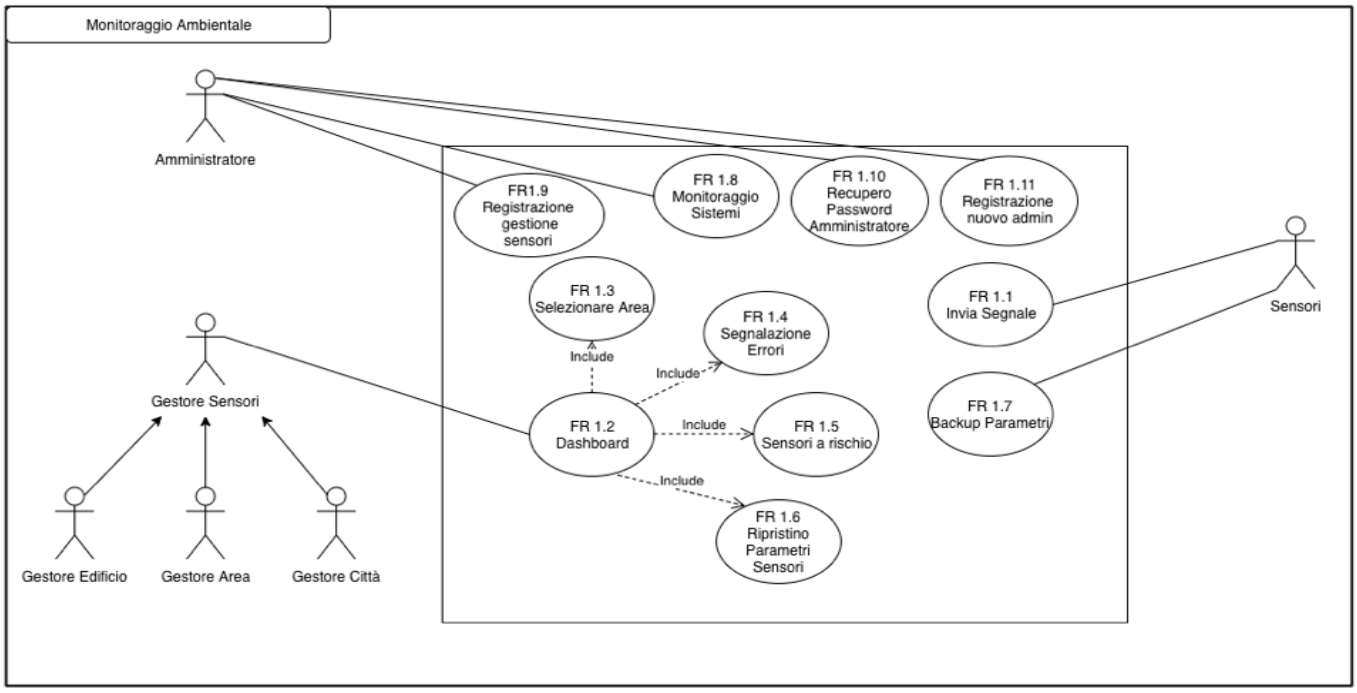

Fig. 1: Use-Case Diagram Monitoraggio Ambientale

ATTORI

1. **Sensori,** dispositivo elettronico con lo scopo di collezionare variabili ambientali (ad esempio, temperatura, luminosità, pressione, umidità) di una determinata area geografica;
2. **Gestore Sensori**, persona che si occupa della gestione (Dashboard Monitoraggio Ambientale) e della corretta funzionalità dei sensori inerenti ad una determinata area geografica.
   I Gestori dei Sensori saranno generalizzati a seconda dell'area geografica dove operano (ad esempio, **gestore di un edificio, gestore di un'area, gestore dell'intera città**);
3. **Amministratore**, colui che garantisce il corretto funzionamento del Sistema.

USE-CASES

- **FR 1.1**: Il sensore invia periodicamente i segnali (informazioni ambientali e sullo stato di funzionamento);
- **FR 1.2**: Il Gestore Sensori può accedere all'interno della sua area riservata mediante la Dashboard Monitoraggio Ambientale;
- **FR 1.3**: Il Gestore Sensori, può selezionare un'area geografica e visionare i sensori inerenti all'area d'interesse con tutte le loro informazioni;
- **FR 1.4**: il Gestore Sensori può segnalare un malfunzionamento del Sistema mediante l'utilizzo di ticket;
- **FR 1.5**: Vengono segnalati i sensori che presentano valori fuori soglia;
- **FR 1.6**: Il Gestore Sensori può effettuare un ripristino sui parametri di un sensore;
- **FR 1.7**: Il sensore effettua periodicamente ed automaticamente un backup dei suoi parametri;
- **FR 1.8**: L'Amministratore del Sistema effettua un'operazione di monitoraggio sul corretto funzionamento del Sistema Software;
- **FR 1.9**: L'Amministratore può registrare un nuovo Gestori dei Sensori assegnando delle credenziali d'accesso (es: username e password).
- **FR 1.10**: L'Amministratore del sistema può recuperare l'username e password mediante una chiave di recupero.
- **FR 1.11**: L'amministratore del sistema potrà aggiungere un nuovo amministratore assegnando delle nuove credenziali (es: username e password).

A1.2 Tabular description of the most relevant use case

Il Team ha deciso di dare priorità ai seguenti Use-Case:

FR 1.2

USE CASE	Dashboard.	
Goal in Context	Il Gestore Sensori può accedere alla sua area riservata tramite dashboard.	
Scope & Level	Questo Use Case ha lo scopo di permettere al Gestore Sensori di accedere al Sistema e svolgere le operazioni che sono di sua competenza.	
Preconditions	Ci aspettiamo che il Gestore Sensori in maniera del tutto sicura e rapida possa entrare nella dashboard. Ci aspettiamo che il Gestore Sensori abbia le credenziali di accesso alla sua area riservata.	
Success End Condition	Grazie a questo Use Case il gestore sensori può svolgere molte operazioni all'interno della dashboard, quali monitoraggio dei sensori e altre funzioni che garantiscono il corretto funzionamento degli stessi.	
Failed End Condition	Un mancato accesso al sistema da parte del gestore sensori andrebbe a compromettere il funzionamento dei Sensori di sua responsabilità.	
Primary, Secondary Actors	Gestore Sensori Gestore Edificio, Gestore Area, Gestore Città.	
Trigger	IL gestore di un determinato sensore effettua il login nella propria area riservata attraverso la Dashboard.	
DESCRIPTION	Step	Action
	1	Inserire Username
	2	Inserire Password
	3	Accesso alla Dashboard
EXTENSIONS	Step	Branching Action

	1a	-Selezionare Area interessata. -Segnalazione Errori del Sistema. -Info sui Sensori. -Ripristino Parametri Sensori.
SUB-VARIATIONS		Branching Action
	1	In base alla tipologia del gestore che opera nella Dashboard verranno evidenziati i parametri dei sensori più adeguati.

FR 1.8

USE CASE	Monitoraggio Sistemi.	
Goal in Context	Operazione di monitoraggio del Sistema Software.	
Scope & Level	L'amministratore effettua un'operazione di monitoraggio sul corretto funzionamento del Sistema.	
Preconditions	L'amministratore deve riuscire a monitorare il Sistema per un corretto funzionamento, attraverso opportuni privilegi che ha, rispetto ai gestori.	
Success End Condition	Una corretta operazione di monitoraggio permette di avere un Sistema sempre affidabile, sicuro ed efficiente.	
Failed End Condition	Una scorretta operazione di monitoraggio comprometterebbe il funzionamento del sistema e delle sue funzionalità che potrebbero risultare non veritiere. (Es. errore di sistema)	
Primary, Secondary Actors	Amministratore del sistema, Gestore dei Sensori.	
Trigger	L'amministratore del sistema verifica il corretto funzionamento del sistema stesso.	
DESCRIPTION	Step	Action

	1	Inserire Username, Password
	2	Accesso al Sistema
	3	Operazione di monitoraggio
EXTENSIONS	Step	Branching Action
	1a	_
SUB-VARIATIONS		Branching Action
	1	<list of variation s>

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

1. La GUI deve permettere all'utente (Gestore dei Sensori o Amministratore) di accedere al Sistema Software inserendo la propria username e password:

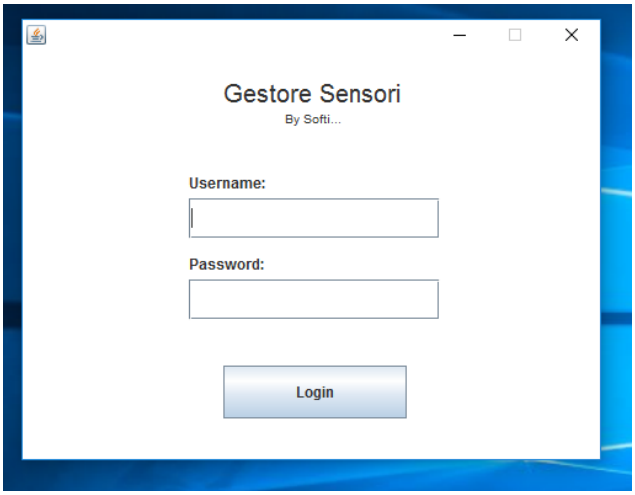

Fig. 2: Interfaccia Grafica della Login

2. Nel caso in cui non vengano inseriti l'username e la password verrà visualizzata una finestra dove si chiederà di inserire obbligatoriamente i dati per accedere alla Dashboard richiesta.

Fig. 3: Interfaccia Grafica Messaggio della Login

3. La Dashboard Admin deve permettere all'Amministratore di aggiungere un nuovo Admin e/o Gestore dei Sensori, recuperare l'username e la password di un altro Amministratore mediante il bottone "Chiave di Recupero", leggere e rispondere ai ticket inviati dai Gestori dei Sensori e monitorare l'operato degli ultimi:

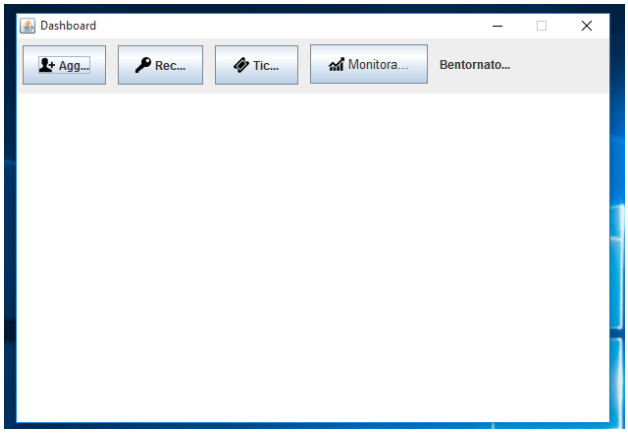

Fig. 4: Dashboard Admin

4. Cliccando sul bottone "Aggiungi", verrà mostrata una finestra dove sarà possibile aggiungere un nuovo Admin o Gestore dei Sensori:

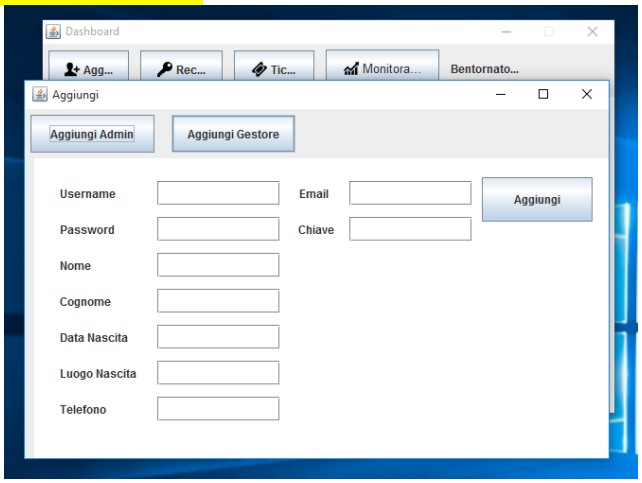

Fig. 5: Interfaccia Grafica Aggiungi Admin e/o Gestore Sensori

5. Cliccando il bottone “Ticket” verrà visualizzata una finestra dove sarà possibile rispondere ad un ticket precedentemente inviato e visualizzarli passando come parametro di ricerca l’ID del ticket:

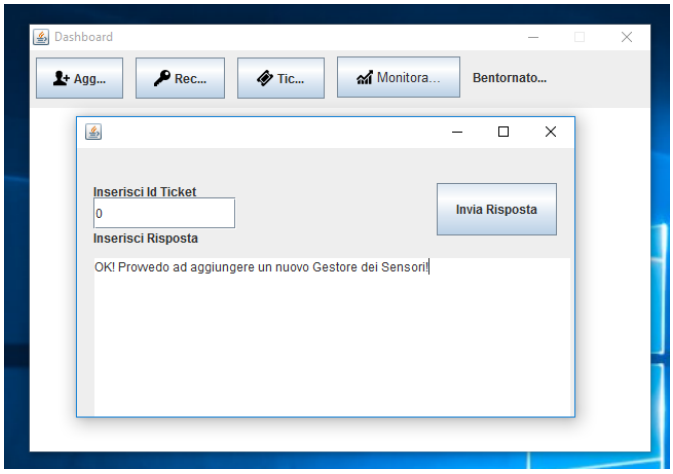

Fig. 6: Interfaccia Grafica del Ticket (Amministratore)

6. La Dashboard Gestore Sensori deve permettere al Gestore di monitorare le aree geografiche che gli sono state assegnate ed inviare i ticket agli amministratori nel caso in cui egli ha dei problemi di varia natura:

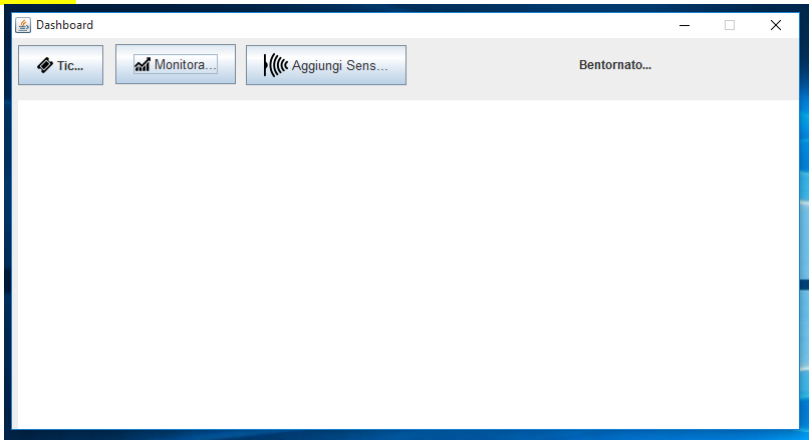

Fig. 7: Dashboard Gestore dei Sensori

7. Cliccando il tasto “Monitorare”, si aprirà una sottofinestra laterale dove sarà possibile selezionare un sensore ed accedere alle sue informazioni, oppure visualizzare una lista dei sensori:

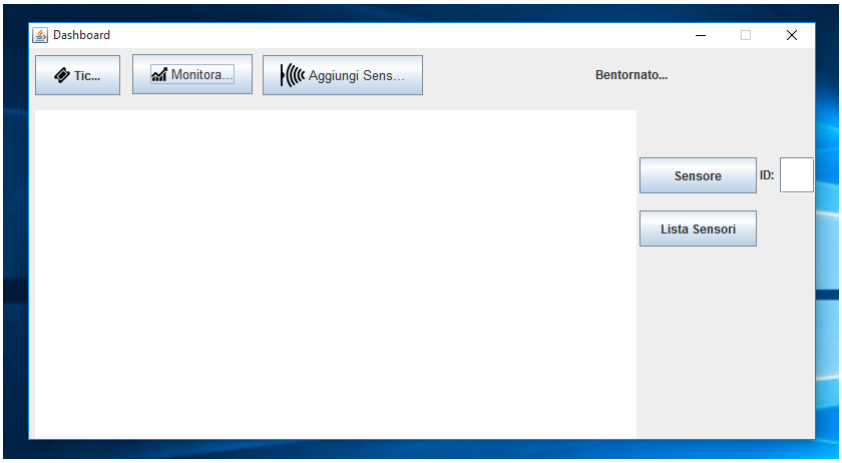

Fig. 8: Dashboard Gestore dei Sensori Bottone Monitorare

8. Attraverso la GUI (Dashboard Gestore), il gestore potrà impostare i massimali delle variabili ambientali del sensore appena installato e le variabili ambientali ottimali per un futuro ripristino nel caso in cui il sensore avrà i valori fuori soglia.

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

1. *Deve implementare una funzione per salvare il segnale nel database;*
2. *Deve implementare una funzione per effettuare periodicamente ed automaticamente il backup dei sensori;*
3. *Deve implementare una funzione con il quale l’amministratore del sistema può inserire un nuovo gestore dei sensori o amministratore;*
4. *Decide le regole per l’uso del sistema;*
5. *Deve implementare una funzione che garantisce al gestore dei sensori di ripristinare i parametri dei sensori nel caso in cui le variabili ambientali dei sensori risultino fuori soglia;*
6. *Deve essere possibile recuperare le credenziali;*
7. *Deve essere possibile gestire e monitorare determinati sensori in una certa area geografica;*
8. *Deve essere in grado di gestire un malfunzionamento del sistema tramite l’utilizzo di ticket.*

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

1. *Il Database deve essere in grado di memorizzare i dati relativi ai segnali dei sensori, disporre di uno Storage nel quale immagazzinare tali dati senza correre il rischio che vengano persi.*
2. *Il Database deve essere sempre operativo.*
3. *Deve essere capace di rispondere e gestire tutte le richieste mandate dai servizi del sistema nel minor tempo possibile.*

A.2 Non-Functional Requirements

N.F.R. 1 DEPENDABILITY:

- ***N.F.R. 1.1 FAULT TOLERANCE****: Il sistema deve continuare ad offrire i servizi anche se si è persa la funzionalità di una componente. (Esempio: In un piano di un edificio, devono essere presenti almeno due sensori in modo tale da garantire il corretto monitoraggio del piano anche nel caso in cui un sensore risulti danneggiato/fuori uso).*
- ***N.F.R. 1.2 SECURITY****: Il sistema deve essere in grado di proteggersi da attacchi esterni, accidentali o intenzionali. La security diventa un requisito essenziale, visto che il nostro sistema dovrà essere connesso alla rete per permettere ai sensori, ai gestori dei sensori e all'amministratore di inviare dati rilevanti, come ticket, segnali, ecc…..*

N.F.R. 2 SCALABILTY*: Il sistema garantisce un'architettura scalabile per supportare future espansioni.*

N.F.R. 3 USABILITY*: Il sistema deve offrire una user-experience-friendly. Ovvero, deve risultare facile da utilizzare. (Es: Ripristino dei parametri dei sensori)*

N.F.R. 4 PERFORMANCE*: Il sistema deve risultare efficiente e deve lavorare con tempi di esecuzione accettabili, anche nel caso in cui ci siano più accessi in parallelo dei gestori.*

A.3 Excluded Requirements

Il team ha deciso di escludere le seguenti funzionalità:

- **Comunicazione tra ditta che utilizza il nostro sistema software con l'Amministratore:** non è di nostra competenza conoscere il modo con cui la ditta chiede di creare *username* e *password* di un nuovo Gestore di Sensori (es. E-mail, fax, recapito telefonico). L'unica cosa di nostra competenza e' assegnare *username* e *password* ai gestori per permettergli di accedere alla dashboard.
- **Fase realizzativa dei sensori in maniera fisica:** non e' di nostra competenza la realizzazione e manutenzione del sensore o parti di esso. L'unica cosa di nostra competenza è la configurazione delle variabili ambientali del sensore mediante il backup del sensore.

A.4 Assumptions

- Potranno utilizzare il Sistema solo I gestori dei sensori che hanno delle credenziali d'accesso (Username e Password);
- Il sensore dovrà avere, al momento della sua prima configurazione, un backup di default ed un range di valori accettabili inerenti alle variabili ambientali.
- Il sensore acquisisce i valori ambientali in un lasso di tempo determinato.
- I sensori non devono comunicare tra loro.
- Il segnale verrà eliminato programmaticamente ogni 6 ore.
- Esiste almeno un Amministratore al momento della creazione del Sistema Software.
- Il sistema dovrà supportare lo storage and il processing di almeno 150.000 messaggi al minuto.

A.5 Prioritization

ID	Requisito	Priorità
FR 1.2	Dashboard	Alta
FR 1.8	Monitoraggio Sistemi	Alta
FR 1.5	Sensori a rischio	Media/Alta
FR 1.1	Invia Segnali	Media/Alta
FR 1.4	Segnalazione Errori	Media
FR 1.6	Ripristino Parametri Sensori	Media
FR 1.9	Registrazione gestione sensori	Media
FR 1.10	Recupero Password Amministratore	Media
FR 1.11	Registrazione Nuovo Admin	Media
FR 1.3	Selezionare Area	Bassa
FR 1.7	Backup Parametri	Bassa

B. Software Architecture

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

Il team ha deciso di utilizzare **Draw.io** per modellare il Component Diagram:

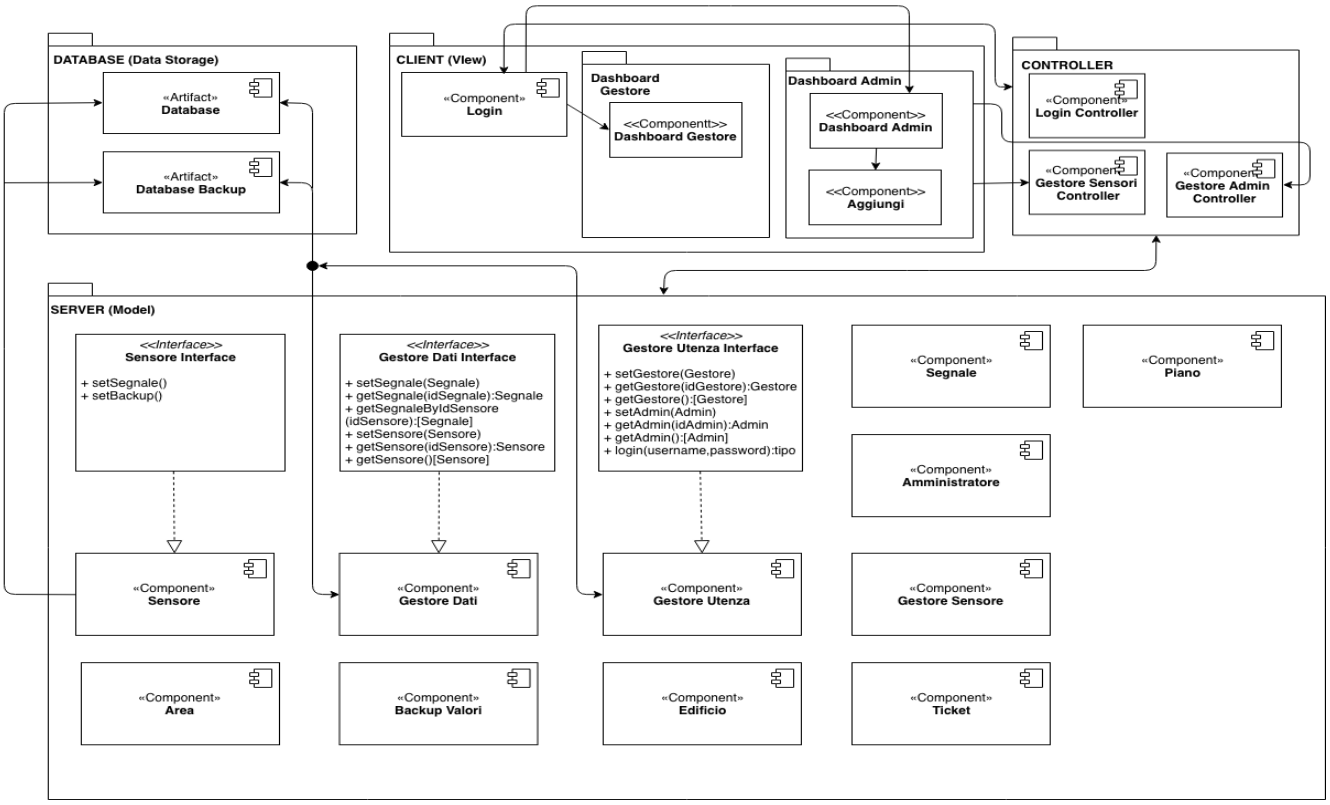

Fig. 9: Component Diagram Monitoraggio Ambientale

Il team ha deciso di utilizzare ***MVC Pattern (Model-View-Controller)***, ovvero un Pattern Architetturale molto diffuso nello sviluppo software, in particolare nell'ambito della programmazione orientata ad oggetti. La suddivisone in strati facilita la scalabilità e la manutenzione del software.

Il pattern è basato sulla separazione dei compiti fra i componenti softwareche interpretano tre ruoli principali:

- ***Model*** fornisce i metodi per accedere ai dati utili all'applicazione;
- ***View*** visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
- ***Controller*** riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.

Le Macro-componenti che suddividono la nostra architettura, sono le seguenti:

- **Il DATABASE:** Il sistema dovrà includere una ridondanza sul database in modo tale da non rischiare una perdita di dati conterrà, ovvero due componenti denominate "*database*" e "*databasebackup*". Hanno il compito di immagazzinare i dati e garantire che gli stessi non vengano persi.
- **CLIENT (GUI):** conterrà altri due package chiamati rispettivamente: *"Dashboard gestore"* e *"Dashboard Admin"*, che al suo interno avranno come componenti: *"Dashboard gestore", "Dashboard Admin"* e *"Aggiungi"*. Questo macro blocco ha il compito di interfacciare l'utente con il sistema (***View***). "*Dashboard admin*" rappresenta l'interfaccia grafica tra l'amministratore del sistema ed il sistema stesso mentre la "*dashboard gestore*" rappresenta l'interfaccia grafica tra il gestore dei sensori ed il sistema del monitoraggio ambientale. Il component "*login*" svolge una funzione importante, ovvero ha il compito di identificare un determinato utente che effettua l'accesso in modo tale da garantire una protezione da attacchi esterni, accidentali o intenzionali (***requirement non functional security***).
- ***CONTROLLER***: è una macro componente che ha il compito di elaborare le richieste in ingresso, gestire gli input e le interazioni del singolo utente ed eseguire la logica del sistema appropriato. Al suo interno troveremo tre componenti, denominate: "*login controller*", "*gestore sensori controller*" e "*gestore admin controller*".
- **SERVER**: è un macro componente che fornisce i metodi per accedere ai dati utili all'applicazione (***Model***). Il ***Model*** non si occupa soltanto dell'accesso fisico ai dati ma anche di creare il necessario livello di astrazione tra il formato in cui i dati sono memorizzati ed il formato in cui i livelli di ***controller e view*** si aspettano di riceverli; oltretutto fornisce una interpretazione intermedia dei dati arricchendo il database con nuove informazioni. Cosa importante, il ***Model*** non contiene direttamente i dati del database, ma ha solo il compito di fornire metodi e di restituire i dati al *controller.*

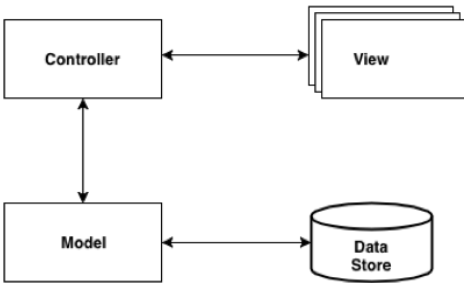

Fig. 10: Schema del Pattern MVC

*L*a macro-componente **SERVER** conterrà i seguenti components: "*Sensore*": E' una classe che verrà implementata dalla component "*Interfaccia Sensore*" che al suo interno conterrà i seguenti metodi: "*setSegnale()*","*setBackup()*" che avranno il compito di scrivere all'interno del macro-componente Database; Il component "*Gestore Dati*" è uno tra i component più importanti, implementato dall'interfaccia "Gestore*Dati*", contenente tutti i metodi attinenti alle funzionalità del sistema, ad esclusione di quelle di utenza. Esso avrà inoltre il compito di scrivere all'interno del macro-componente "**DATABASE**".
Il component "Gestore*Utenza*", sarà implementato dall'interfaccia "*Gestore Utenza*", che contiene tutti i metodi attinenti alle funzionalità di utenza (*setGestore, getGestore, setAdmin, getAdmin, login, getGestore(Gestore), getAdmin(Admin)*) e, avrà il compito di scrivere all'interno del macro-componente "**DATABASE**". Queste tre classi (*"Sensore", "Gestori Dati", "Gestore Utenza"*) avranno anche il compito di garantire il *Requirement Non Functional Scalability*. I components "*Segnale*", "*Amministratore*",*"Area"."BackupValori","Edificio","Piano","Ticket"* rappresentano gli oggetti che verranno utilizzati nel ***Model***.

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

Il team ha deciso di utilizzare **MagicDraw** per modellare il Sequence Diagram:

SEQUENCE DIAGRAM SCENARIO 2: INVIO SEGNALE

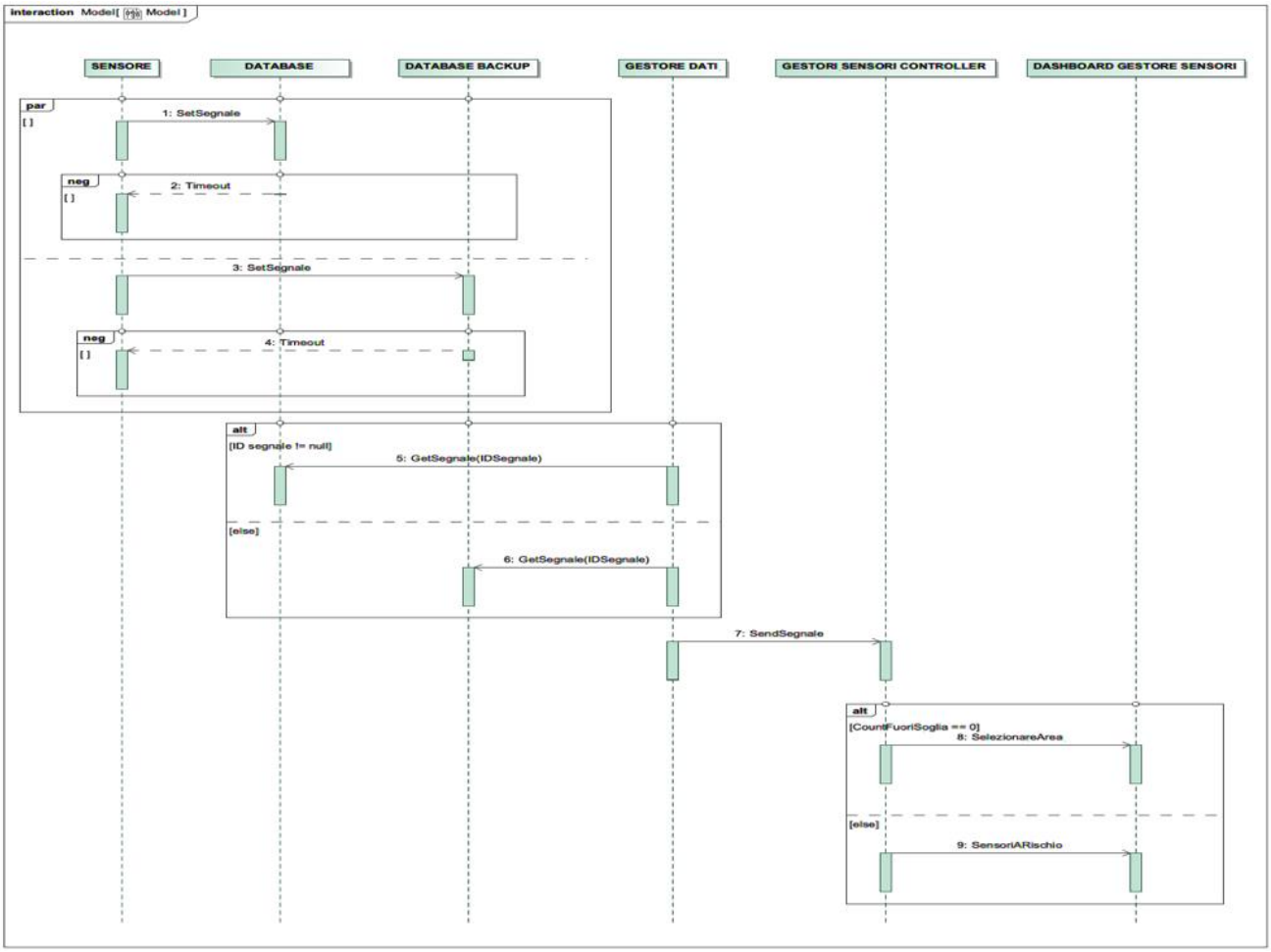

Fig. 11: Sequence Diagram inerente allo Scenario 2: INVIO SEGNALE

Il **Sensore** invierà i dati del segnale al **Database** e al **Database Backup**, in parallelo, mediante il metodo "*1:setSegnale*", "*3:setSegnale*"; Se il segnale non viene scritto sul Database o sul Database Backup, essi invieranno una risposta chiamata "*Timeout*".
Il GestoreDati invierà il messaggio "*sendSegnale*" al **GestoreSensoriController**il quale avrà il compito di far visualizzare i dati del segnale nella zona della dashboard appropriata.
Il GestoreSensoriController a seconda del valore della variabile "contFuoriSoglia", interfaccierà i dati del segnale con il sensore associato in una determinata area della **Dashboard GestoreSensore**. Se "countFuoriSoglia" è uguale a 0 allora i dati saranno visualizzati su "*SelezionareArea*", altrimenti su "*Sensori a rischio*".

SEQUENCE DIAGRAM SCENARIO 4: RIPRISTINO

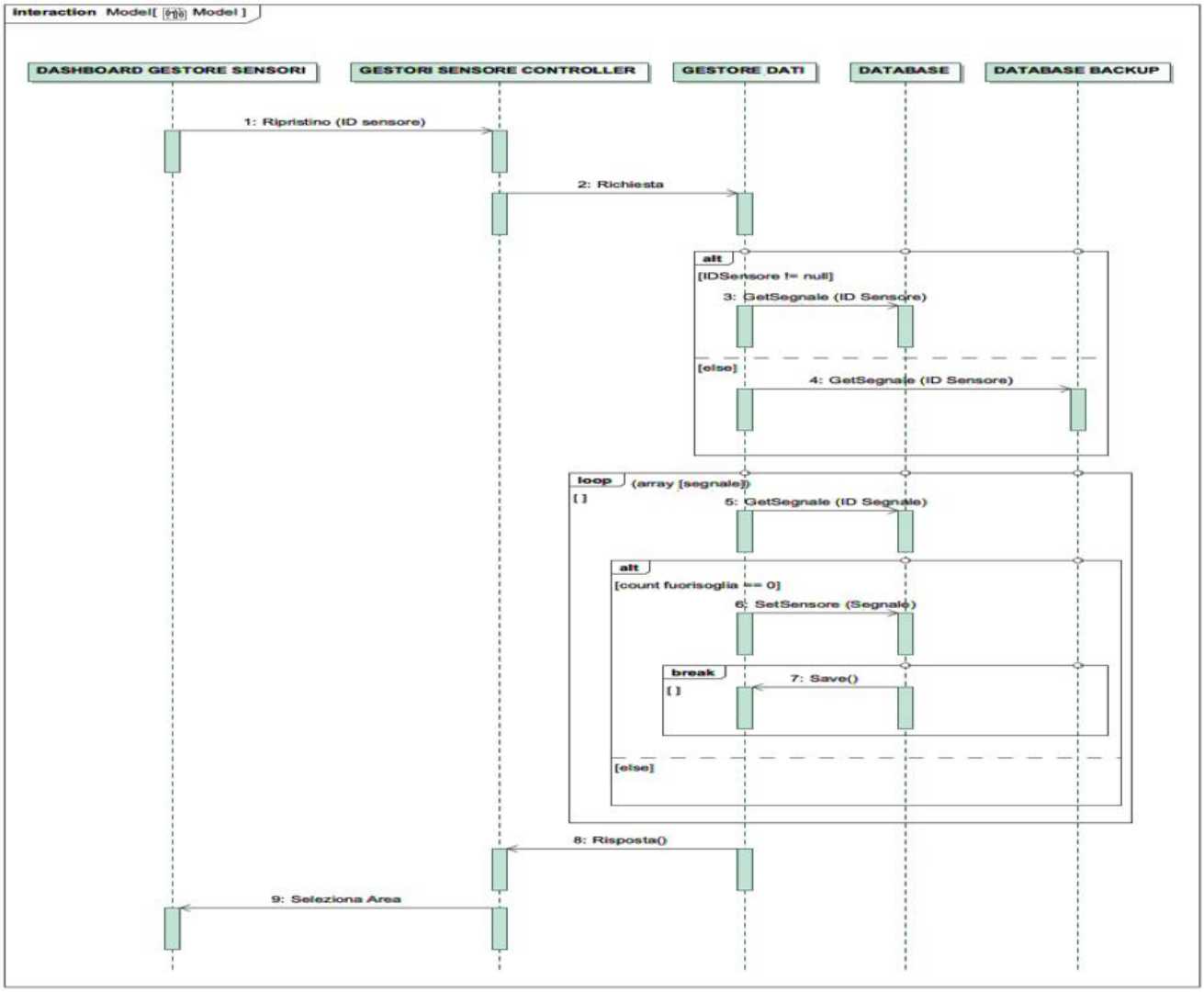

Fig. 12: Sequence Diagram inerente allo Scenario 4: RIPRISTINO

Il Gestore mediante la **Dashboard Gestion Sensori** effettuerà il *ripristino* passando come parametro "idSensore" al **GestoreSensoreController**. Tale gestore interfaccierà la *richiesta*al **Gestore Dati.**Il **GestoreDati**invierà il messaggio "*getSegnale(idSensore)*" al **Database**per prelevare i dati inerenti ai segnali del sensore passato come parametro; nel caso in cui "idSensore" fosse uguale a null, il **GestoreDati**invierà il messaggio "*getSegnale(idSensore)*" al **DatabaseBackup**.
Il GestoreDati scorrerà l'array dei segnali inerente a quel sensore fino a quando non avrà un segnale con i valori a norma.
Il GestoreDati invierà un messaggio al Database chiamato "setSensore" passando come parametro il segnale a norma e uscendo cosi dal loop.
Il GestoreDati invierà una risposta al GestoreSensoreController che interfaccerà le modifiche di quel sensore nella zona della Dashboard "Selezionare Area".

C. ER Design

Il team ha deciso di utilizzare **draw.io** per la realizzazione dell'ER-Model:

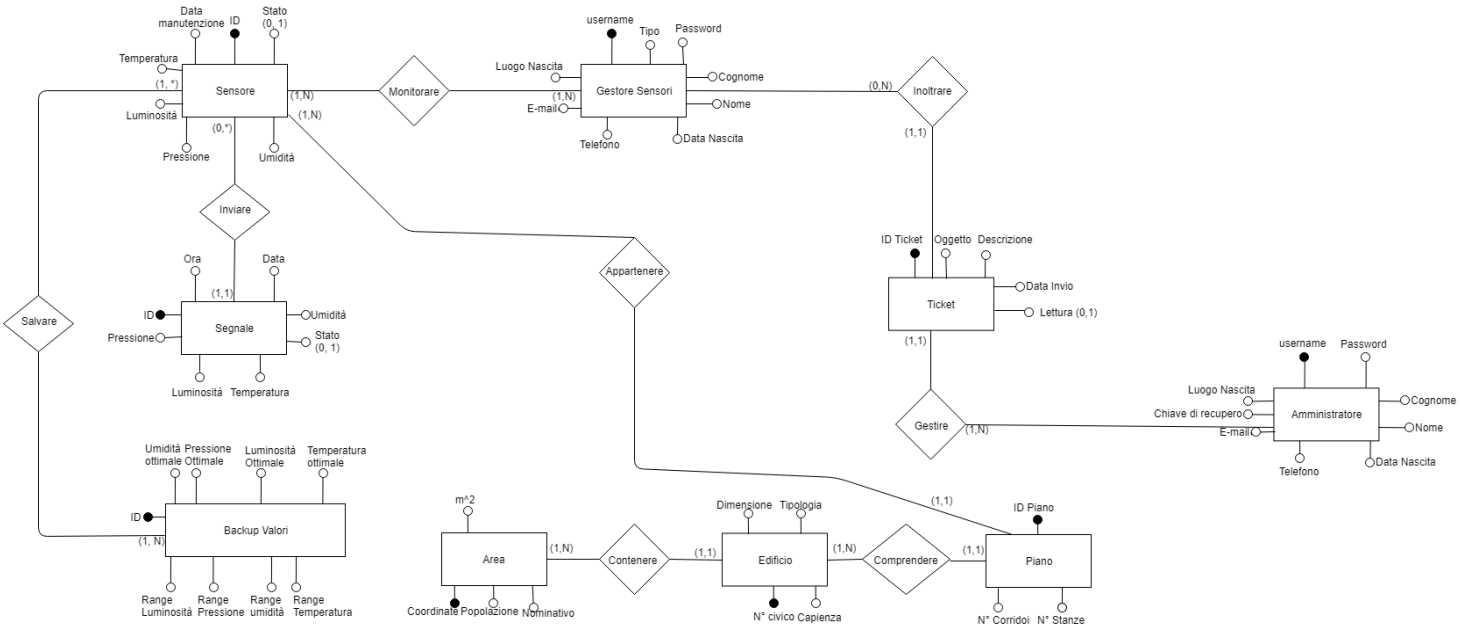

Fig. 13: Modello ER Monitoraggio Ambientale

Modello Relazionale:

Sensore (**ID**, DataManutenzione, Stato, Temperatura, Luminosità, Pressione, Umidità, *Piano*, *Gestore, Edificio*).

Segnale (**ID**, Data, Ora, Stato, Umidità, Temperatura, Luminosità, Pressione, *IDSensore*).

BackupValori (**ID**, RangeLuminosità, RangePressione, RangeUmidità, RangeTemperatura, UmiditàOttimale, PressioneOttimale, TemperaturaOttimale, LuminositàOttimale, *IDSensore*).

GestoreSensori (**Username**, Password, Nome, Cognome, Tipo, LuogoNascita, DataNascita, Email, Telefono).

Area (**Coordinate**, Popolazione, Nominativo, m^2).

Edificio (**NCivico**, Capienza, Tipologia, Dimensione, *Area*).

Piano (**IDPiano**, NCorridoi, NStanze, *Edificio*).

Ticket (**IDTicket**, Oggetto, Descrizione, DataInvio, Lettura, *Mittente, Destinatario*).

Amministratore (**Username**, Password, Nome, Cognome, DataNascita, LuogoNascita, Telefono, Email, ChiaveDiRecupero).

DESCRIZIONE E-R MODEL

1) Consideriamo innanzitutto l'entità **SEGNALE**. Tale entità è costituita dagli attributi ***id****, data, ora, stato, umidità, temperatura, luminosità, pressione*. Tra questi, l'***id*** rappresenta l'attributo che permette di contraddistinguere univocamente ciascun segnale, *data* e *ora* rappresentano, appunto, giorno e ora dell'invio di un segnale da parte del sensore associato mediante la ForeignKey *IDSensore*, e i rimanenti attributi rappresentano le informazioni ambientali (*umidità, temperatura, luminosità, pressione*) e le informazioni di stato di funzionamento (0,1) che il sensore ha inviato nel momento t = x, dato come detto precedentemente dalla data e ora.

2) L'entità **SENSORE**, è costituita dagli attributi delle variabili ambientali che vengono passati al segnale ad esso associato e dovrà occuparsi quindi di inviarli periodicamente. Un sensore potrà inviare (0,*) segnali: questa cardinalità sta a significare che un sensore è in grado di inviare indefiniti segnali oppure nessuno nel caso di un'anomalia. L'attributo *data manutenzione*, rappresenta appunto un'eventuale manutenzione che è stata effettuata sul sensore. Chiaramente ciascun SENSORE ha un'**id** (caratterizzato da un codice numerico auto incrementale) come chiave primaria che permette di distinguere univocamente il sensore rispetto ad altri.

3) Tale entità deve occuparsi anche di salvare le informazioni che le vengono passate per verificare in seguito se rientrano in parametri accettabili e veritieri, e per permettere al Gestore dei Sensori di effettuare il ripristino delle variabili ambientali attinenti ad esso. Tutto questo rappresentato dall'associazione tra, appunto, SENSORE e l'entità **BACKUP VALORI**. Nello specifico i sensori possono salvare, cioè backuppare, uno o indefiniti valori, ovvero con cardinalità (1,*). Tornando all'entità BACKUP VALORI, costituita, quindi, dagli attributi *umidità ottimale, temperatura ottimale, luminosità ottimale, pressione ottimale* utilizzati dal gestore sensori per permettere il ripristino delle variabili ambientali del sensore nel caso in cui essi risultano fuori soglia, e i corrispondenti range di valori (quindi parametri veritieri).

I sensori sono distribuiti all'interno di zone geografiche, ovvero grazie ad essi si potrà monitorare un'intera zona, gli edifici che la compongono e infine i piani che appartengono a ciascun edificio.

4) In dettaglio uno o più sensori appartengono a ciascun piano, identificato dall'entità **PIANO.** I suoi attributi sono l'***id piano,*** *N. stanze, N. corridoi.* L'id (rappresentato da un codice numerico) serve per contraddistingure i vari piani e gli altri due attributi descrivono appunto il numero di stanze e di corridoi che formano ciascun piano.

5) Detto ciò, dall'associazione tra l'entità PIANO e l'entità **EDIFICIO** si è in grado di monitorare in dettaglio tutti gli edifici. Ciascun Edificio potrà avere più piani (cardinalità 1,N) e ciascun piano descriverà qundi una parte di esso (cardinalità 1,1). Gli attributi dell'entità EDIFICIO sono ***Ncivico****, capienza, dimensione, tipologia.* Il N° civico rappresenta la chiave primaria (rappresentato da un numero), *dimensione, capienza* e *tipologia* forniscono informazioni dei vari edifici, quindi se sono ospedali, biblioteche, ecc... . Da tutti questi dati si potra' distinguere piani che ad esempio hanno codici uguali ma comunque appartengono ad edifici diversi.

6) I vari edifici poi sono contenuti in aree. Ovvero, associazione **EDIFICO** e **AREA**. Anche qui, ogni area conterrà uno o più edifici (cardinalità 1,N) e ciascun edificio descriverà ogni parte di una zona (cardinalità 1,1). L'entità AREA ha come attributi secondari *nominativo, npopolazione, metri quadrati* e come attributo principale (chiave primaria) le **coordinate** di dove si trova.

7) Passiamo ora all'entità **GESTORE DEI SENSORI**. Il gestore ha i seguenti attributi: ***username****, tipo, password, cognome, nome, data di nascita, luogo di nascita, telefono, e-mail*. In dettaglio, l'attributo ***username*** identifica univocamente tutti i gestori e rappresenta con la *password* le credenziali per accedere alla propria area riservata. Mentre l'attributo *tipo* rappresenta la categoria a cui appartengono i gestori: gestore *città*, gestore *area*, gestore *edificio*. I vari gestori devono monitorare la zona d'interesse di loro competenza e quindi vedere in modo semplice ed accurato e con il giusto livello di dettaglio le informazioni rilevanti. Queste due operazioni sono rappresentate dall'associazione "monitorare" tra entità SENSORE ed entità GESTORE DEI SENSORI con cardinalità (1,N) per rappresentare il fatto che i gestori possono monitorare più sensori; e la possibilità da parte dei gestori di eseguire un'operazione di zoom-in,zoom-out sulla zona di loro competenza.

8) I gestori non possono comunicare tra di loro. Se si verifica un malfunzionamento di qualunque tipo, il gestore deve comunicarlo agli amministratori del sistema, tramite opportuni **TICKET**. Questa azione e' rappresentata dall'associazione "inoltrare" tra entità GESTORE DEI SENSORI e TICKET. Associazione con cardinalità (0,N) per descrivere che i gestori possono inviare più di un ticket e con cardinalità (1,1) per rappresentare il fatto che lo stesso ticket non potrà essere inoltrato da gestori diversi.

9) Infine troviamo l'entità **AMMINISTRATORE**, ovvero coloro che devono garantire il corretto funzionamento del sistema. Caratterizzato dai seguenti attributi: ***username****, password, cognome, nome, data di nascita, luogo di nascita, telefono, e-mail, chiave di recupero*. La chiave primaria e' rappresentata dall'*username*. L'attributo chiave di recupero, specifica che nel caso in cui un amministratore perde un dato tra username e password sia possibile recuperarlo o quantomeno cambiarlo. Come abbiamo accennato poco prima, l'amministratore deve gestire i ticket che riceve dai vari gestori. Ciò è specificato dall'associazione "gestire" tra entità TICKET e AMMINISTRATORE. Anche qui le cardinalità (1,1) ed (1,N) rappresentano il fatto che uno stesso ticket non può essere gestito da più amministratori ma comunque un amministratore può ricevere e gestire più di un ticket.

D. Class Diagram of the implemented System

Il team ha deciso di utilizzare **draw.io** per la realizzazione del Class Diagram:

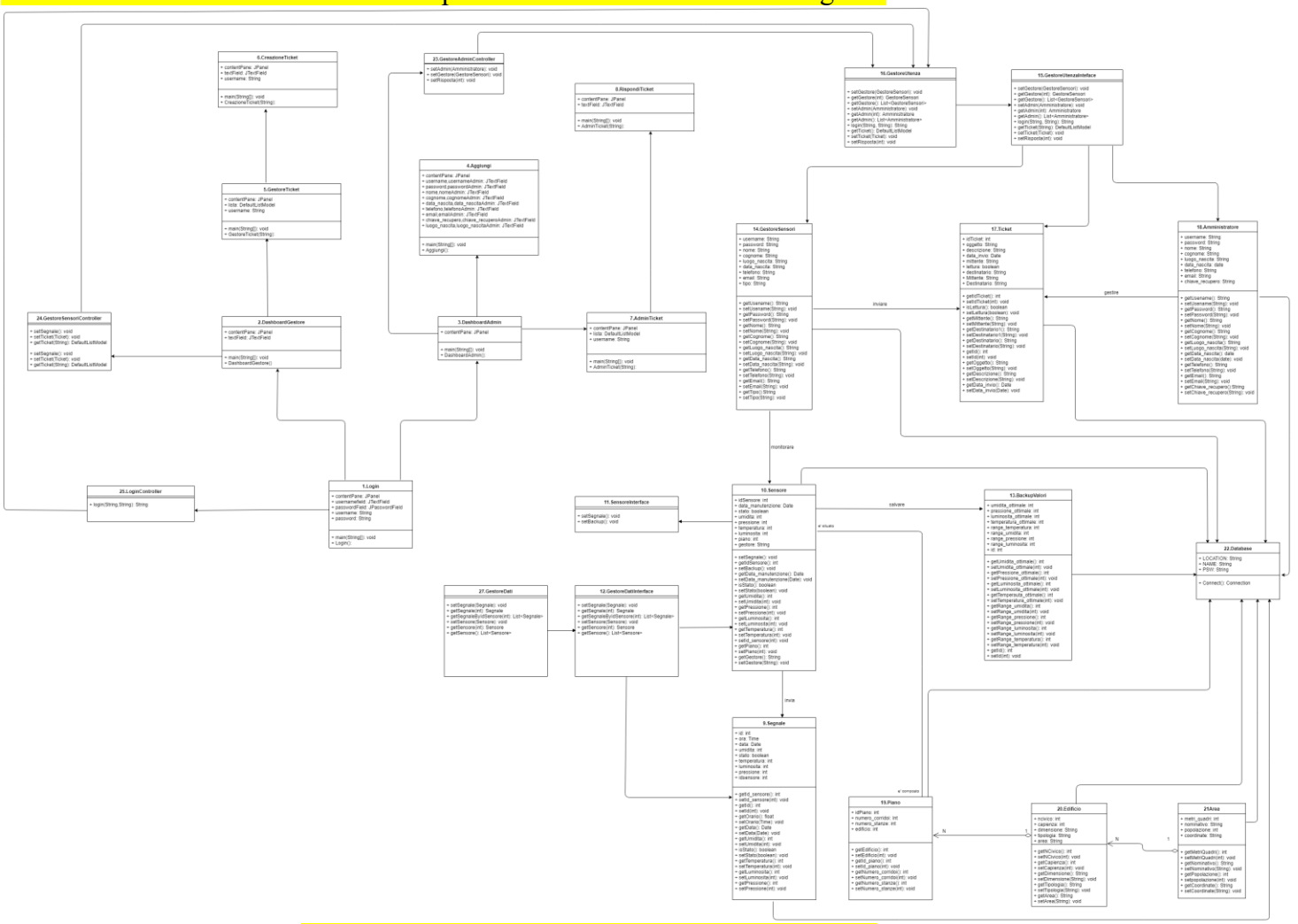

Fig.14: Class Diagram Monitoraggio Ambientale

Il ***Class Diagram*** riporta tutte le classi usate all'interno del nostro sistema software:

1) **LOGIN:** E' una classe presente nel file *Login.java* nel package *View* che permette l'accesso alle Dashboard da parte di un Gestore dei Sensori o di un Admin; E' una classe importante perché rappresenta l'interfaccia grafica del Primo Avvio del nostro sistema software.
2) **DASHBOARDGESTORE**: E' una classe presente nel file *DashboardGestore.java* nel package *View.DashboardGestore* che permette ai Gestore dei Sensori di monitorare i sensori ed inviare dei Ticket; E' una classe importante perché rappresenta l'interfaccia grafica del nostro sistema software.
3) **DASHBOARDADMIN**: E' una classe presente nel file *DashboardAdmin.java* nel package *View.DashboardAdmin* che permette all'Admin di aggiungere un nuovo Admin/Gestore, gestire

i Ticket e monitorare il sistema software; E' una classe importante perché rappresenta l'interfaccia grafica del nostro sistema software.

4) **AGGIUNGI**: E' una classe presente nel file *Aggiungi.java* nel package *View.DashboardAdmin* che permette all'Admin di aggiungere un nuovo gestore o un nuovo Admin;
5) **GESTORETICKET**: E' una classe presente nel file *GestoreTicket.java* nel package *View.DashboardGestore* che permette di visualizzare/gestire i Ticket inviati dal Gestore dei Sensori;
6) **CREAZIONETICKET**: E' una classe presente nel file *CreazioneTicket.java* nel package *View.DashboardGestore* che permette al gestore dei sensori di creare un nuovo Ticket;
7) **ADMINTICKET:** E' una classe presente nel file *AdminTicket.java* nel package *View.DashboardAdmin* che permette di listare/gestire tutti i Ticket che i Gestori dei Sensori hanno inviato all'Admin;
8) **RISPONDITICKET**: E' una classe presente nel file *RispondiTicket.java* nel package *View.DashboardAdmin* che permette di rispondere ai Ticket dei Gestori;
9) **SEGNALE**: E' una classe presente nel file *Segnale.java* nel package *Model.Components* che permette di costruire oggetti di tipo Segnale nel momento in cui viene invocato il metodo ***setSegnale()*** dalla classe Sensore;
10) **SENSORE:** E' una classe presente nel file *Sensore.java* nel package *Model.Components* che permette di effettuare delle operazioni di scrittura/lettura sul Database, nel preciso nella tabella Sensore; E' una classe importante perché senza di essa non sarebbe possibile effettuare un'operazione di monitoraggio, l'invio del segnale mediante il metodo ***setSegnale(),*** e il backup delle variabili ambientali ottimali, mediante il metodo ***setBackup()*** ed il range su cui il sensore si deve basare per avvertire il gestore dei sensori di presenza di valori fuori soglia;
11) **SENSOREINTERFACE:** E' una classe Interfaccia che è implementata dal Sensore che permette di inviare i Segnali ed effettuare il Backup;
12) **GESTOREDATIINTERFACE**: E' una classe Interfaccia che opera sul Sensore e sul Segnale che permette di effettuare delle operazioni di supporto nella definizione di nuovi metodi che operano sui dati presenti nel sensore e nel segnale;
13) **BACKUPVALORI**: E' una classe presente nel file *BackupValori.java* nel package *Model.Components* che permette di effettuare operazioni di lettura/scrittura sul Database. E' una classe molto importante perché senza di essa non si potrebbero effettuare operazioni come il Ripristino sulle variabili ambientali dei Sensori mediante i metodi di ***get e set*** delle variabili ambientali ottimali;
14) **GESTORESENSORI:** E' una classe presente nel file *GestoreSensori.java* nel package *Model.Components* che permette di effettuare operazioni di lettura/scrittura sul database;
15) **GESTOREUTENZAINTERFACE:** E' una classe Interfaccia che opera sul GestoreSensori, Ticket e Amministratore e permette di effettuare operazioni come: modificare, restituire il Gestore, l'Admin, il Ticket e la sua eventuale risposta;
16) **GESTOREUTENZA:** E' una classe che implementa la classe GestoreUtenzaInterface dove sono presenti le definizioni dei metodi riepilogati nel punto precedente. Classe molto importante dove è presente la logica della Login e della modifica, restituzione e aggiunta dei dati inerenti all'Admin e al Gestore dei Sensori nel Database.
17) **TICKET:** E' una classe presente nel file *Ticket.java* nel package *Model.Components* che permette di effettuare operazioni di lettura/scrittura sul Database, nello specifico nella relazione ticket;

18) **AMMINISTRATORE:** E' una classe presente nel file *Amministratore.java* nel package *Model.Components* che permette di effettuare operazioni di lettura/scrittura sul Database, nel dettaglio nella relazione Amministratore;
19) **PIANO:** E' una classe presente nel file *Piano.java* nel package *Model.Components* che permette di costruire oggetti di tipo Piano. L'insieme degli oggetti Piano rappresentano un ***agglomerato*** dell'oggetto Edificio.
20) **EDIFICIO:** E' una classe presente nel file *Edificio.java* nel package *Model.Components* che permette di costruire oggetti di tipo Edificio. L'insieme degli oggetti Edificio rappresentano un ***agglomerato*** dell'oggetto Area.
21) **AREA:** E' una classe presente nel file *Area.java* nel package *Model.Components* che permette di costruire oggetti di tipo Area ed, ovviamente, di leggere e scrivere all'interno della relazione Area;
22) **DATABASE:** Rappresenta la classe che permette lo storage dei dati, infatti sono presenti metodi come ***Connect()*** con l'obiettivo di instaurare una connessione tra il database e il programma in Java;
23) **GESTOREADMINCONTROLLER**: E' una classe presente nel file *GestoreAdminController.java* nel package *Controller* che permette di modificare i dati presenti nel *Model* nello specifico in Admin, Gestore e Risposta;

N.B.: *Le classi 5), 6), 8) e 9) rappresentano delle classi importanti per permettere la buona riuscita della messaggistica tra il Gestore dei Sensori e l'Admin.*

24) **GESTORESENSORECONTROLLER**: E' una classe presente nel file *GestoreSensoreController.java* nel package *Controller* che permette di modificare i dati presenti nel *Model* nello specifico in segnale, sensore e ticket;
25) **LOGINCONTROLLER:** E' una classe presente nel file *LoginController.java* nel package *Controller* che permette di leggere i dati passati dalla login nel *Model* e restituire una stringa contenente *l'username e password* dell'utente che accede.
26) **GESTOREDATI**: E' una classe che implementa la classe *GestoreDatiInterface* dove sono presenti le definizioni dei metodi riepilogati nel *punto 12)*. Classe molto importante dove è presente la logica dell'invio del segnale e della configurazione del backup inerente al Sesnore nel Database.

E. Design Decisions

1) DECISION PROGRAMMING LANGUAGES: Il team ha deciso per lo sviluppo del software Monitoraggio Ambientale di utilizzare il linguaggio Java e di creare in particolare una Desktop App. Tale decisione è stata concordata perché si è ritenuto opportuno avere un ambiente software sulla propria macchina di lavoro. Inoltre abbiamo pensato che, in futuro grazie alla progettazione del team, questo sistema software potrebbe essere esteso anche per altre piattaforme (ad esempio Web) grazie alla scalabilità utilizzata.

2) **DECISION MVC PATTERN:** il team ha deciso di impiegare il "pattern MVC" per rappresentare l'architettura del software e il component diagram ad esso collegato. Tale decisione e' stata concordata in primo luogo per organizzare il software in modo ottimale, in secondo luogo perche' grazie a questa tipologia di pattern possiamo organizzare il codice in modo piu' logico, dividendolo quindi in tre parti. Siamo cosi' in grado di facilitare la **scalabilita'** e la **manutenzione** dell'applicazione.

3) **ARCHITTETURA DATABASE:** Il team da deciso di utilizzare, per l'implementazione del database **monitoraggioambientale,** il **linguaggio d'interrogazione MySQL** che permette di sviluppare database che si basano sui modelli relazionali e sull'**engine INNODB**. Il **modello Relazionale** viene considerato attualmente il modello più semplice ed efficace, perché è più vicino al modo consueto di pensare i dati, e si adatta in modo naturale alla classificazione e alla strutturazione dei dati. Il modello relazionale è più intuitivo e più espressivo per la strutturazione dei dati, rispetto agli altri modelli come quello gerarchico e reticolare. E' stato deciso di adottare l'engine INNODB rispetto l'engine MyISAM perché è molto più rigido nell'integrità dei dati e nella referenzialità tra le relazioni mediante le ForeignKey, ed è molto più veloce nella fase di scrittura e lettura sulle tabelle visto che i dati presenti nelle relazioni Segnale e BackupValori saranno modificati in un lasso di tempo molto piccolo.

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

- ***REQUISITI FUNZIONALI:***
    1. **Dashboard (Non del tutto implementato)**: Dopo aver effettuato l'accesso dalla GUI della Login, a seconda dell'utente (Amministratore o Gestore dei Sensori) verrà visualizzata la Dashboard opportuna. **(Vedere GUI Requirements Fig.re 4 e 7)**;
    2. **Invio Segnale (Non del tutto implementato)**: La funzione dell'invio di un segnale da parte di un sensore sarà implementata nella Classe GestoreDati.java presente nel package Model.Components nello specifico nel metodo setSegnale(). Al suo interno avverrà la connessione al Database e la scrittura dei segnali nella relazione Segnale **(Vedi Modello Relazionale)**. Per permettere ai sensori di inviare periodicamente i segnali con le varie informazioni ambientali, viene generato un Thread per l'invio del segnale ogni minuto al momento del lancio dell'applicazione, nello specifico, nel lancio della Dashboard GestoreSensori nel file DashboardGestore.java situato nel package View.DashboardGestore;
    3. **Backup Parametri Sensore (Al momento non implementato)**: Utilizzando il metodo setBackup() presente nella Classe Sensore nel file Sensore.java, verrà implementata la scrittura del backup nella relazione BackupValori;
    4. **Ripristino Parametri Sensore (Al momento non implementato):** Il Gestore dei Sensori mediante un bottone chiamato "Ripristino" prenderà i dati del Backup del sensore su cui ha deciso di effettuare il ripristino e potrà, appunto, tramite i metodi set, modificare i parametri;
    5. **Interazione (Implementato):** A seconda della Dashboard che l'utente sta utilizzando, sarà possibile instaurare una comunicazione tra Amministratore e Gestore. Nel caso dell'Amministratore, andando nella sezione "Ticket", potrà visualizzare la lista dei ticket e rispondere ad un determinato messaggio inviatogli da un Gestore. **(Vedi GUI Requirements Fig.ra 6).** Viceversa, nel caso del Gestore dei Sensori, andando nella sezione "Ticket", potrà inviare un ticket per informare gli Amministratori di un determinata problematica. Nel momento in cui un Amministratore risponderà al ticket, tutti gli altri non potranno rispondere a quel determinato messaggio. Questo perché ogni ticket ha un campo lettura con valore booleano che viene settato ad 1 nel momento in cui un determinato Amministratore lo leggerà;
    6. **Registrazione Gestore Sensori (Implementato):** Tramite la Dashboard Admin, gli Amministratori potranno inserire uno o più Gestori dei Sensori assegnandogli le specifiche credenziali d'accesso. **(Vedi GUI Requirements Fig. 5).** Nello specifico cliccando sul bottone "Aggiungi Gestore".

- ***REQUISITI NON FUNZIONALI:***
    1. **DEPENDABILITY:**
        - **FAULT TOLERANCE (Non del tutto implementato):** *Il sistema deve continuare ad offrire i servizi anche se si è persa la funzionalità di una componente.* Questo vincolo viene soddisfatto, dal punto di vista implementativo, istanziando, ovvero effettuando operazioni di scrittura nella relazione Sensore di almeno due sensori per piano;
        - **SECURITY (Implementato)***: Il sistema deve essere in grado di proteggersi da attacchi esterni, accidentali o intenzionali.* Questo vincolo viene soddisfatto attraverso l'interfaccia Login dove ciascun utente deve immettere le proprie credenziali (Username e Password) per poter accedere al Sistema. Al momento del click sul bottone "Login" **(Vedi GUI Requirements Fig.re 2 e 3)**, verrà effettuato un controllo sul Tipo per verificare per prima cosa se è un Amministratore o un Gestore e, nel caso in cui risulti un Gestore, un controllo sul 'Tipo' di Gestore (Edificio, Area, Città);
    2. **SCALABILITY (Non del tutto implementato):** *Il sistema garantisce un'architettura scalabile per supportare future espansioni.* Questo vincolo è stato soddisfatto attraverso l'utilizzo dei bottoni 'Aggiungi Sensore' ed 'Aggiungi Admin/Gestore' nelle corrispondenti Dashboard Admin e Dashboard Gestore Sensori **(Vedi GUI Requirements Fig.re 5 e 7)**.
    3. **USABILITY (Implementato):** *Il sistema deve offrire una User-Experiencie-Friendly*. Questo vincolo verrà soddisfatto dalle Dashboard di ciascun utente che può utilizzare il Sistema. **(Vedi GUI Requirements)**
    4. **PERFORMANCE (Non Implementato):** *Il sistema deve risultare efficiente e deve lavorare con tempi d'esecuzione accettabili.* Dato che il team sta realizzando un prototipo del software, ancora non si è grado di stimare performance che risultino essere efficienti in termini di Tempi d'esecuzione ed Occupazione della Memoria.

G. Effort Recording

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

Il team ha deciso di utilizzare ***ProjectLibre*** per modellare il Pert Chart:

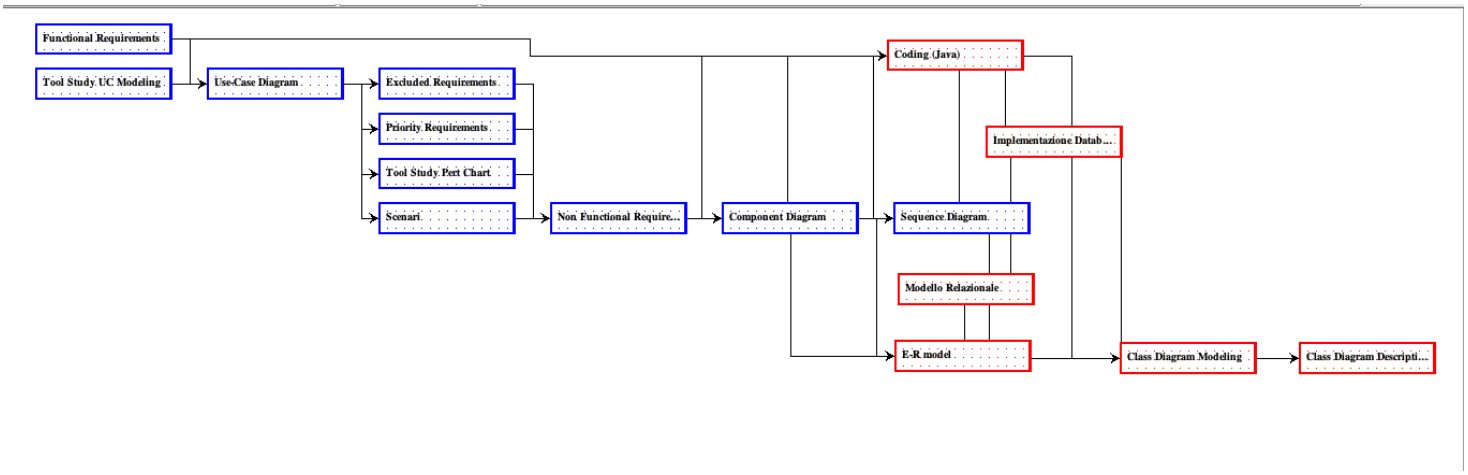

Fig. 15: Pert Charts Monitoraggio Ambientale

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

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

Categorization*When logging the time spent on the project, please create different sub- categories. Specifically, it is important to clearly distinguish between two main categories: the time spent for "****learning****" (the modeling languages, the tools, etc.) from the time needed for "****doing****" (creating the models, taking the decisions, …). Learning tasks are in fact costs to be paid only once, while doing costs are those that will be repeated through the project.*
For each category, please define sub-categories. Examples follow. You may add other sub-categories you find useful.

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

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

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

TASK	*DOING*	*LEARNING*
Deliverable Review	4 H	
ER Modeling	8 H 30 MIN	
Coding Java	24H 50 MIN	
Component Diagram Review	2 H	
Appendix Description	2 H	
Relationship Model Description	2 H 30 MIN	
Business Logic Requirements	1 H	
DB Requirements	1 H	
Class Diagram Modeling and Description	9 H	
Implementation Database	5 H 30 MIN	
Class Diagram Modeling	9 H 30 MIN	
GUI Requirements	3 H	
Class Diagram Description	1 H 30 MIN	
Pert Charts Modeling	2 H 40 MIN	
Explain how the RFs e NFRs are satisfied by design	4 H	
ER Model Description	2 H 30 MIN	
TOTAL TIME	*83 H 30 MIN*	

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

Appendix. Code

*<Report in this section a **documented** version of the produced code. I do not need a copy&cut of your code, but rather an explanation of how your code satisfies the Functional and Non functional requirements.<Show some screenshots of the code behavior>*

<please upload your executable code in the dropbox folder>

Il sistema per poter funzionare ha bisogno dei seguenti programmi:

- - MySQL Community Server: (solo utenti MacOS)
  https://dev.mysql.com/downloads/mysql/
- - MySQL Workbench: (solo utenti MacOS)
  https://dev.mysql.com/downloads/workbench/
- Java SE Development Kit 8: (per tutti gli utenti)
  https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- MySQL Installer: (solo utenti Windows)
  https://dev.mysql.com/downloads/windows/installer/8.0.html

Per poter usare il nostro sistema:

Per sistema operativo MacOS:

- Avviare MySQL Community Server tramite il menù "preferenze di sistema" su sistema cliccando sull'icona del programma e successivamente su "start my sql server". Inserire la password scelta in fase di installazione (consigliamo "root" per entrambi i campi) ed accertarsi del corretto funzionamento del Server tramite le spie verdi.

- Avviare successivamente MySQL Workbench ed eseguire una scansione dei server disponibili, verrà trovato il server MySQL precedentemente installato. Cliccare sul server ed inserire le password scelte per eseguire la connessione.
  Nella seconda schermata cliccare su:
  Data Import -> import from self-contained file -> tramite il menù di ricerca selezionare il server "monitoraggioambientale.sql" -> infine cliccare su "start import" in basso a destra della schermata.
  Una volta effettuato con successo queste operazioni sarà possibile visualizzare il nostro server "monitoraggio ambientale" sulla sinistra della finestra corrente.

- Avviare il terminale ed eseguire la riga di comando "java -jar softing.jar" per lanciare il programma che deve trovarsi nella stessa Path di esecuzione del comando.
  Verrà mostrata una finestra di login, da qui è possibile inserire le credenziali per accedere alla Dashboard di appartenenza.

Su sistema operativo Windows:

1. Scaricare ad avviare l'installer di MySQL e seguire la procedura guidata per installare sia il server che il Workbench sulla propria macchina, quando richiesto selezionare "Developer Default"

    1.1. Una volta pronto verrà chiesto di configurare i prodotti appena installati, scegliere l'impostazione "Standalone MySQL Server"

    1.2. Successivamente nella finestra "Type and Networking" passare oltre senza modificare alcuna impostazione

    1.3. Scegliere infine delle credenziali (è consigliato usare "root" per entrambi i campi) che verranno usate dall'applicativo ad ogni accesso

    1.4. È possibile lasciare le successive impostazioni di default

- Se tutto è andato a buon fine sarà possibile accedere al database tramite le stesse modalità descritte precedentemente per MacOS.

- Per le future connessioni tramite il menù "start" dentro la cartella "MySQL" è possibile avviare nuovamente tutti i software necessari

- Mantenendo sia il server MySQL che il MySQL Workbench in esecuzione basterà eseguire con un doppio click sull'applicativo softing.jar per lanciare il programma.

Descrizione sistema:

Una volta avviato il software ci si troverà davanti una pagina di login, qui sarà possibile accedere alla propria dashboard inserendo le credenziali corrette:

Come amministratore:

Username: Muccini23
Password: muc123

Come Gestore dei sensori:

Username: Tobia56
Password: tob56