"Software Engineering"

Course

a.a. 2018-2019

Template version 1.0

Deliverable #1

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

Dashboard Monitoraggio Ambientale

Date	03/12/2018
Deliverable	D1
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
Requisito Non Funzionale Survivability non importante	19/11/2018	19/11/2018	Dopo un'attenta riflessione, il team ha deciso di non considerare la Survivability uno dei requisiti non funzionali più importanti.
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.
Ridenominazione dello Use Case Sensori a Rischio	12/11/2018	30/11/2018	Dopo la creazione del Sequence Diagram inerente allo scenario INVIO SEGNALE, il team ha deciso di rinominare lo use-case INFO SUI SENSORI con SENSORI A RISCHIO visto che il nome precedentemente usato non coglieva il reale obiettivo dello use-case.

A. Requirements Collection

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 inviare 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:*

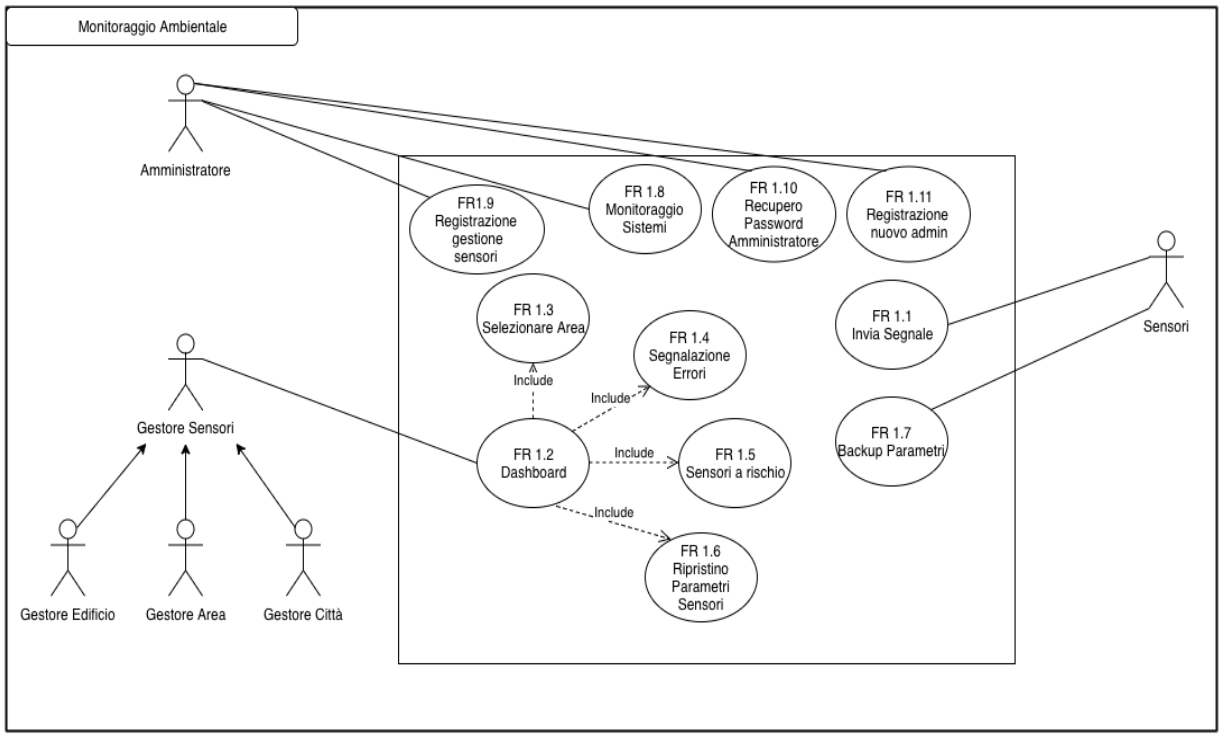


Fig. 1: Use-Case Diagram Monitoraggio Ambientale

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).

SCENARI

SCENARIO 1: PRIMO AVVIO

Mediante la Dashboard, il *Gestore* dei sensori 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.

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>

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 DATABASE****: Il sistema dovrà includere una ridondanza sul database in modo tale da non rischiare una perdita di dati, ovvero dobbiamo avere un database di backup.*
- ***N.F.R. 1.3 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 STORAGE*: Il sistema dovrà supportare lo storage and il processing di almeno 150.000 messaggi al minuto.*

N.F.R. 4 USABILITY*: Il sistema deve offrire una user-experience-friendly. Ovvero, deve risultare facile da utilizzare. (Es: Ripristino dei parametri dei sensori)*

N.F.R. 5 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.
- 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.

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.1 The static view of the system: Component Diagram

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

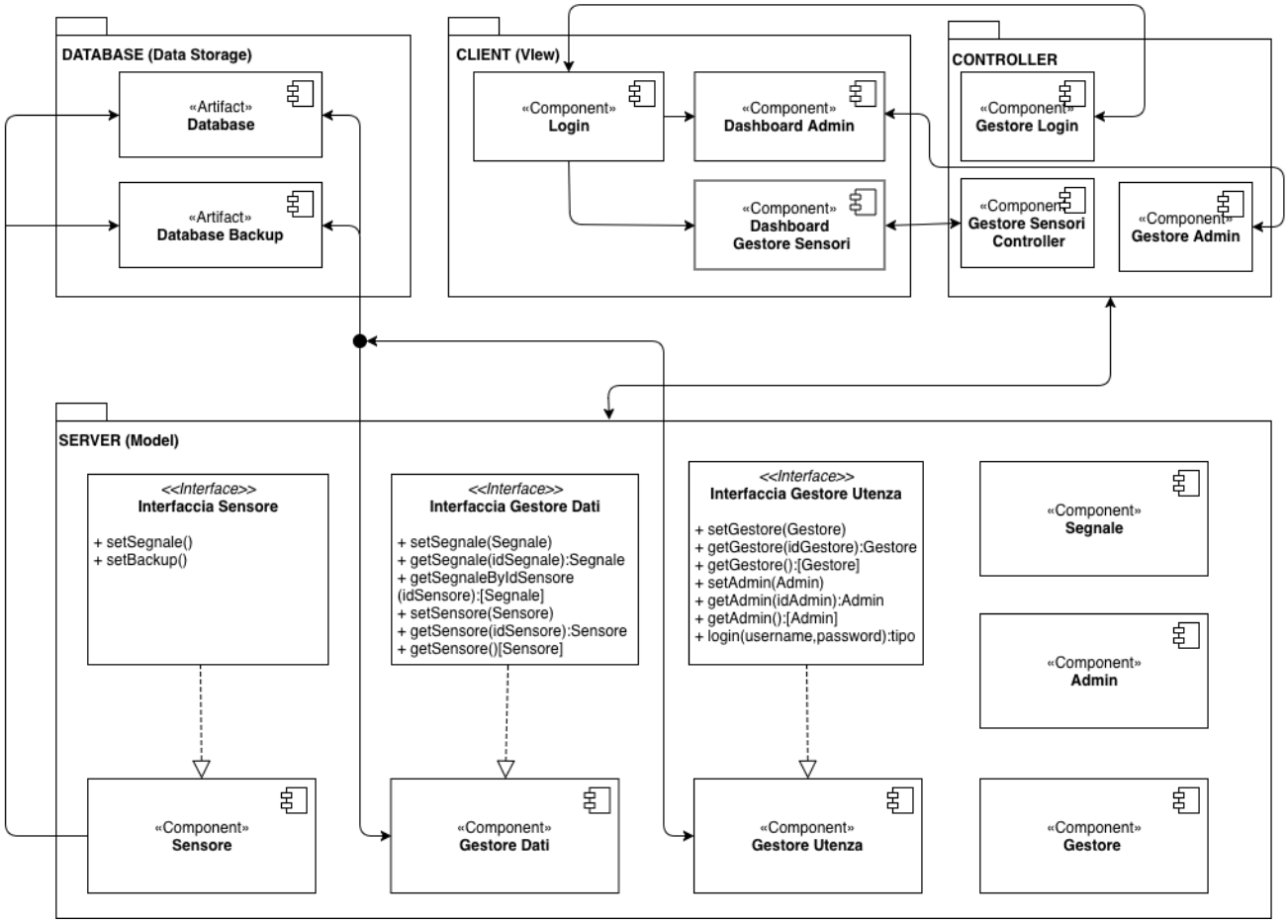


Fig. 2: Component Diagram Monitoraggio Ambientale

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 software che 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:** conterrà due componenti denominate "*database*" e "*database backup*"; hanno il compito di immagazzinare i dati e garantire che gli stessi non vengano persi.
- **CLIENT (GUI):** conterrà tre componenti denominate: "*login*", "*dashboard admin*", "*dashboard gestore sensori*". 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 sensori*" 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: "*gestore login*", "*gestore sensori controller*" e "*gestori admin*".
- **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.*

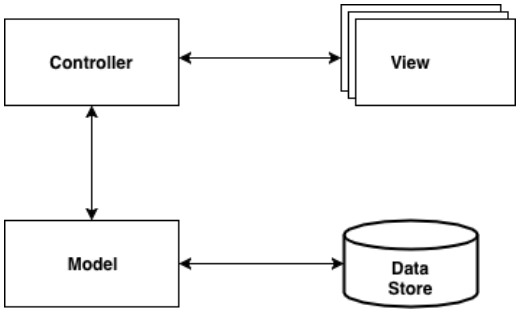


Fig. 3 Schema del Pattern MVC

La 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*”, “*Admin*”, “*Gestore*” 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

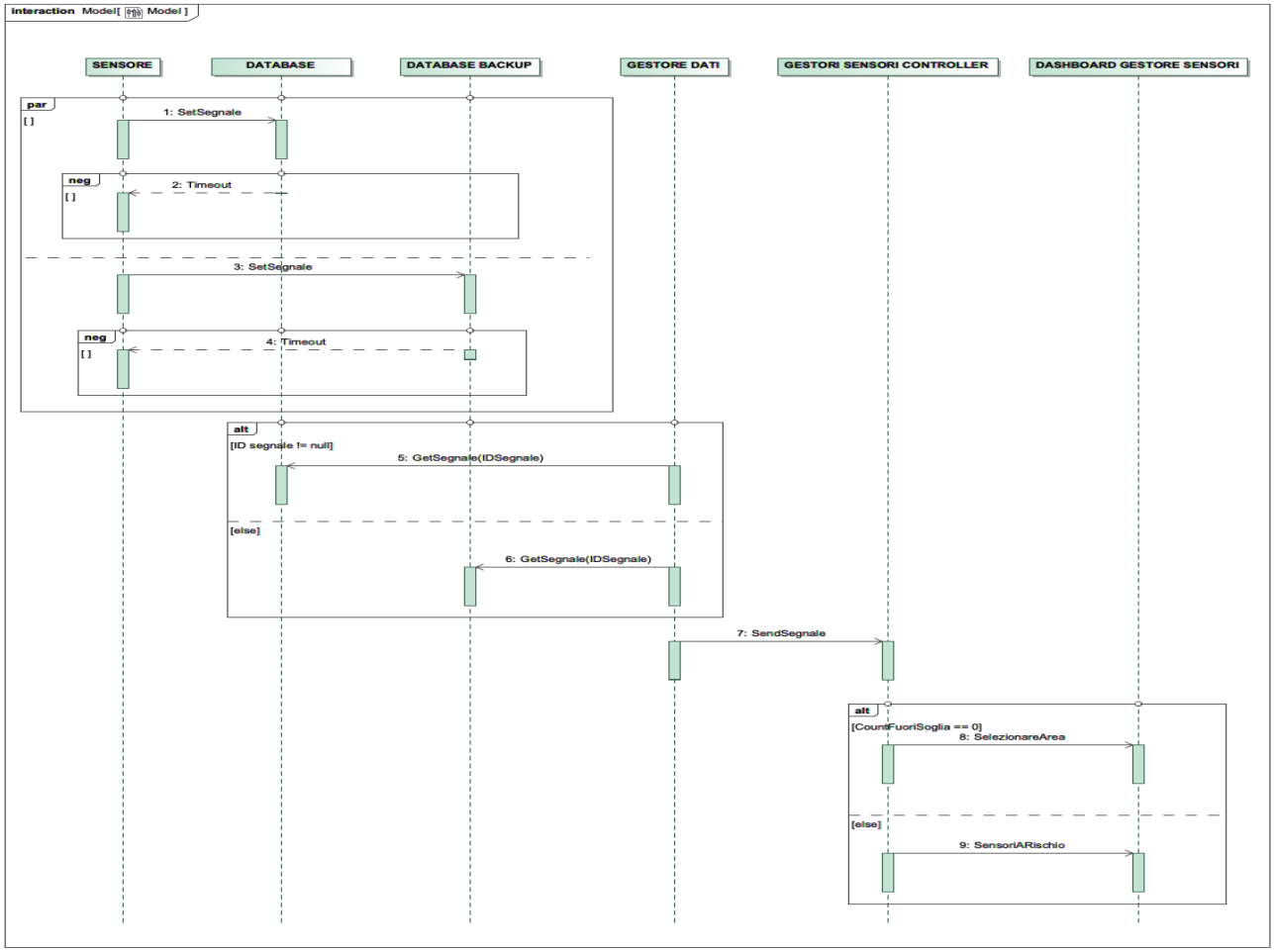


Fig. 4 Sequence Diagram inerente allo Scenario 2: INVIO SEGNALE

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

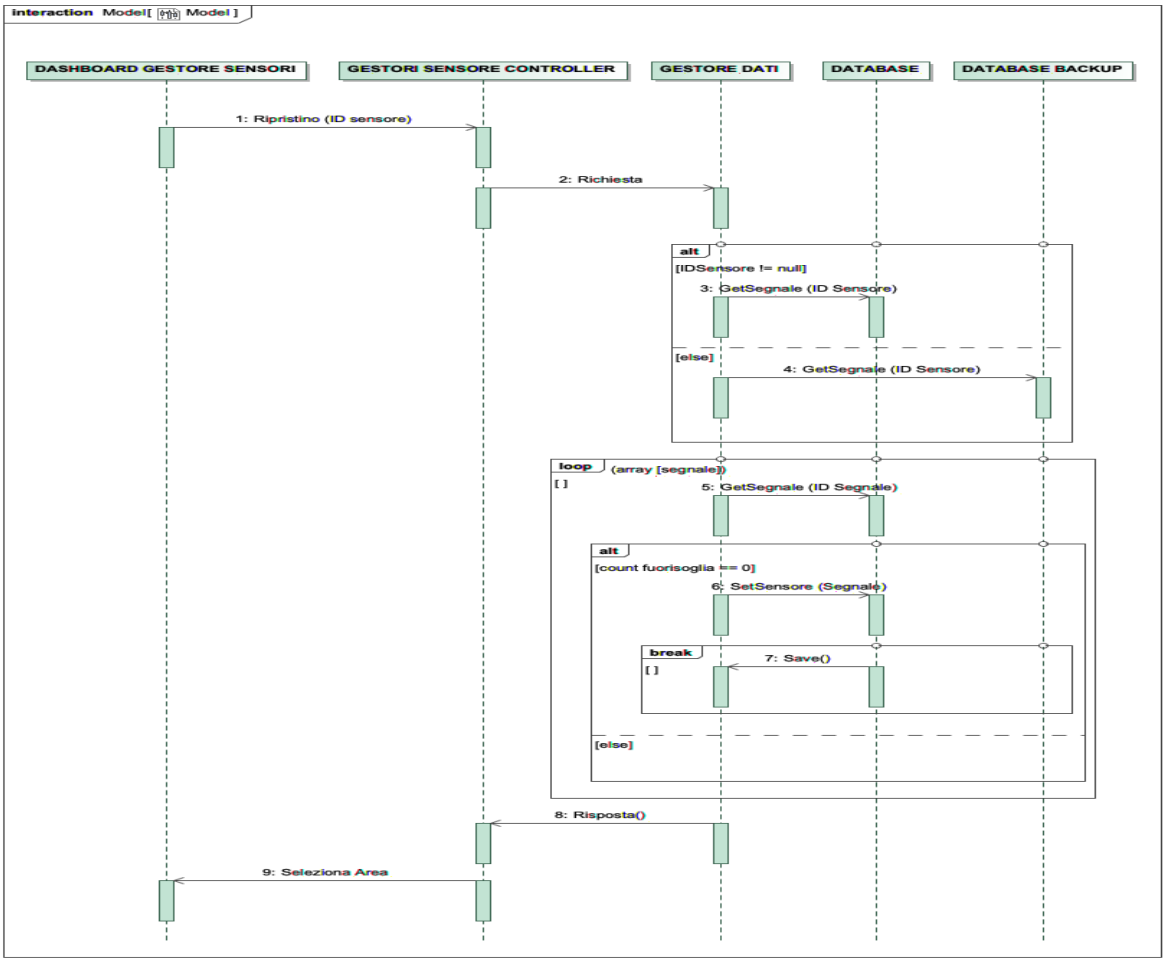


Fig. 5 Sequence Diagram inerente allo Scenario 4: RIPRISTINO

Il Gestore mediante la **Dashboard GestioneSensori** 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".

E. Design Decisions

1) FUNCTIONAL REQUIREMENTS:

- DECOMPOSIZIONE FUNZIONALE: Il sistema viene scomposto in funzionalità, ordinato per operazione e decomposto in altri piccoli moduli;
- DECOMPOSIZIONE OBJECT-ORIENTED: Il sistema viene scomposto in classi, ogni classe è un'entità importante nel dominio dell'applicazione, le classi possono essere scomposte in piccole classi.

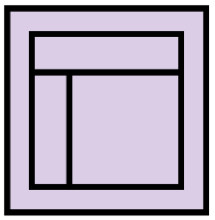
SCELTA:
Il team ha deciso di optare per la Decomposizione Funzionale perchè ritenuto più consono per svolgere tale progetto.

2) NON FUNCTIONAL REQUIREMENTS: Il team ha deciso di dar più importanza ai seguenti requisiti non funzionali essendo essenziali per il corretto funzionamento del sistema:

- Dependability;
- Scalability;
- Storage;
- Usability;
- Performance.

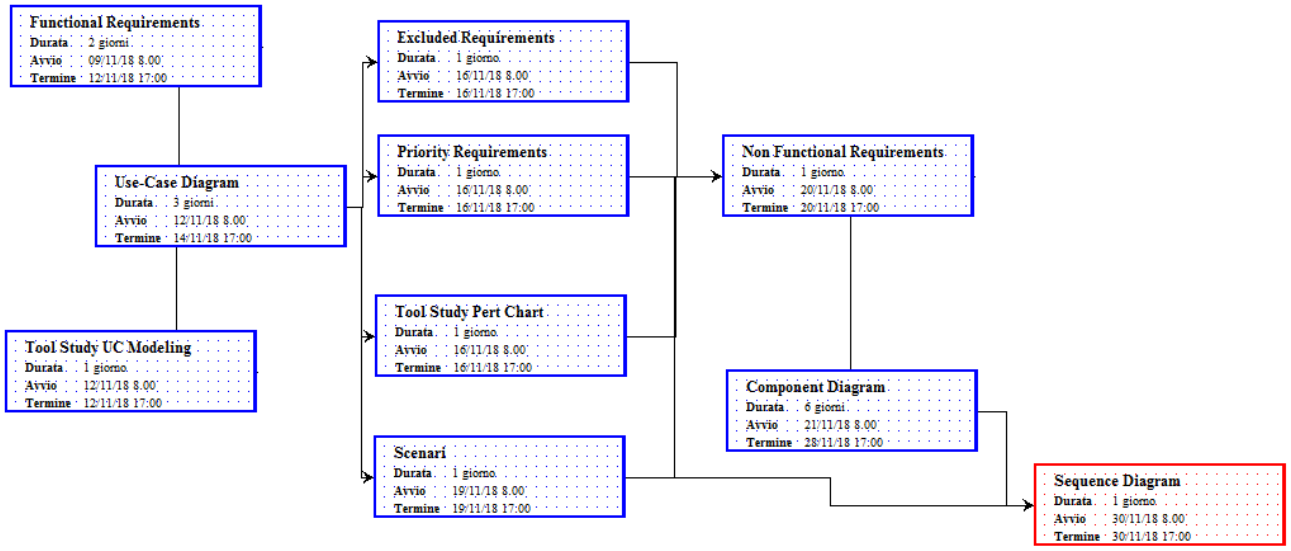
3) DECISION PROGRAMMING LANGUAGES: Il team ha deciso per lo sviluppo del software Monitoraggio Ambientale di utilizzare il linguaggio Java e di creare in particolare una Desktop App. Tale decisione è stata concordata perché si è ritenuto opportuno avere un ambiente software sulla propria macchina di 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.

G. Effort Recording



PERT

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



Logging

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

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

Categorization

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

Learning ***Doing:***

- ***Requirements Engineering***
- ***Non functional Requirements***
- ***Use Case Diagrams***
- ***Tool study***
- ***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 Study	1 h	2 h
Requirements Discovery	15 h 40 min	
Requirements Modeling (UC diagrams)	17 h 30 min	
Tool Study		4 h 30 min
Use-Case Cockburn Modeling	50 min	
Prioritization	1h 15 min	
List of Challenging/Risky Requirements of Tasks	35 min	
List of not-doing Requirements	2 h	
Scenery Discovery	2 h 40 min	

Architettura del Software	30 min	
List of Non-Functional Requirements	3 h 30 min	
Component Diagram Study		13 h 20 min
Component Diagram Discovery	2 h	
Component Diagram Review	18 h 20 min	
Modeling Component Diagram	4 h	
Component Diagram Description	6 h	
MVC Study		4 h
Magic Draw Study		1 h
Sequence Diagram Modeling and Description	26h 20 min	
TOTAL TIME	***102 H 10 MIN***	***24 H 50 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