**“Software Engineering”**

**Course**

**a.a. 2017-2018**

**Template version 1.0**

**Deliverable #2**

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

**Industrial Robot dashboard**

|  |  |
| --- | --- |
| **Date** | 27/01/2018 |
| **Deliverable** | 3 |
| **Team (Name)** | A.I.M. |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| Davide Morelli | *243745* | *davide.morelli1@student.univaq.it* |
| Giammarco Cupido | *244239* | *jacklambretta93@hotmail.it* |
| Iari Icaro | *243742* | *iari.icaro@student.univaq.it* |
| Paolo Tramontozzi | *242900* | *paolo.tramontozzi@student.univaq.it* |
| Stefano Martella | *243848* | *stefano.martella@student.univaq.it* |

Table of Contents of this deliverable

Sommario

[List of Challenging/Risky Requirements or Tasks 4](#_Toc504842202)

[A. Requirements Collection 6](#_Toc504842203)

[A.1 Functional Requirements 6](#_Toc504842204)

[A1.1 GUI Requirements 6](#_Toc504842205)

[A1.2 Business Logic Requirements 6](#_Toc504842206)

[A1.3 DB Requirements 7](#_Toc504842207)

[Use case diagram: 8](#_Toc504842208)

[Use case 1 9](#_Toc504842209)

[Use case 2 10](#_Toc504842210)

[Use case 3 11](#_Toc504842211)

[Use case 4 12](#_Toc504842212)

[Use case 5 13](#_Toc504842213)

[Use case 6 14](#_Toc504842214)

[Use case 7 15](#_Toc504842215)

[A.2 Non Functional Requirements 16](#_Toc504842216)

[B2 Dependability requirements 16](#_Toc504842217)

[B3 Organizational requirements 16](#_Toc504842218)

[B4 Scalability requirements 17](#_Toc504842219)

[A.3 Content 17](#_Toc504842220)

[A.4 Assumptions 18](#_Toc504842221)

[A.5 Prioritization 18](#_Toc504842222)

[B. Analysis Model 20](#_Toc504842223)

[Boundary Objects: 20](#_Toc504842224)

[Entity Objects: 21](#_Toc504842225)

[Controller Objects: 21](#_Toc504842226)

[C. Software Architecture 22](#_Toc504842227)

[C.1The static view of the system: Component Diagram 22](#_Toc504842228)

[C.2 The dynamic view of the software architecture: Sequence Diagram 25](#_Toc504842229)

[D. ER Design 28](#_Toc504842230)

[E. Class Diagram of the implemented 29](#_Toc504842231)

[F. Design Decisions 31](#_Toc504842235)

[G. Explain how the FRs and the NFRs are satisfied by design 34](#_Toc504842236)

[A1.1 GUI Requirements 34](#_Toc504842237)

[A1.2 Business Logic Requirements 35](#_Toc504842238)

[A1.3 DB Requirements 35](#_Toc504842239)

[B1.1 GUI NFRequirements 36](#_Toc504842240)

[B1.2 Business Logic NFRequirements 36](#_Toc504842241)

[H. Effort Recording 39](#_Toc504842242)

[GANTT 39](#_Toc504842243)

[Appendix. Code 41](#_Toc504842244)

# 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** |
| Gestione dei robots e dei clusters down. | 16/11/2017 | 16/11/2017 | Abbiamo scelto di utilizzare un contatore per i robots che tenga traccia del numero dei segnali down(un robot è down se ha almeno un segnale down) e un contatore per i clusters che tenga traccia del numero dei robots down al suo interno(un cluster è down se ha almeno un robot down). Questo ci ha permesso di ridurre la complessità del problema, infatti, non è necessario sapere ***quali*** sono i segnali(o robots nel caso dei clusters) down ma ***quanti***. |
| Difficoltà nella realizzazzione del simulatore per inviare i messaggi. | 12/12/17 | 14/12/17 | La prima implementazione realizzata con uno script bash risultava lenta nell’inviare i messagi tramite HTTP, questo ci impediva di testare la reale velocità del sistema. Utilizzando Java abbiamo raggiunto il risultato desiderato. |
| Tecnologie da usare | 11/12/17 | 11/12/17 | Abbiamo scelto di usare le tecnologie apprese durante il percorso di studi in quanto soddisfano i requsiti per lo sviluppo del progetto. Per il database abbiamo preferito l’utilizzo di MongoDB piuttosto che MySQL dopo aver effettuato diversi test, i risultati sono elencati nello specifico nella sezione delle Design Decision. |
| Difficoltà nel tenere il conto del down time dell’ultima ora e di conseguenza difficoltà nel calcolo dell’IR. | 16/12/17 | 16/12/17 | Salviamo la data e l’ora di quando un robot va down(T1); quando il robot torna up(T2) sottraendo T2 a T1 si ottiene il tempo di down(DT1) da T1. La somma dei tempi DTi, dell’ultima ora, cosi calcolati ci permette di ricavare la percentuale di IR.  È stato adottato lo stesso principio per i clusters, in particolare l’inizio del down time(T1) coincide con il passaggio da 0 a 1 del contatore dei robots down in un determinato cluster mentre la fine del down time (T2) coincide con il passaggio da 1 a 0 del contatore dello stesso cluster. |
| Organizzazione del gruppo. | 11/12/17 | 11/12/17 | Abbiamo deciso di organizzare degli incontri di gruppo durante la settimana. |

# A. Requirements Collection

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

## A.1 Functional Requirements

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

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

### A1.1 GUI Requirements

* Il Sistema deve prevedere una GUI dalla quale poter visualizzare le percentuali di IR di robots e clusters.
* La GUI deve presentare una dashboard accessibile tramite web, essa deve

permette all’ingegnere di accedere alla sezione dei clusters e a quella dei robots per monitorarne le percentuali di IR.

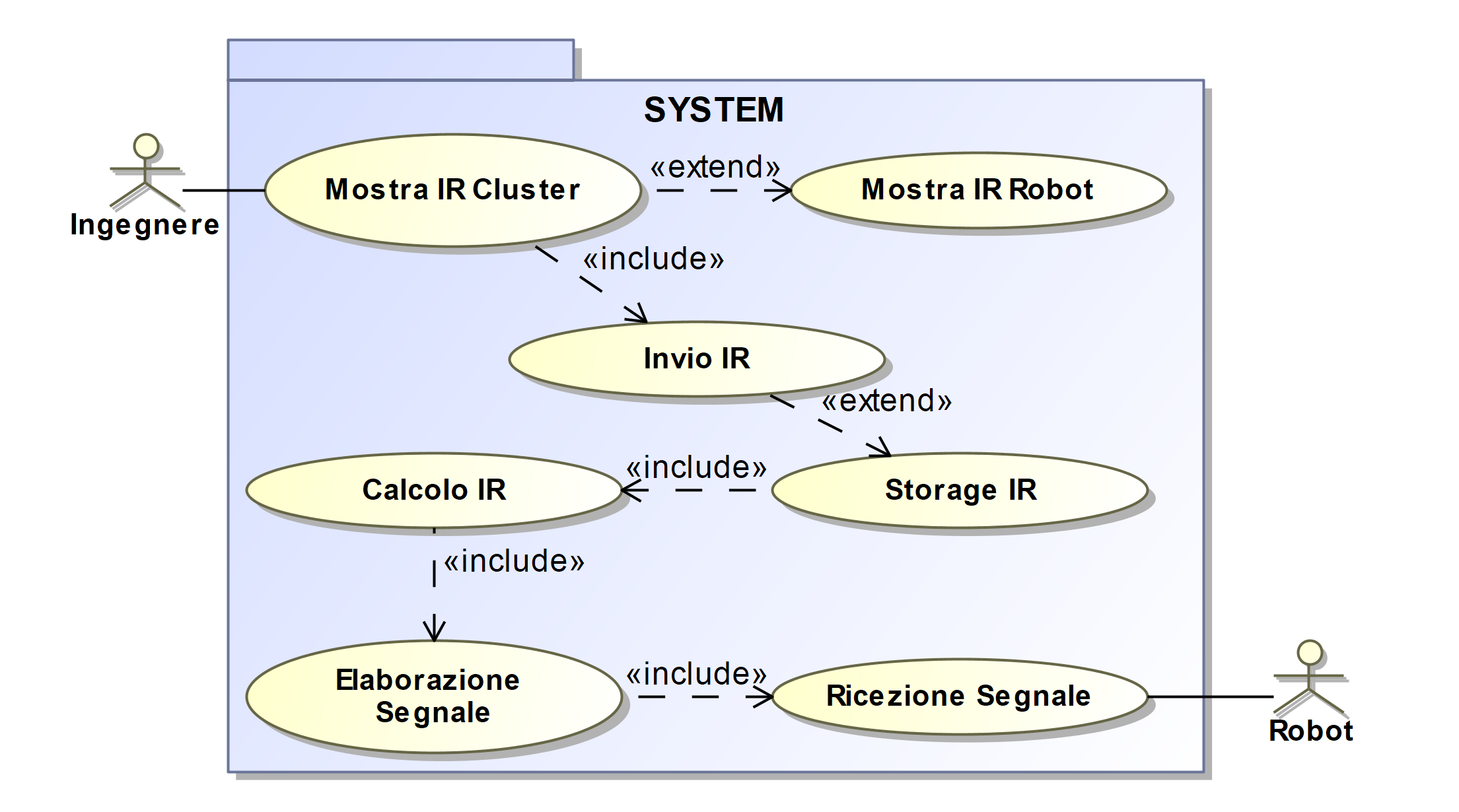
* + La sezione dei cluster deve permettere all’ingegnere di visualizzare i valori di IR di ogni cluster.
  + La sezione dei robot deve permettere all’ingegnere di visualizzare i valori di IR di ogni robot.
* Deve essere possibile navigare attraverso le sezioni muovendosi avanti e indietro nella dashboard.

### A1.2 Business Logic Requirements

* Il sistema nella sua parte logica deve avere una componente che si occupi della gestione dei messaggi in arrivo dall’ESB e che proceda al loro storage.
* Il sistema nella sua parte logica deve avere una componente che si occupi di aggiornare i records nel database, eseguire queries su di esso per poi strutturare i dati che verranno inviati ai clients.
* Il sistema nella sua parte logica deve avere una componente che si occupi di inviare ai clients che ne fanno richiesta(ingegneri) i dati relativi alle percentuali di IR.

### A1.3 DB Requirements

* Il sistema deve prevedere lo storage degli IR dei robots e dei clusters su apposito database per garantire una maggiore affidabilità.
* Il database deve essere collegato alla parte logica del sistema per poter essere interrogato e fornire le informazioni utlili sui robots e sui clusters.

**

# Use case diagram:

[*Figura 1*](https://drive.google.com/open?id=1LuG35waMJmEkNBSheF_qunpYtDwKHahs)

|  |  |  |
| --- | --- | --- |
| USE CASE 1 | **Ricezione segnale** | |
| Goal in Context | Il robot invia un segnale relativo al suo cambiamento di stato e il sistema lo riceve. | |
| Primary | Robot. | |
| Preconditions | Il robot deve avere una connessione al server per poter inviare il messaggio tramite HTTP. | |
| Success End Condition | Il robot invia al sistema il messaggio con le informazioni sul suo cambiamento di stato e il sistema è in grado di riceverlo per poi elabolarlo. | |
| Failed End Condition | Il robot non è in grado di inviare il messaggio. | |
| Trigger | I paramentri di una componente del robot escono al di fuori del range di specifica di quella determinata componente. | |
| DESCRIPTION | Step | Action |
|  | 1 | Un robot invia il messaggio contenente le informazioni sul suo cambiamento di stato. |
|  | 2 | Il sistema riceve il messaggio di cambiamento di stato del robot. |

|  |  |  |
| --- | --- | --- |
| USE CASE 2 | **Elaborazione messaggio** | |
| Goal in Context | Il sistema dopo avere ricevuto il messaggio di cambio stato dal robot si occupa di elaborarlo. | |
| Primary | - | |
| Preconditions | Il sistema deve aver ricevuto un messaggio da un robot. | |
| Success End Condition | Il sistema analizza e interpreta il messaggio ricevuto. | |
| Failed End Condition | Il sistema non è in grado di processare il messaggio ricevuto. | |
| Trigger | Il sistema ha ricevuto un messaggio dal robot che non è stato ancora processato. | |
| DESCRIPTION | Step | Action |
|  | 1 | Un robot invia il messaggio contenente le informazioni sul suo cambiamente di stato. |
|  | 2 | Il sistema riceve un messaggio da un robot. |
|  | 3 | Il messaggio viene elaborato. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE 3 | **Calcolo IR** | | | |
| Goal in Context | Dopo avere elaborato un messaggio il sistema procede a calcolare la percentuale di IR. | | | |
| Primary | - | | | |
| Preconditions | Il sistema deve avere elaborato un messaggio in arrivo. | | | |
| Success End Condition | Viene calcolata la percentuale di IR del robot e del cluster ai quali il messaggio faceva riferimento. | | | |
| Failed End Condition | Il sistema non è in grado di calcolare la percentuale di IR. | | | |
| Trigger | Il sistema ha ricevuto un messaggio e ha concluso la sua elaborazione. | | | |
| DESCRIPTION | Step | | Action | |
|  | 1 | | Un robot invia un messaggio con le informazioni sul suo cambiamento di stato. | |
|  | 2 | | Il sistema riceve il messaggio. | |
|  | 3 | | Il messaggio viene processato e analizzato. | |
|  | 4 | | Viene calcolata la percentuale di IR del robot e del cluster a cui il messaggio faceva riferimento. | |
| USE CASE 4 | | **Storage IR** | | |
| Goal in Context | | La percentuale di IR viene salvata. | | |
| Preconditions | | Il sistema deve aver calcolato la percentuale di IR di un determinato robot e il cluster di appartenenza.. | | |
| Success End Condition | | I dati vengono inseriti all’interno del database. | | |
| Failed End Condition | | Il sistema non è in grado di procedure al salvataggio dei dati. | | |
| Trigger | | Viene calcolata la percentuale di IR di un robot e del suo cluster di appartenenza. | | |
| DESCRIPTION | | Step | | Action |
|  | | 1 | | Un robot invia un messaggio |
|  | | 2 | | Il sistema riceve il messaggio. |
|  | | 3 | | Il messaggio viene processato. |
|  | | 4 | | Viene calcolata la percentuale di IR. |
|  | | 5 | | I valori dell’IR vengono salvati. |

|  |  |  |
| --- | --- | --- |
| USE CASE 5 | **Invio percentuali di IR** | |
| Goal in Context | Le percentuali di IR vengono inviate ai clients che ne fanno richiesta. | |
| Primary | - | |
| Preconditions | I dati relativi alle percentuali di IR devono essere stati precedentemente salvati. | |
| Success End Condition | I dati vengono inviati correttamente ai clients che ne hanno fatto richiesta. | |
| Failed End Condition | Il sistema non è in grado di inviare i dati. | |
| Trigger | Uno o più clients richiedono i dati relativi all’IR di robots e clusters. | |
| DESCRIPTION | Step | Action |
|  | 1 | L’ingegnere si collega al sistema per visualizzare le percentuali di IR.. |
|  | 2 | I dati vengono strutturati e preparati all’invio. |
|  | 3 | I dati vengono inviati ai clients che ne hanno fatto richiesta. |

|  |  |  |
| --- | --- | --- |
| USE CASE 6 | **Visualizzazione IR Clusters** | |
| Goal in Context | L’ingegnere interagisce con il sistema e si aspetta di visualizzare l’IR relativo ai clusters. | |
| Primary | Ingegnere. | |
| Preconditions | L’ingegnere deve avere una connessione alla dashboard. | |
| Success End Condition | L’ingegnere visualizza in rosso i clusters con percentuale di IR > del 40 % e in verde gli altri. | |
| Failed End Condition | L’ingegnere riceve un messaggio di errore. | |
| Trigger | L’ingegnere seleziona l’area dove sono presenti i clusters di cui vuole conoscere il livello di IR. | |
| DESCRIPTION | Step | Action |
|  | 1 | L’ingegnere si collega alla dashboard. |
|  | 2 | Il sistema invia i dati relativi all’IR all’ingegnere. |
|  | 3 | L’ingegnere seleziona un’area sulla dashboard. |
|  | 4 | L’ingegnere visualizza le percentuali di IR dei clutsers presenti nell’area selezionata. |

|  |  |  |
| --- | --- | --- |
| USE CASE 7 | **Visualizzazione IR Robots** | |
| Goal in Context | L’ingegnere interagisce con il sistema e si aspetta di visualizzare l’IR relativo ai robots. | |
| Primary | Ingegnere. | |
| Preconditions | L’ingegnere deve avere una connessione alla dashboard. | |
| Success End Condition | L’ignegnere visualizza in rosso i robots con percentuale di IR > del 40 % e in verde gli altri. | |
| Failed End Condition | L’ingegnere riceve un messaggio di errore. | |
| Trigger | L’ingegnere seleziona l’area e successivamente un cluster dove sono presenti i robots di cui vuole conoscere il livello di IR. | |
| DESCRIPTION | Step | Action |
|  | 1 | L’ingegnere si collega alla dashboard. |
|  | 2 | Il sistema invia i dati relativi all’IR all’ingegnere. |
|  | 3 | L’ingegnere seleziona un’area sulla dashboard. |
|  | 4 | L’ingegnere seleziona un cluster nell’area selezionata. |
|  | 5 | L’ingegnere visualizza le percentuali di IR dei robots presenti nel cluster selezionato. |

## A.2 Non Functional Requirements

*<List and describe here the* ***most important*** *non functional requirements.>*   
B1 Product requirementsB1.1 Performance requirements

* La dashboard dovrà aggiornarsi automaticamente almeno ogni 5 minuti.  
  Requisito da specifica. Vincolo a cui il sistema deve sottostare per una corretta monitorizzazione.  
  + Un tempo di refresh sotto al minuto è considerato preferenziale.
* Il sistema deve essere in grado di gestire contemporaneamente N segnali, dove N è il numero di robots che il sistema gestisce.
* Il sistema deve essere in grado di gestire fino a 100 ingegneri contemporaneamente che fanno richiesta per visualizzare le percentuali di IR.

B2 Dependability requirements B2.1 Availability requirements

* Il sistema deve fornire un'affidabilità minima di up time del 98%.  
  Il sistema deve essere di grande affidabilità in modo da non compromettere il funzionamento di altri sistemi collegati ad esso.

#### B2.2 Safety requirements

* Il sistema deve essere in grado di gestire tutti i segnali che vengono inviati dai robots, la mancata elaborazione di anche uno solo di essi può comportare il malfunzionamento dell’intero sistema.

B3 Organizational requirements B3.1 Development requirements

* La versione finale del sistema deve essere consegnata entro e non oltre il 27/01/2018.

### **B4 Scalability requirements**

* Il sistema deve prevedere ampliamenti futuri come l’aumento del numero dei clusters e di conseguenza della quantità di robots da gestire.

## A.3 Content

*<Describe the* ***data provenance*** *(use of external API, web service, DB …)>*

**Per semplicità dividiamo il progetto in tre parti:**

**Parte web:** Abbiamo scelto di utilizzare ***HTML5***, ***CSS*** e ***JavaScript***.

**Parte logica:** Abbiamo scelto di utilizzare ***Java***(9.0.1). Inoltre importeremo librerie esterne a java per la gestione del formato JSON(java-json.jar) e la creazione del web socket(java-websocket.jar).

**Parte DB:** Per la parte del database abbiamo scelto di utilizzare ***MongoDB*** e di conseguenza il driver necessario(mongo-java-driver.jar) per permettere la comunicazione tra parte logica e il database stesso.  
  
**Il progetto è stato sviluppato utilizzando un IDE che offre supporto multi-linguaggio tramite componenti aggiuntivi, Eclipse.**

## A.4 Assumptions

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

* Il budget per l’hardware è illimitato.
* I robot non possono essere interrogati.
* Al tempo 0 i robots sono considerati UP.
* **I messaggi inviati dai robot sono in formato JSON e sono strutturati nel seguente modo:**{  
   “robot\_id” : <ID del robot>,   
   “cluster\_id” : <ID del cluster>,  
   “area\_id” : <ID dell’area>,  
   “signal\_state” : <UP | DOWN>,  
   “message\_time” : <Data e ora in cui il messaggio è inviato>  
  }

**La scelta di formattare la data e l’ora nel messaggio ci permette di calcolare gli IR indipendentemente dalla velocità che il sistema ha nell’elaborare i messaggi ricevuti dai robots.**

A.5 Prioritization*<please list your requirements in priority>*

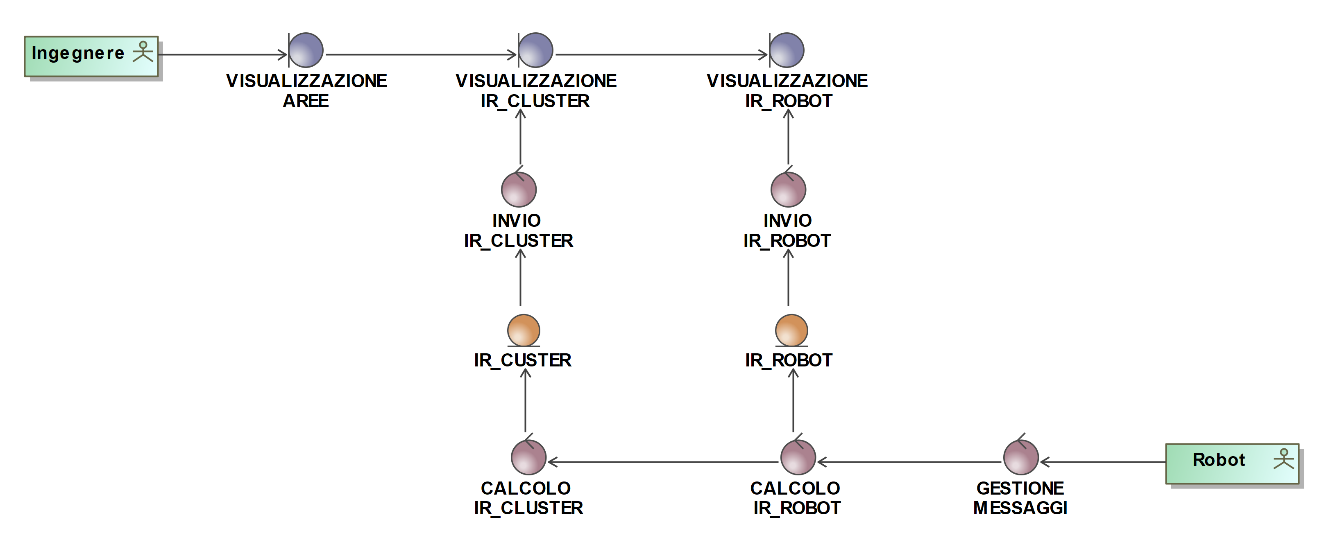
* Il sistema deve prevedere, nella sua una parte logica, una componente che si occupi di processare i messaggi in arrive dall’ESB e di procedere al loro storage.
* Il sistema nella sua parte logica deve avere una componente che si occupi di aggiornare i records nel database, eseguire quries su di esso per poi strutturare i dati che verranno inviati ai clients.
* Il sistema nella sua parte logica deve avere una componente che si occupi di inviare ai clients che ne fanno richiesta(ingegneri) i dati relativi alle percentuali di IR.
* Il Sistema deve prevedere un database dove salvare gli IR dei rispettivi robot e cluster per poi essere visualizzati sulla dashboard dall’ingegnere.
* Il sistema deve essere in grado di gestire fino a 100 ingegneri contemporaneamente chef anno richiesta al server per la visualizzazione delle percentuali di IR.
* Il sistema deve essere in grado di gestire contemporaneamente N segnali, dove N è il numero di robot che il sistema gestisce. Per il corretto funzionamento bisogna gestire situazione di concorrenza tra i vari segnali, in modo che non ne vada perso qualcuno in uno stesso istante di tempo.
* Il sistema deve fornire un'affidabilità minima di up time del 98%.
* Il sistema deve essere scalabile.
* Il Sistema deve prevedere una GUI dalla quale sarà possibile monitorare le percentuali di IR.
* La GUI deve presentare una dashboard accessibile tramite web, essa deve

permette all’ingegnere di accedere alla sezione dei clusters e a quella dei robots.

* + La sezione dei cluster deve permettere all’ingegnere di vedere i valori di IR ogni cluster.
  + La sezione dei robot deve permettere all’ingegnere di vedere i valori di IR di ogni robot.
* La dashboard dovrà aggiornarsi automaticamente almeno ogni 5 minuti.  
  Requisito da specifica. Vincolo a cui il sistema deve sottostare per una corretta montitorizzazione.
* Tempi di aggiornamento sotto al minuto sono considerati preferenziali.
* Deve essere possibile navigare attraverso le sezioni andando avanti e indietro.

# B. Analysis Model

*<In this section, you shall report the Analysis Model produced for your system>*



[*Figura 2*](https://drive.google.com/open?id=1GQM-3nvqjN_o8_WL3IrU-iDG6K6BYi7E)

La dashboard si presenterà all’ingegnere mostrando tutte le aree presenti, selezionando un’area l’ingegnere sarà quindi in grado di visualizzare l’IR di tutti i clusters contenuti nell’area precedentemente selezionata, da qui, selezionando un clusters l’ingegnere sarà infine in grado di visualizzare tutti i robots contenuti nel clusters selezionato con i rispettivi IR.

Questa scelta è stata presa per mantenere la dashboard ordinata e di facile utilizzo evitando quindi di mostare l’IR di tutti i robots o tutti i clusters in un’unica pagina che sarebbe risultata confusionaria e di difficile navigazione.

## Boundary Objects:

**Interfaccia area:**

Dall’interfaccia iniziale l'utente può scegliere quale area selezionare.

**Visualizza IR Clusters:**

Tramite la sezione apposita l’utente può vedere l’IR dei clusters e a quale area appartengono.

**Visualizza IR Robots:**

Tramite la sezione apposita l’utente può vedere l’IR dei robots e a quale cluster appartengono.

## Entity Objects:

**IR\_Cluster:**

Valori di inefficiency rate dei cluster.

**IR\_Robot:**

Valori di inefficiency rate dei robot.

## Controller Objects:

**Invio IR Cluster:**

Attività del Sistema che si occupa di inviare I valori di IR dei cluster all’interfaccia grafica.

**Invio IR Robot:**

Attività del Sistema che si occupa di inviare I valori di IR dei robot all’interfaccia grafica.

**Calcolo IR Cluster:**

Attività del Sistema che si occupa di calcolare gli IR dei cluster.

**Calcolo IR Robot:**

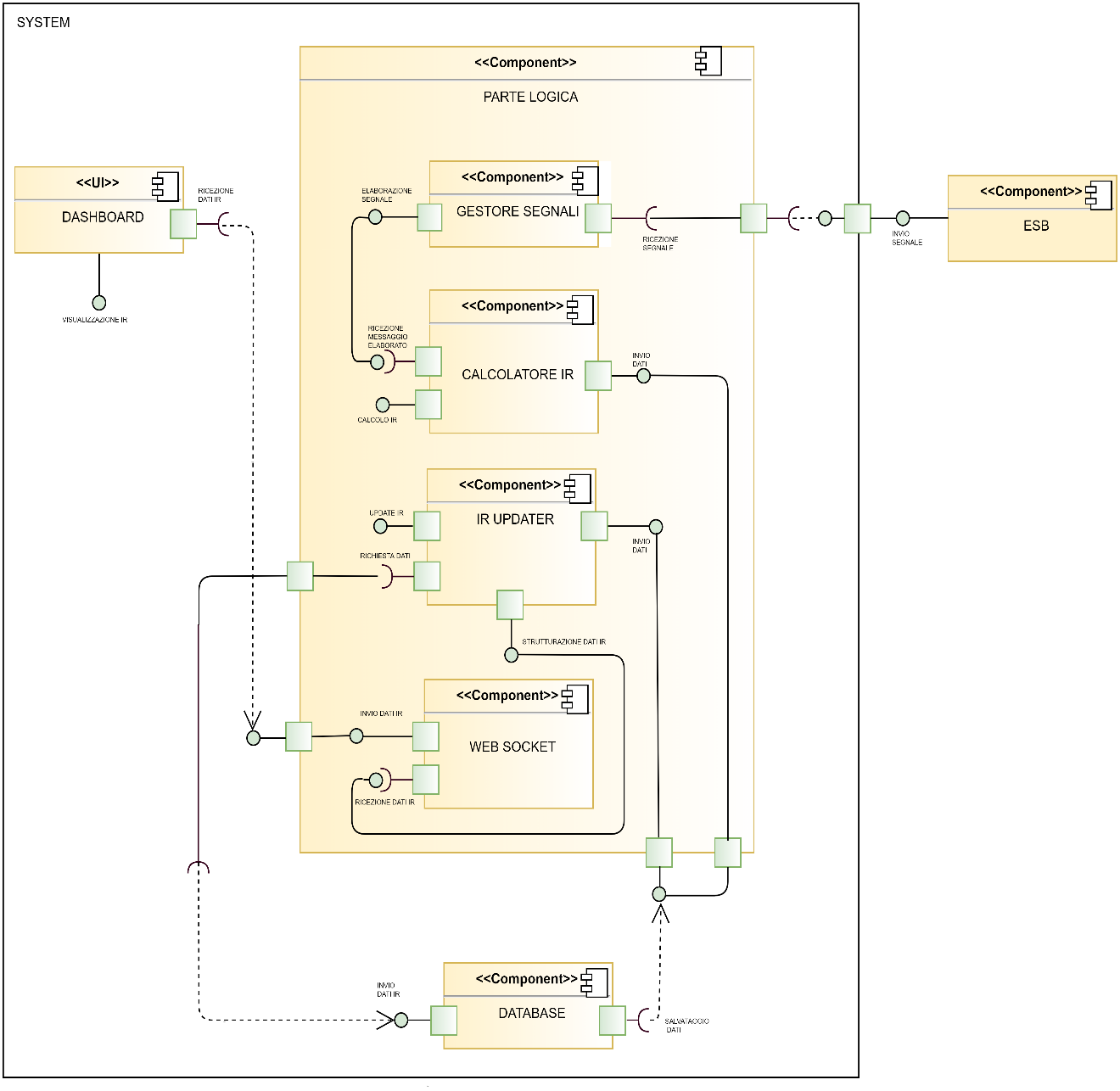
Attività del Sistema che si occupa di calcolare gli IR dei robot.

**Gestione messaggi:**

Attività del Sistema che si occupa di processare i messaggi ricevuti dai robot**.**

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

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



[*Figura 3*](https://drive.google.com/open?id=160DSHYKRuIHaTGUi_lJ-z6sjL3zZlfeJ)

Il component diagram (figura 3) è suddiviso in tre macro-componenti: **Parte web, parte logica, database**.   
Per una maggiore integrazione del sistema con l’ambiente è stata aggiunta una componente esterna al esso, l’ESB, dalla quale arrivano i messaggi da processare.

**Parte Logica:**Costituisce il core del sistema in quanto si occupa della gestione dei messaggi, del calcolo dell’IR di robots e clusters, del salvataggio dello stesso e dell’invio di questi dati ai clients che ne fanno richiesta, per questo motivo è la componente del sistema sulla quale ci siamo concentrati maggiormente e che ha richiesto un più alto grado di attenzione.   
Nel dettaglio i componenti:  
  
 **Gestore segnali:**

I segnali che arrivano dalla componente ESB vengono processati da   
questacomponente, essi vengono elaborati per poi procedere al calcolo dell’IR.  
 **Calcolatore IR:** Una volta ricevuto il messaggio elaborato dalla componente che si occupa di processarli essi vengono utilizzati per calcolare l’IR del robot che ha inviato il messaggio e eventualmente del suo cluster di appartenenza.  
 Dopo il calcolo dell’IR i dati vengono passati alla componente Database   
 per procedere allo storage.

**IR Updater:** Questacomponente si occupa di aggiornare periodicamente l’IR *se la finestra temporale di down di un robot/cluster non è ancora stata chiusa.*  
 Le percentuali di IR aggiornate vengono passate alla componente Database per procedere allo storage.  
 I dati vengono poi strutturatinel formato JSON per essere inviati alla componente Web Socket.

**Web Socket:**

Questa componente si occupa di gestire la connessione con i clients e di inviare i dati relativi all’IR già stati strutturati nel formato JSON dalla componente IR Updater.

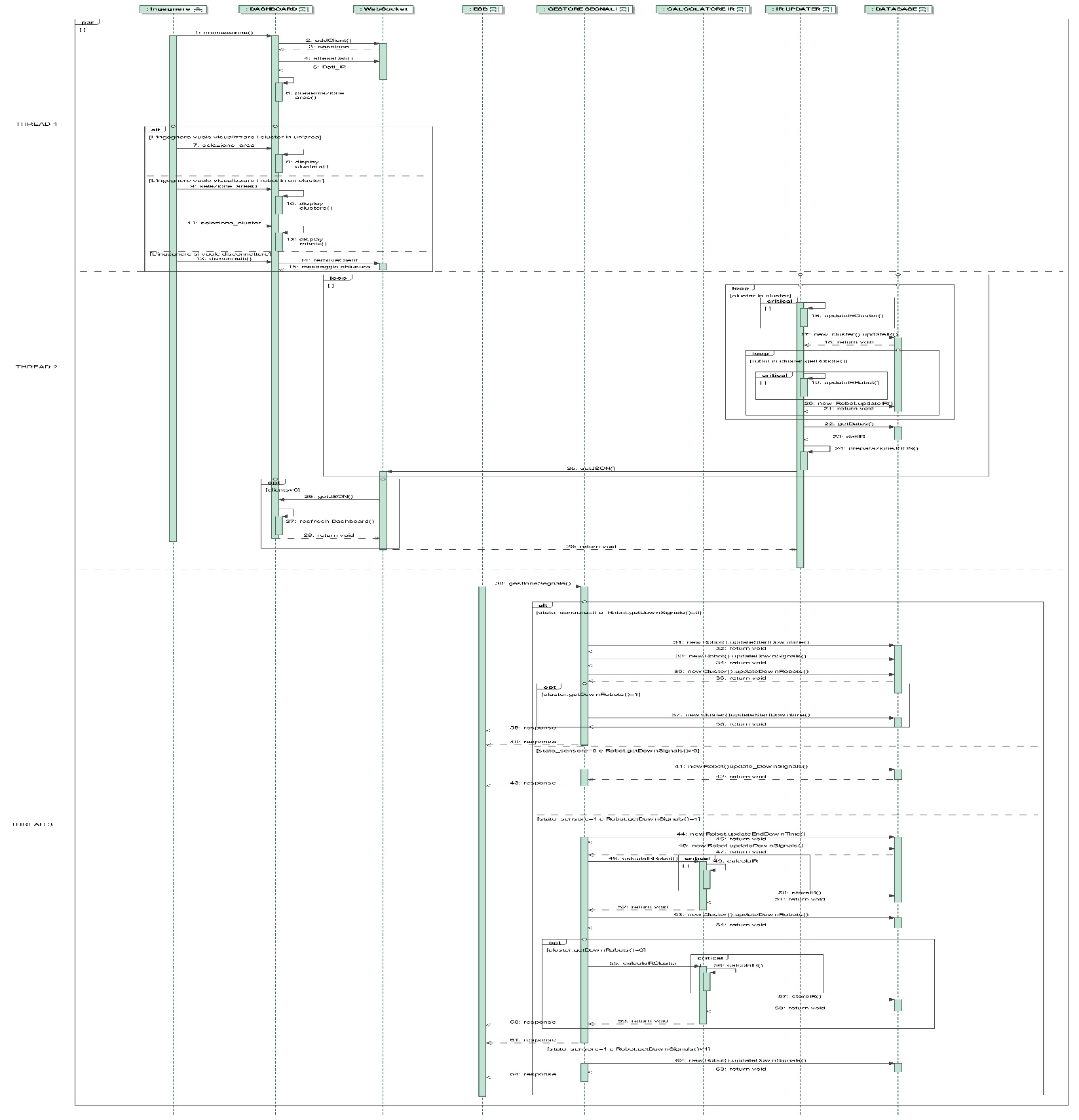
**Database:**

Componente che si occupa di salvare sul server tutti i dati che il sistema gestisce, questo ci permette di avere una copia di backup nel caso il gestore dei messaggi nella parte logica dovesse avere un arresto improvviso.   
Grazie a questa componente il livello di affidabilità del sistema è maggiore perché se la parte logica dovesse avere un periodo di inattività saremmo comunque in grado di recuperare i dati.

**Parte Web( rappresentata dalla componenete Dashboard):**

Sarà la parte del sistema accessibile dagli ingegneri, essi si potranno collegare al server e ricevere tramite web socket, gestito dalla parte logica, i dati relativi all’IR dei robots e dei clusters e potranno navigare all’interno della dashboard per monitorare l’andamento dei livelli di inefficiency rate.

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



[*Figura 4*](https://drive.google.com/open?id=1DspYajPv6tU2T6OHa_qxZEBUm5AQ4a0s)

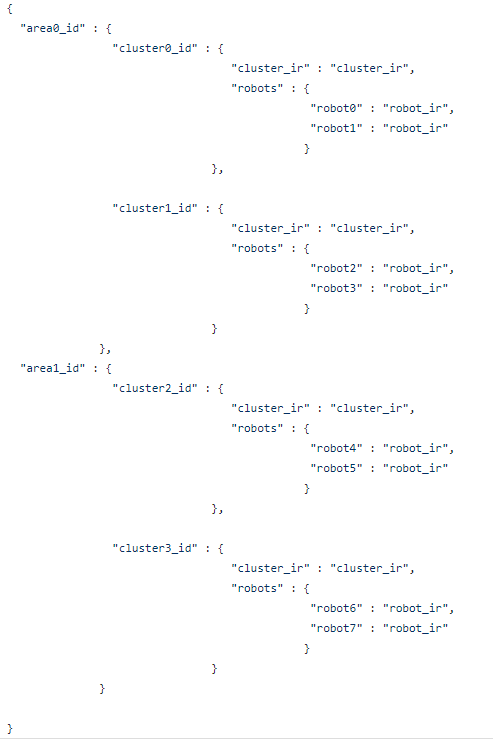
*Ctrl + click su “Figura 4” per una visualizzazione migliore.*

Il sequence diagram(Figura 4) è diviso nei tre threads che compongono il sistema:

**THREAD 1:**Questo thread è quello che si occupa di gestire l’interazione con i clients. L’ingegnere si collega alla dashboard, ciò porta alla creazione del canale di comunicazione che viene gestito dalla componente web socket, una volta che questa componente riceve il JSON da inviare ai clients dal THREAD 2 si occupa di inviarlo a tutti i clients connessi, il codice JavaScript che gestisce la dashboard si occuperà di mostrare i dati all’ingegnere.  
Tramite l’interazione con la dashboard l’ingegnere sarà in grado di navigare al suo interno.

**THREAD 2:**

Questo thread si occupa di aggiornare periodicamente le percentuali di IR di tutti i robots/clusters le cui fineste di down sono ancora aperte.  
Una volta aggiornati tutti i records nel database questo stesso thread si occupa di prelevare dal database tutte le percentuali di IR per poi strutturarle nel formato JSON nel seguente modo:



Il JSON verrà poi inviato alla componente Web Socket.

**THREAD 3:**

Questo thread è incaricato di gestire i messaggi che arrivano dalla componente ESB, ad ogni messaggio relativo ad un robot possono verificarsi 4 differenti scenari:

1. **Il messaggio è un messaggio di DOWN e il robot a cui esso fa riferimento ha il contatore dei segnali down a 0:**

In questo caso si procede ad impostare l’orario di inizio del downtime per quel robot, ad aggiornare il numero di segnali down di quel robot e ad aumentare il contatore di robot down del cluster di appartenenza del robot.

* 1. **Il numero di robot down del cluster passa da 0 a 1**

Si imposta l’orario di inizio di downtime del cluster.

1. **Il messaggio è un messaggio di DOWN e il robot a cui esso fa riferimento ha il contatore dei segnali down maggiore di 0:**Si procede ad aggiornare il numero di segnali down del robot.
2. **Il messaggio è un messaggio di UP e il robot a cui esso fa riferimento ha il contatore dei segnali down a 1:**In questo caso si procede a salvare l’orario dello start downtime del robot con la durata dello stesso(arrivo del messaggio corrente meno inizio downtime).  
   Si decrementa inoltre il contatore di robot down del cluster di appartenenza del robot.

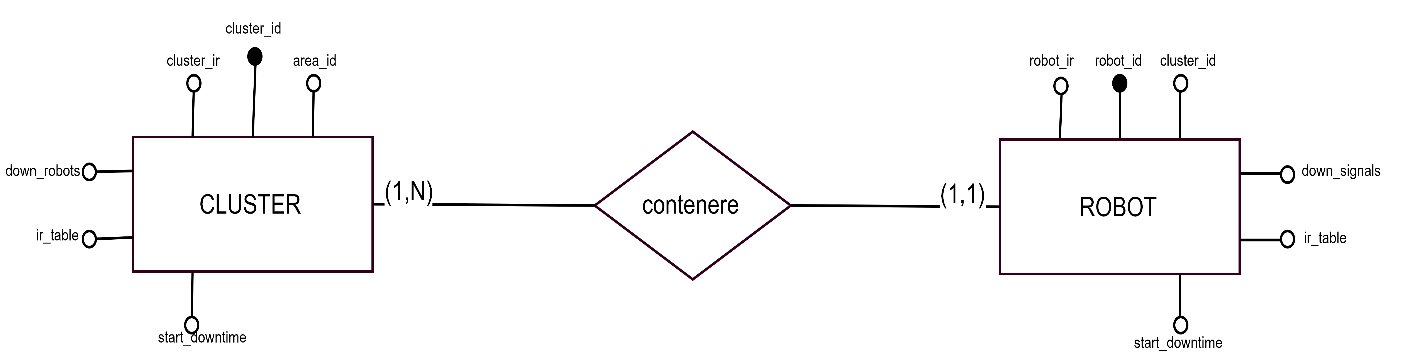
3.1 **Il numero di robot down del cluster passa da 1 a 0** In questo caso si procede a salvare l’orario dello start downtime del cluster con la durata dello stesso(arrivo del messaggio corrente meno inizio downtime).

1. **Il messaggio è un messaggio di UP e il robot a cui esso fa riferimento ha il contatore dei segnali down maggiore di 1**

Si procede ad aggiornare il numero di segnali down del robot.

# D. ER Design

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



[*Figura 5*](https://drive.google.com/open?id=12nz0ipznGA7U_QE8tKZrRlXd06K_kmfn)

Algebra relazionale:

* Robot (**\_id**, cluster\_id, robot\_ir, down\_signals, start\_downtime, ir\_table)
* Cluster (**\_id**, area\_id, cluster\_ir, down\_robots, start\_downtime ir\_table)
* Contiene (**Cluster**, **Robot**)

Vincolo di integrità referenziale tra cluster\_id di Cluster e robot\_id di Robot.

Le principali entità nel rappresentate nel diagramma ER(figura 5) sono cluster e robot.   
Il robot ha i seguenti attributi:

1. Il suo codice identificativo,
2. Il codice identificativo del cluster di appartenenza,
3. La sua attuale percentuale di IR relativa all’ultima ora,
4. Il numero attuale di segnali down,
5. I suoi intervalli di downtime nell’ultima ora,
6. La data e l’ora dell’inizio dell’ultimo down time.

Il cluster ha i seguenti attributi:

1. Il suo codice identificativo,
2. Il codice identificativo dell’area di sua appartenenza,
3. La sua attuale percentuale di IR relativa all’ultima ora,
4. Il numero attuale di robots down,
5. I suoi intervalli di downtime nell’ultima ora,
6. La data e l’ora dell’inizio dell’ultimo down time.

# E. Class Diagram of the implemented

# System

[*Figura 6*](https://drive.google.com/open?id=1CW_BWhKgcyeq4icyCVqmSycdAxI7Q-cx)

*Ctrl + click su “Figura 6” per una visualizzazione migliore.*

Il class diagram(figura 6) è diviso in 5 principali packages:

1. MODEL   
   All’interno di questo package sono contenute le classi che rapprensentano le principali entità del sistema: aree, clusters e robots.   
   È all’interno di queste stesse classi che avviene il calcolo dell’IR e l’update dei robots e dei clusters in base al tipo di segnale ricevuto.
2. DATABASE  
   Qui è presente l’enumerazione Java che si occupa di stabilire una connessione con il database, essa è implementata sfruttando il ***connection pooling*** in modo da evitare che per ogni accesso al DB debba essere stabilita una nuova connessione che andrebbe a pesare molto sui tempi del sistema ad ogni nuova richiesta di scrittura o lettura.   
   All’interno di questo package troviamo inoltre altri due sub-packages necessari alla realizzazione del design pattern ***Data-access-object*** (DAO).

2.1 INTERFACESData access object interface. Questa package contiene le interfacce   
 che definiscono le operazioni che possono essere eseguite su un   
 model object(area, cluster, robot).2.2CONCRETEData access object concrete class. Queste classi implementano le   
 interface presenti nel package sovrastante, esse si occupano  
 dell’effettiva interazione con il database sfruttando il connettore  
 presente all’interno del package database.

1. ***SERVER***

All’interno di questo package sono contenute le classi che si occupano della realizzazione di due servers: il primo per la ricezione e la gestione dei messaggi, il secondo per stabilire una connessione tramite web socket con i clients e inviare tramite questo canale i dati necessari che verranno poi elaborati sui clients stassi tramite JavaScript per mostrare le percentuali di IR.

1. ***SERVICES***

Questo package contiene due classi. La prima(IRUpdater) si occupa di aggiornare periodicamente il database, una volta che i records sono aggiornati la seconda classe(ClientsSender) si occupa di generarare un JSON contenente le informazioni presenti nel database che verrà inviato a intervalli regolari ai clients connessi tramite web socket(Broadcast) al server presente nel package sovrastante.  
I clients che faranno richiesta al server sulla porta 80 riceveranno un file HTML e un file JavaScript che eleborando il JSON ricevuto sarà in grado di mostare le percentuali di IR dei robots e dei clusters.

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

**Broadcasting, Web socket:**

La scelta di sfruttare un canale di comunicazione che rimanga sempre aperto tra server e clients ci ha permesso di gestire con relativa semplicità un gran numero di richieste da parte degli ingegneri.  
Dai test effettuati le dashboard di più di 10 clients connessi si aggiornavano ***quasi in contemporanea e con coerenza dei dati***, questo con il server connesso ad una rete con velocità di upload di 0.3 Mbps!(Ci aspettiamo che la connessione che sfrutterà il server della Micron abbia una velocità nettamente superiore).

***PRO*:** Gestione dei clients semplificata, elaborazione dati lato server.  
 ***CONTRO*:** Nel caso di un arresto improvviso del server che gestisce il web socket tutti i clients connessi perderebbero la connessione esso e la

Dashboard smetterebbe di aggiornarsi.

**Calcolo dell’IR:**

Salviamo la data e l’ora di quando un robot va down(T1), identificata dalla voce *start\_downtime*; quando il robot torna up(T2) sottraendo T2 a T1 si ottiene il tempo di down(DT1) da T1. La somma DTtot dei tempi DTi, dell’ultima ora, cosi calcolati ci permette di ricavare la percentuale di IR.  
È stato adottato lo stesso principio per i clusters, in particolare l’inizio del down time(T1) di un cluster coincide con il passaggio da 0 a 1 del contatore dei robots down di quel determinato cluster mentre la fine del down time (T2) coincide con il passaggio da 1 a 0 del contatore dello stesso.

***NOTA:***

Questo metodo non ci permette di tenere in considerazione il tempo di down se un robot/cluster ha iniziato il suo down time ma non lo ha ancora concluso, ovvero se c’è almeno un segnale/robot down al momento del calcolo dell’IR, per questo motivo la componente che si occupa di aggiornare i valori di IR aggiunge a DTtot la differenza tra il momento attuale(quello dell’aggiornamento) e quello dell’inizio dell’ultimo downtime a tutti i robots/clusters la cui finestra temporale di down non è ancora stata chiusa. Questo ci permette di avere il calcolo dell’IR riferito a tutta l’ultima ora.

***PRO*:** Semplificazione per il calcolo dell’IR.  
 ***CONTRO*:**  -

**Scelta Database:**

Per lo storage dei dati abbiamo deciso di utilizzare MongoDB che ci permette di strutture i dati in modo semplice e più veloce rispetto a quello che ci offre un DBMS come MySQL.

Siamo giunti a questa conclusione dopo aver effettuato vari test, qui sotto i tempi medi risultanti:

***Inserimento di robots generati casualmente:***

MySQL --> 56846 robots in 2,13238333 minuti.

MongoDB --> 56745 robots in 0,35666667 minuti.

***Aggiornamento dei records nel database:***

MySQL --> 77725 robots and 100 clusters in 3,41666667 minuti.

MongoDB --> 77568 robots and 100 clusters in 0,42798333 minuti.

***Tempo per acquisire dal database tutte le informazioni necessarie per il sistema:***

MySQL --> 90000 robots e 100 clusters in 0,03288333 minuti.

MongoDB --> 90000 robots e 100 clusters in 0,02388333 minuti.

Inoltre il connettore Java per MongoDB, mongo-java-driver, implementa di per se il connection pooling, il che ci ha permesso di risparmiare tempo nella scrittura del codice.

***PRO*:** Maggiore velocità nella gestione dei dati.  
 ***CONTRO*:**  Tempo speso nell’apprendere la tecnologia NoSQL.

**Scelta per la gestione dei robots e dei clusters down:**

Abbiamo implementato un contatore per i robots che tenga traccia del numero dei segnali down(un robot è down se ha almeno un segnale down) e un contatore per i clusters che tenga traccia del numero dei robots down al suo interno(un cluster è down se ha almeno un robot down). Questo ci ha permesso di ridurre la complessità del problema, infatti, non è necessario sapere ***quali*** sono i segnali(o robots nel caso dei clusters) down in un determinato lasso di tempo ma soltanto ***quanti*** ne sono.

***PRO*:** Semplificazione per tenere conto dei robots e dei clusters down.  
 ***CONTRO*:**  -

**Scelta elaborazione dati lato server:**

Abbiamo deciso che l’elaborazione dei dati per il calcolo dell’IR dei clusters e dei robots avverà dal lato server per poi essere inviati (tramite Web Socket) ai clients che ne faranno richiesta. In questo modo si evita di inviare a ciascun client tutti i dati(grezzi) relativi ai clusters e ai robots e lasciare che sia lo stesso server ad elaborarli.

***PRO*:** I dati non devono essere elaborati dai clients, essi saranno coerenti su tutte le dashboard.  
 ***CONTRO*:**  -

# G. Explain how the FRs and the NFRs are satisfied by design

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

FUNCTIONAL:

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

* ***Il Sistema deve prevedere una GUI dalla quale poter visualizzare le percentuali di IR di robots e clusters.***

La GUI è stata realizzata tramite HTML5, CSS e JavaScript.

La GUI è stata realazziata in modo da poter essere accesibile tramite web, questo grazie all’implementazione di un Server realizzato tramite Java in grado di essere raggiunto tramite la rete locale o da remoto. In questo modo gli ingegneri possono visualizzare la dashboard in qualsiasi momento conoscendo l’indirizzo IP del server.

* ***La GUI deve presentare una dashboard accessibile tramite web, essa deve***

***permette all’ingegnere di accedere alla sezione dei clusters e a quella dei robots per monitorarne le percentuali di IR.***

* + ***La sezione dei cluster deve permettere all’ingegnere di visualizzare i valori di IR di ogni cluster.***
  + ***La sezione dei robot deve permettere all’ingegnere di visualizzare i valori di IR di ogni robot.***

I valori di IR dei robots e dei clusters sono elabarati dal server e sono inviati ai clients tramite web socket in formato JSON, il JavaScript lato client si occupa di elaborare il JSON ricevuto e di mostrare i livelli di IR in base alla selezione dell’utente(ingegnere).

* ***Deve essere possibile navigare attraverso le sezioni muovendosi avanti e indietro nella dashboard.***

Il codice JavaScript ci permette di navigare all’interno della dashboard.

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

* ***Il sistema deve prevedere, nella sua una parte logica, una componente che si occupi di processare i messaggi inviati dai robots e calcolare gli IR dei robots e dei clusters salvandoli nel database.***

Abbiamo realizzato un server che è in ascolto dei messaggi HTTP in arrivo dai clients sulla porta 8000. Questi messaggi vengono elaborati e poi inseriti opportunamente nel database.

* ***Il sistema nella sua parte logica deve avere una componente che si occupi di aggiornare i records nel database, eseguire quries su di esso per poi strutturare i dati che verranno inviati ai clients.***

È stata realizzata una classe Java(IRUpdater, package services) che ogni 30 secondi si occupa di aggiornare tutti i records nel database.  
Questa classe si occupa inoltre di eseguire queries sul database e generare un file JSON in accordo con i dati prelevati.

* ***Il sistema nella sua parte logica deve avere una componente che si occupi di inviare ai clients che ne fanno richiesta(ingegneri) i dati relativi alle percentuali di IR.***

È stata realizzata una classe Java(ClientsSender, package services) che riceve un oggetto di tipo JSON dalla classe IRUpdater e lo invia tramite web socket ai clients connessi al server(**BROADCAST**).

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

* ***Il sistema deve prevedere lo storage degli IR dei robots e dei clusters su apposito database per garantire una maggiore affidabilità. Il database deve essere collegato alla parte logica del sistema per poter essere interrogato e fornire le informazioni utlili dei robots e dei clusters.***

Tramite il design pattern DAO siamo stati in grado di realizzare nella parte logica del sistama una componente in grado di interagire con il database per lo storage dei dati.  
La parte logica e il database sono messi in comunicazione dal driver di MongoDB per Java.

NON FUNCTIONAL:

B1.1 GUI NFRequirements (da riempire a partire dalla Versione 2)

* ***La dashboard dovrà aggiornarsi automaticamente almeno ogni 5 minuti.  
  Requisito da specifica.***
  + ***Un tempo di refrash sotto al minuto è considerato preferenziale.***

**Siamo riusciti a tenere il refresh della dashboard sotto il minuto**(circa 20 secondi) grazia al database impiegato che ci permette di avere un accesso rapido ai dati e soprattuto grazie alla divisione dei compiti dei vari componenti della parte logica del sistema.  
La componente che si occupa di aggiornare i records nel database infatti impiega circa 15 secondi per terminare la sua attività, dopodichè la componente che si occupa di prelevare le informazioni dal database e generare il JSON da inviare ai clients prende la restante parte del tempo.

## B1.2 Business Logic NFRequirements

* ***Il sistema deve fornire un'affidabilità minima di up time del 98%.  
  Il sistema deve essere di grande affidabilità in modo da non compromettere il funzionamento di altri sistemi collegati ad esso.***

L’affidabilità del sistema è garantita grazie all’utilizzo del database che ci permette di recuperare tutti i dati nel caso di un arresto improvviso di qualche componente del sistema. Se il sistema dovesse arrestarsi inaspettatamente potrebbe essere riavviato senza perdere alcuna informazione, infatti i messaggi che non sono stati elaborati nel periodo di inattività verrebbero comunque gestiti senza perdere alcuna informazione.

Questo è possibile grazie al utilizzo del protocollo tcp per la trasmissione dei messaggi. Essendo questo un protocollo con ricevuta di ricezione, il gestore del BUS, su nostra richiesta può continuare a tenere in un Buffer i messaggi che riceve dai robot, ma che non invia a noi, poiché il primo ad uscire non viene ricevuto(logicamente come in una queue). Al riavvio del “Receiver” i messaggi arretrati vengono processati, senza perdita di informazioni, poiché il receiver và a ricostruire il punto di shut down, a partire dal database e, i messaggi stessi contengono data e ora dell’invio da parte del robot. L’affidabilità del DB è garantita dalla possibilità di adottare soluzioni RAID per la sua implementazione(va tenuto in considerazione che con 90.000 robot, il peso del DB è di circa 2 megabytes).

* ***Il sistema deve essere in grado di gestire contemporaneamente N segnali, dove N è il numero di robot che il sistema gestisce. Per il corretto funzionamento bisogna gestire situazione di concorrenza tra i vari segnali, in modo che non ne vada perso qualcuno in uno stesso istante di tempo.***

I segnali vengono gestiti grazie ad una coda con il criterio FIFO(First-in-first-out) 90000 messaggi contemporanei sono gestiti in un tempo medio di 30 secondi. Simulando una situazione di stress(cioè che ogni robot invia un segnale ogni secondo);i l limite, su una rete locale, che ci permettesse di rimanere al passo con il numero di messaggi ricevuti, elaborandoli senza rimanere indietro, è di circa 7 milioni. Questo numero è comunque da considerarsi una base di partenza visto la esigua potenza delle macchine che abbiamo avuto a disposizione per eseguire i test.

* ***Il sistema deve essere in grado di gestire fino a 100 ingegneri contemporaneamente che fanno richiesta al server per visualizzare le percentuali di IR.***Siamo riusciti gestire un gran numero di richieste da parte dei clients che fanno richiesta al server grazie al web socket che mette in comunicazione le due parti. Avendo il sistema e l’ingegnere sulla stessa rete locale, il limite di connessioni contemporanee è la potenza stessa della macchina implementante il sistema stesso ad essere il collo di bottiglia. In caso di sistema e ingegneri dislocati su reti differenti, il collo di bottiglia è dato dal numero di connessioni contemporanee gestibili dalla rete dove risiede la macchina.
* ***Il sistema deve essere scalabile.Bisogna prevedere ampliamenti futuri come l’aumento del numero di robot.***

La scalabilità del sistema è garantita dalla struttura del prototipo fornito in quanto ogni messaggio ricevuto viene analizzato e scomposto nei vari campi utili al fine di smistare il robot nel corretto cluster e il cluster nella propria area di appartenenza.   
Con questo procedimento rendiamo il sistema “adattivo” in accordo ai messaggi in arrivo, quindi non abbiamo una struttura predefinita delle area, dei cluster al loro interno e dei robots all’interno dei clusters ma le nostre strutture si adattano in base al messaggio ricevuto.

Come menzionato prima, i nostri mezzi non ci hanno permesso di testare il limite delle tecnologie attuali come CPU server, connessioni a banda ultralarga e infrastrutture ad-hoc, per trovare appunto il numero limite di robot e ingegneri gestibili. Con un ampliamento del budget a nostra disposizione e conteggiando 15 giorni extra per eseguire i test, è possibile fare test più accurati e approfonditi.

# H. Effort Recording

GANTT*Make a GANTT 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.*

*Figura 7*

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

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

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

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

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

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

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

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

Appendix. Code   
*<Report in this section a* ***documented*** *version of the produced code>  
<Discuss how your code satisfies the FR and NFR>  
<Show some screenshots of the code behavior>*

*<please upload your executable code in the dropbox folder>*

Il Prototipo presentato rispetta tutti i vincoli sopraelencati, oltre al sistema per la monitorizzazione della percentuale di IR sono state fornite anche delle classi che permettono di simulare l’invio dei messaggi da parte dei robots. Di seguito il link alla repository di GitHub dove è possibile trovare il codice, nel README.md è illustrata la procedura per impostare il sistema. **Il prototipo finale non è sul branch MASTER ma sul branch DB.**

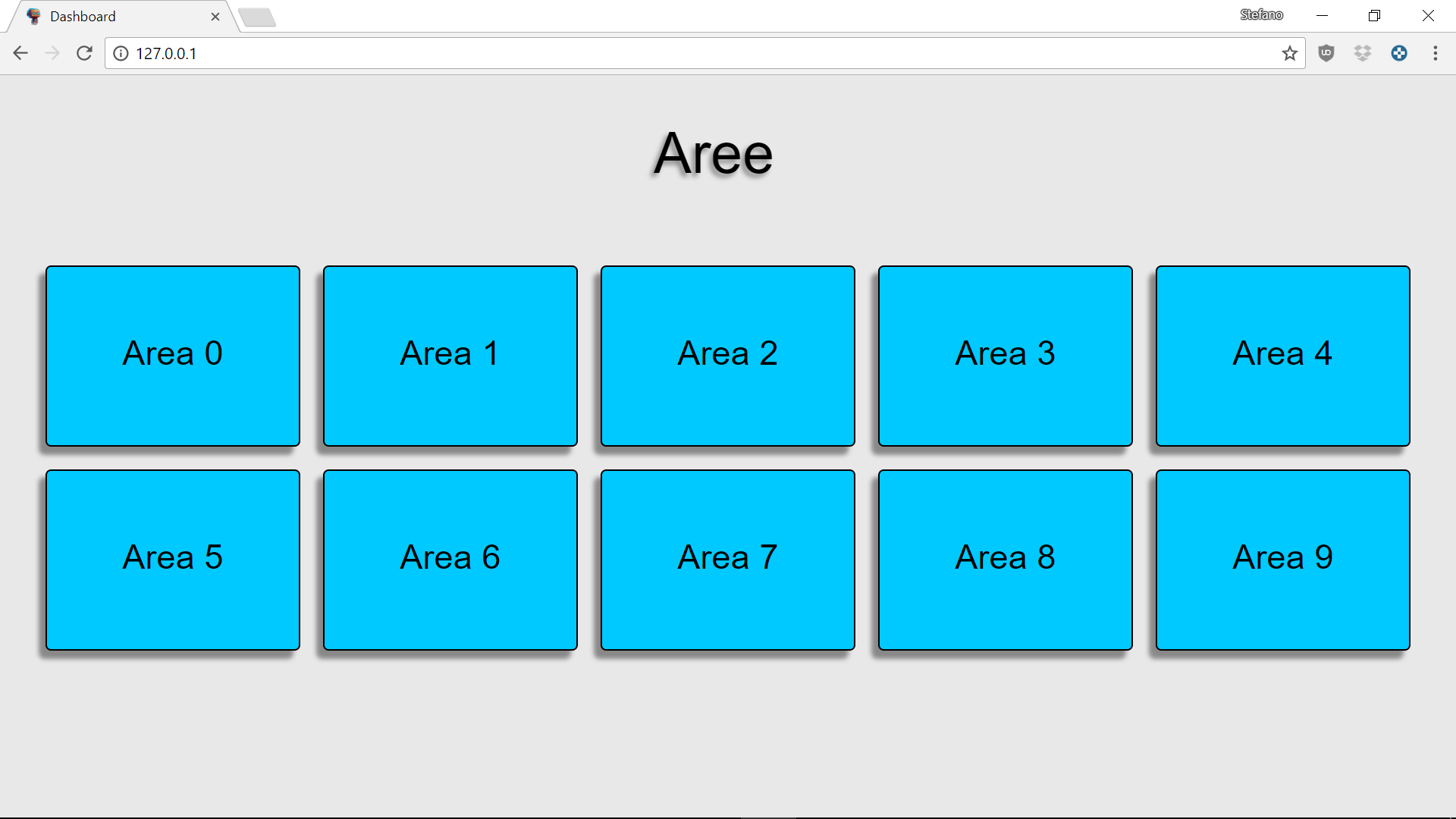
***Link alla repository di GitHub:*** [*https://github.com/paolozu/seAIMgroup/tree/db*](https://github.com/paolozu/seAIMgroup/tree/db)

**Di seguito qualche screenshots della dashboard:**

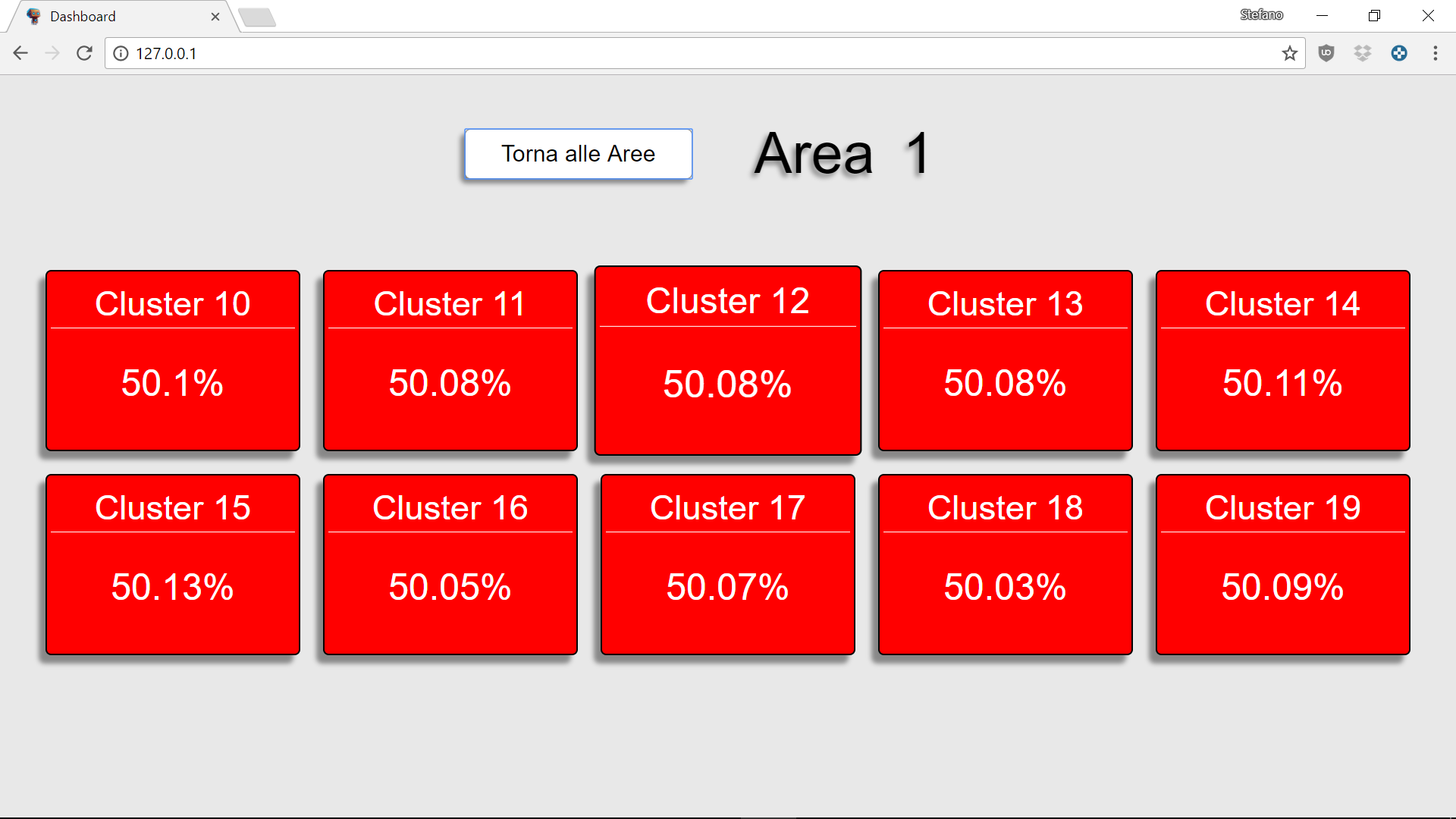
Quella di seguito è la situazione che si presenterà all’ingegnere quando si connetterà per la prima volta alla dashboard.



Una volta che il server avrà inviato i dati al client che ne ha fatto richiesta l’ingegnere si troverà a poter selezionare l’area di suo interesse.



Una volta selezionata l’area di interesse sulla dashboard verranno mostrate tutte i clusters presenti all’interno di quell’area, nello screenshot si può notare che esse sono tutte di colore rosso perchè il loro IR è superiore al 40%.

**

Dopo aver selezionato il cluster di proprio interesse l’ingegnere sarà in grado di visualizzare la percentuale di IR dei robots all’interno di quell cluster.  
Nello screenshot è possibile verificare che i robots che hanno una percentuale di IR maggiore del 40% hanno un colore di background rosse mentre gli altri in verde.

