

# PROGETTO PROGRAMMAZIONE 3 SISTEMA DOMOTICO

EDUARDO AUTORE

MATRICOLA:0124001586



EDUARDO AUTORE

UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

# Sommario

Traccia .....	2
Requisiti .....	3
UML .....	4
Pattern Utilizzati .....	5
Gestione Eccezioni .....	14
Esecuzione .....	16
MODALITA' COLLAUDO .....	18
INSTALLAZIONE SENSORI .....	19
AGGIUNGI COMPONENTI.....	21
RESETTA SENSORI .....	22
GESTIONE OGGETTI CASA .....	23
GESTIONE VALORI MISURAZIONE.....	24
MOSTRA STATISTICHE.....	25
MODALITA' INTERVENTO.....	26

## Traccia

Si vuole sviluppare un sistema domotico per il monitoraggio di una casa.

Diversi sensori sono installati per il monitoraggio (e.g., temperatura, corrente elettrica, ecc.) o per l'intervento (e.g., getto d'acqua, getto d'aria, ecc.).

Ogni sensore è nello stato acceso o spento e quelli di monitoraggio registrano i valori di riferimento.

L'accesso al sistema può avvenire in modalità collaudo o in modalità attivato.

In modalità collaudo si possono effettuare le seguenti operazioni:

- installare un nuovo sensore.
- aggiungere componenti ad un sensore (e.g., ad una telecamera si aggiunge un modulo audio).
- resettare tutti i sensori.
- periodicamente è possibile mostrare le statistiche dei sensori (e.g., data e ora dell'allarme).

In modalità attivato si possono effettuare le seguenti operazioni

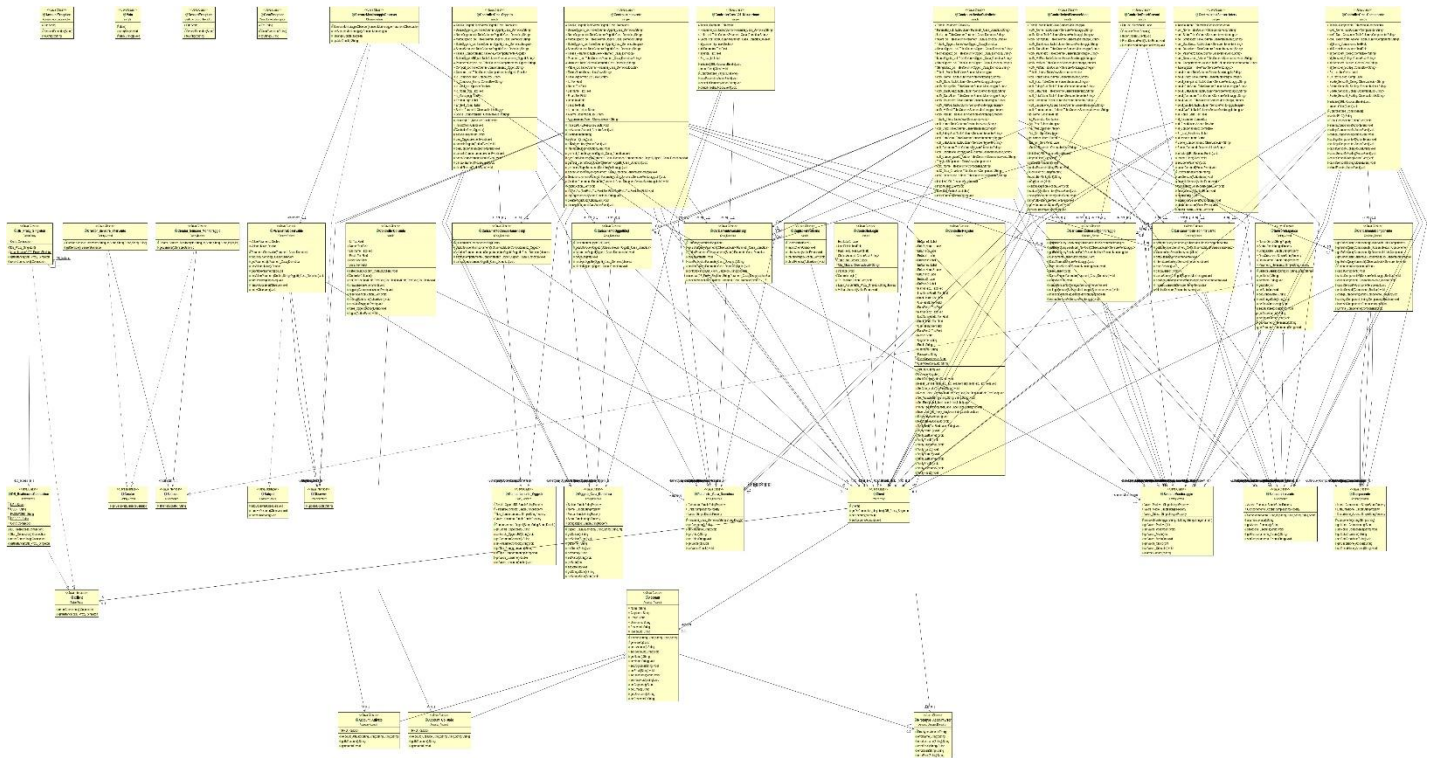
- se un sensore per il monitoraggio lancia un allarme viene attivato il corrispondente sensore per l'intervento. In caso di più allarmi vengono gestiti con un politica FIFO.
- se un sensore per il monitoraggio cessa l'allarme viene disattivato il corrispondente sensore per l'intervento. In caso di più allarmi vengono gestiti con un politica FIFO.

Scrivere un programma che permetta di simulare il sistema di monitoraggio.

## Requisiti

1. Applicazione Web o programma standalone con supporto grafico.
2. Attenersi ai principi della programmazione SOLID.
3. Usare il linguaggio Java.
4. Inserire sufficienti commenti (anche per Javadoc) e annotazioni.
5. Gestione delle eccezioni.
6. Usare file o database.

# UML



## Pattern Utilizzati

**Singleton Pattern:** Pattern utilizzato per la gestione della connessione con il Database. Infatti questo pattern ha come scopo quello di assicurarsi che una classe abbia una sola istanza e di fornire un punto d'accesso globale ad essa.

In questo caso la classe singleton è anche una classe del pattern proxy a cui viene delegato il compito di fornire un surrogato (o placeholder) per la classe contenente la reale connessione al database al fine di controllarne l'accesso.

```
5 public class DB_Proxy_Singleton_Connection implements IClient{
6     //UNICA ISTANZA DELLA CLASSE
7     private static DB_Proxy_Singleton_Connection DB_instance=null;
8     private Connection Conn;
9
10    protected DB_Proxy_Singleton_Connection(){
11        //METODO STATICO ACCESSIBILE GLOBALMENTE CHE RITORNA L'UNICA ISTANZA DELLA CLASSE
12        private static DB_Proxy_Singleton_Connection createInstance(){
13            if(DB_instance == null){
14                DB_instance=new DB_Proxy_Singleton_Connection();
15                DB_RealAccessConnection db_access = new DB_RealAccessConnection();
16
17                //ISTANZA SINGLETON A CUI VIENE DELEGATO IL COMPITO DI FAR ESEGUIRE
18                // LA CONNESSIONE AL DATABASE TRAMITE L'OGGETTO ORIGINALE DB_REAL_ACCESS
19                DB_instance.Conn= db_access.Start_Connection();
20                System.out.println("Connessione ottenuta");
21            }
22            return DB_instance;
23        }
24        @Override
25        public DB_Proxy_Singleton_Connection getInstance() { return createInstance(); }
26
27
28
29        //METODO EREDITATO DALL'INTERFACCIA PROXY CHE RITORNA LA CONNESSIONE AL DATABASE
30        @Override
31        public Connection returnConnection(){
32            if(DB_instance != null)
33                return this.Conn;
34            else{
35                System.out.println("Non è stata creata nessuna connessione\n");
36                return null;
37            }
38        }
39    }
```

**Proxy Pattern:** Pattern utilizzato con funzione di [Protection Proxy](#) , ovvero per controllare l'accesso all'oggetto originale contenente la reale connessione al database e di [Smart Proxy](#) ,ovvero per interporre azioni aggizionali quando si effettua l'accesso all'oggetto originale, in questo caso evitiamo che l'oggetto sia accessibile esternamente ed infine verifichiamo che venga creato una sola volta in modo che l'istanza singleton possa restituire sempre quella stessa connessione.

Interfaccia Proxy per la gestione degli accessi:

```
5 //INTERFACCIA DEL PATTERN PROXY PER GESTIRE L'ACCESSO ALL'OGGETTO ORIGINALE
6 public interface IClient{
7     Connection returnConnection();
8     DB_Proxy_Singleton_Connection getInstance();
9 }
```

Classe oggetto originale contenente la connessione reale al database:

```
9 public class DB_RealAccessConnection implements IClient {
10     //DATI FONDAMENTALI PER EFFETTUARE LA CONNESSIONE AL DATABASE
11     private final static String URL = "jdbc:mysql://localhost:3306/Db_Sensori?useTimezone=true&serverTimezone=UTC";
12     private final static String USER="Eddy";
13     private final static String PASSWORD="password";
14     private final static String DRIVER = "com.mysql.cj.jdbc.Driver";
15     private Connection Conn;
16
17     DB_RealAccessConnection(){}
18
19     //METODO CHE EFFETTUA LA CONNESSIONE AL DATABASE MA CHE PUO' ESSERE RESTITUITO ESTERNAMENTE SOLO DALLA CLASSE DELEGATA
20     protected Connection Start_Connection() {
21         try {
22             Class.forName(DRIVER);
23             this.Conn = DriverManager.getConnection(URL, USER, PASSWORD);
24         } catch (SQLException | ClassNotFoundException ex) {
25             Logger.getLogger(DB_RealAccessConnection.class.getName()).log(Level.SEVERE, msg: null, ex);
26         }
27         return this.Conn;
28     }
29     //METODI EREDITATI DALL'INTERFACCIA PROXY CHE IN QUESTO CASO NON POTRANNO RESTITUIRE NULLA
30     //POICHE' LA CLASSE NON DEVE ESSERE ACCESSIBILE DIRETTAMENTE
31     @Override
32     public Connection returnConnection(){
33         System.out.println("Accesso vietato a questa risorsa");
34         return null;
35     }
36     @Override
37     public DB_Proxy_Singleton_Connection getInstance() {
38         System.out.println("Accesso vietato a questa risorsa");
39         return null;
40     }
41 }
```

Classe Singleton Proxy a cui viene delegato il compito di restituire la connessione tramite l'oggetto originale:

```
5 public class DB_Proxy_Singleton_Connection implements IClient{
6     //UNICA ISTANZA DELLA CLASSE
7     private static DB_Proxy_Singleton_Connection DB_instance=null;
8     private Connection Conn;
9
10    protected DB_Proxy_Singleton_Connection(){
11        //METODO STATICO ACCESSIBILE GLOBALMENTE CHE RITORNA L'UNICA ISTANZA DELLA CLASSE
12    private static DB_Proxy_Singleton_Connection createIstance(){
13        if(DB_instance == null){
14            DB_instance=new DB_Proxy_Singleton_Connection();
15            DB_RealAccessConnection db_access = new DB_RealAccessConnection();
16
17            //ISTANZA SINGLETON A CUI VIENE DELEGATO IL COMPITO DI FAR ESEGUIRE
18            // LA CONNESSIONE AL DATABASE TRAMITE L'OGGETTO ORIGINALE DB_REAL_ACCESS
19            DB_instance.Conn= db_access.Start_Connection();
20            System.out.println("Connessione ottenuta");
21        }
22        return DB_instance;
23    }
24    @Override
25    public DB_Proxy_Singleton_Connection getInstance() { return createIstance(); }
26
27
28
29    //METODO EREDITATO DALL'INTERFACCIA PROXY CHE RITORNA LA CONNESSIONE AL DATABASE
30    @Override
31    public Connection returnConnection(){
32        if(DB_instance != null)
33            return this.Conn;
34        else{
35            System.out.println("Non è stata creata nessuna connessione\n");
36            return null;
37        }
38    }
39 }
```



**Factory Method:** Pattern utilizzato per creare le differenti tipologie di sensori. Questo pattern ha come fine, quello di definire un'interfaccia per creare un oggetto ma lasciando la scelta del tipo alla sua sottoclasse.

In questo caso le sottoclassi possibili sono [Sensore Monitoraggio](#) e [Sensore Intervento](#).

Interfaccia Creator:

```
3 //INTERFACCIA CREATOR DEL PATTERN FACTORY METHOD
4 public interface Creator{
5     SensoreGenerico getSensore();
6 }
7
```

Creator Sensore Monitoraggio:

```
3 //CLASSE "CONCRETE_CREATOR" DEL PATTERN FACTORY METHOD
4 // CHE I OCCUPA DI CREARE IL "PRODUCT" SENSORE MONITORAGGIO
5 public class Creator_Sensore_Monitoraggio implements Creator{
6
7     private Sensore Install;
8
9     public Creator_Sensore_Monitoraggio(String N,int Stato,String Data_C,String Data_A,String P_Mon,int V_Pos, int V_Med, int V_Crit){
10
11         Install=new SensoreMonitoraggio(N,Stato,Data_C,Data_A,P_Mon,V_Pos,V_Med,V_Crit);
12         getSensore();
13     }
14
15     //METODO FACTORY DEL PATTERN
16     @Override
17     public SensoreGenerico getSensore() { return (SensoreMonitoraggio) Install; }
18 }
19
```

Creator Sensore Intervento:

```
3 //CLASSE "CONCRETE_CREATOR" DEL PATTERN FACTORY METHOD
4 // CHE I OCCUPA DI CREARE IL "PRODUCT" SENSORE INTERVENTO
5 public class Creator_Sensore_Intervento implements Creator{
6
7     private Sensore Install;
8
9     public Creator_Sensore_Intervento(String N,int S,String Data_C,String Data_A,String P_rif,String DescrAzione,String ComportAzione){
10         Install=new SensoreIntervento(N,S,Data_C,Data_A,P_rif,DescrAzione,ComportAzione);
11         getSensore();
12     }
13
14     //METODO FACTORY DEL PATTERN
15     @Override
16     public SensoreGenerico getSensore() { return (SensoreIntervento) Install; }
17 }
18
```

Interfaccia Sensore per i Prodotti del Factory Method:

```
4 //INTERFACCIA PER I PRODOTTI DEL FACTORY METHOD
5 public interface Sensore {
6     String allarmeSuonato();
7 }
8
```

Classe astratta Sensore Generico che implementa l'interfaccia Sensore:

```
6 //CLASSE ASTRATTA CHE IMPLEMENTA L'INTERFACCIA "SENSORE" UTILIZZATA PER IL PATTERN FACTORY METHOD
7 //CHE RIASSUME LE CARATTERISTICHE COMUNI DI TUTTE LE SOTTOCLASSI
8 public abstract class SensoreGenerico implements Sensore {
9     private final SimpleStringProperty Nome;
10    private SimpleIntegerProperty Stato;
11    private SimpleStringProperty StringaStato;
12    private final SimpleStringProperty DataCreazione;
13    private final SimpleStringProperty DataAllarme;
14    private final SimpleStringProperty Parametro_riferimento;
15
16    //COSTRUTTORE USATO DALLE SOTTOCLASSI PER INIZIALIZZARE I PARAMETRI DI BASE DEI SENSORI
17    public SensoreGenerico(String N,int S,String Data_C,String Data_A,String Par_rif){
18        Nome=new SimpleStringProperty(N);
19        Stato=new SimpleIntegerProperty(S);
20        DataCreazione=new SimpleStringProperty(Data_C);
21        DataAllarme=new SimpleStringProperty(Data_A);
22        Parametro_riferimento=new SimpleStringProperty(Par_rif);
23
24        if(Stato.get() == 0){
25            StringaStato=new SimpleStringProperty( initialValue: "SPENTO");
26        }
27        else if(Stato.get() == 1){
28            StringaStato=new SimpleStringProperty( initialValue: "ACCESO");
29        }
30    }
31
32    //METODO DA SOVRASCRIVERE NELLE SOTTOCLASSI
33    public abstract String allarmeSuonato();
34
35    public String getNome() { return Nome.get(); }
36
37
38
39    public void setNome(String nome) { this.Nome.set(nome); }
40
41
42    public int getStato() { return Stato.get(); }
43
44
45    public void setStato(int stato) { this.Stato.set(stato); }
46
47
48    public String getStringaStato() { return StringaStato.get(); }
49
50
51    public void setStringaStato(String stringaStato) { this.StringaStato.set(stringaStato); }
52
53
54    public String getDataCreazione() { return DataCreazione.get(); }
55
56
57    private void setDataCreazione(String dataCreazione) { this.DataCreazione.set(dataCreazione); }
58
59
60    public String getDataAllarme() { return DataAllarme.get(); }
61
62
63    public void setDataAllarme(String dataAllarme) { this.DataAllarme.set(dataAllarme); }
64
65
66    public String getParametro_riferimento() { return Parametro_riferimento.get(); }
67
68
69    public void setParametro_riferimento(String parametro_riferimento) {
70        this.Parametro_riferimento.set(parametro_riferimento);
71    }
72
73
74
75
76
77
78
79
80
81
82
83 }
```

## Classe Prodotto Sensore Monitoraggio che eredita da Sensore Generico :

```
5 //CLASSE "SensoreMonitoraggio CHE ESTENDE "SensoreGenerico" E RAPPRESENTA I SENSORI ADIBITI AL MONITORAGGIO DELLA CASA
6 public class SensoreMonitoraggio extends SensoreGenerico {
7
8     private SimpleIntegerProperty Valore_Positivo;
9     private SimpleIntegerProperty Valore_Medio;
10    private SimpleIntegerProperty Valore_Critico;
11
12    public SensoreMonitoraggio(String N, int S,String Data_C,String Data_A,String P_Rif, int V_Pos, int V_Med, int V_Crit) {
13        super(N,S,Data_C,Data_A,P_Rif);
14        Valore_Positivo=new SimpleIntegerProperty(V_Pos);
15        Valore_Medio=new SimpleIntegerProperty(V_Med);
16        Valore_Critico=new SimpleIntegerProperty(V_Crit);
17    }
18
19    //METODO EREDITATO DA "SensoreGenerico" CHE PERMETTE DI MOSTRARE L'ALLARME
20    public String allarmeSuonato() { return "SENSORE MONITORAGGIO "+this.getNome()+" SCATTATO"; }
21
22
23
24    public int getValore_Positivo() { return Valore_Positivo.get(); }
25
26
27
28    public void setValore_Positivo(int valore_Positivo) { this.Valore_Positivo.set(valore_Positivo); }
29
30
31
32    public int getValore_Medio() { return Valore_Medio.get(); }
33
34
35
36    public void setValore_Medio(int valore_Medio) { this.Valore_Medio.set(valore_Medio); }
37
38
39
40    public int getValore_Critico() { return Valore_Critico.get(); }
41
42
43
44    public void setValore_Critico(int valore_Critico) { this.Valore_Critico.set(valore_Critico); }
45
46
47 }
```

## Classe Prodotto Sensore Intervento che eredita da Sensore Generico :

```
6 //CLASSE "SensoreIntervento CHE ESTENDE "SensoreGenerico" E RAPPRESENTA I SENSORI ADIBITI AGLI INTERVENTI NELLA CASA
7 public class SensoreIntervento extends SensoreGenerico {
8     SimpleStringProperty Azione_Eseguita;
9     SimpleStringProperty Comportamento_Azione;
10
11    public SensoreIntervento(String N,int S,String Data_C,String Data_A,String P_rif,String Azione,String Comport) {
12        super(N,S,Data_C,Data_A,P_rif);
13        Azione_Eseguita=new SimpleStringProperty(Azione);
14        Comportamento_Azione=new SimpleStringProperty(Comport);
15    }
16
17    //METODO EREDITATO DA "SensoreGenerico" CHE PERMETTE DI MOSTRARE L'ALLARME
18    public String allarmeSuonato() {
19        return "SENSORE INTERVENTO: "+this.getNome()+" SCATTATO "+getComportamento_Azione()+" "+getParametro_riferimento();
20    }
21
22
23
24    public String getAzione_Eseguita() { return Azione_Eseguita.get(); }
25
26
27
28    public void setAzione_Eseguita(String azione_Eseguita) { this.Azione_Eseguita.set(azione_Eseguita); }
29
30
31
32    public String getComportamento_Azione() { return Comportamento_Azione.get(); }
33
34
35
36    public void setComportamento_Azione(String comportamento_Azione) {
37        this.Comportamento_Azione.set(comportamento_Azione);
38    }
39
40 }
```

**Observer Pattern:** Pattern utilizzato per definire una dipendenza uno a molti tra un **parametro** e molti **sensori monitoraggio**, tale che se un parametro cambia stato, tutte le sue dipendenze (Sensori Monitoraggio) sono notificate e aggiornate automaticamente potendo così segnalare l'allarme corrispondente ed attivare i corrispondenti sensori intervento.

Classe astratta Subject per gli oggetti generici osservati:

```
5 //CLASSE GENERICA PER TUTTI I SUBJECT
6 public abstract class Subject {
7
8     //LISTA DI OSSERVATORI (SENSORI MONITORAGGIO) CHE CONTROLLERANNO I VALORI DEI PARAMETRI
9     private ArrayList<Observer> observerList;
10
11     //CLASSE OGGETTO OSSERVATO ASTRATTA CHE
12     public Subject(){
13         observerList=new ArrayList<>();
14     }
15
16     //METODI CHE VERRANNO EREDITATE DALLE CLASSI FIGLIE
17     public abstract void addObserver(Observer observer);
18     public abstract void removeObserver(Observer observer);
19     public abstract void notifyObservers();
20
21     public ArrayList<Observer> getObserverList() { return observerList; }
22
23     public void setObserverList(ArrayList<Observer> observerList) { this.observerList = observerList; }
24
25
26 }
27
28
```

Classe ParametroObservable per i parametri osservati:

```
8 //CLASSE CONCRETA PER I PARAMETRI OSSERVATI DAI SENSORI MONITORAGGIO
9 public class ParametroObservable extends Subject {
10
11     //CODA (FIFO) DI OSSERVATORI PER LA CORRETTA GESTIONE DEGLI ALLARMI
12     private Queue<Observer> Coda_Sensori;
13
14     private Parametri_Casa_Domotica Parametro;
15     private Double ValoreParametro;
16     private Double ValoreIniziale;
17
18     //COSTRUTTORE PER INIZIALIZZARE I PARAMETRI
19     public ParametroObservable(Parametri_Casa_Domotica P){
20         super();
21         Coda_Sensori=new LinkedList<>();
22         Parametro=P;
23         ValoreParametro=P.getValore();
24         ValoreIniziale=P.getValore();
25     }
26
27     public Queue<Observer> getCoda_Sensori() { return Coda_Sensori; }
28
29     public Parametri_Casa_Domotica getParametro() { return Parametro; }
30
31     public double getValoreParametro() { return ValoreParametro; }
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

39 //METODO PER SETTARE IL VALORE AGGIORNATO DI UN PARAMETRO E CONTROLLARE TRAMITE I
40 //SENSORI MONITORAGGIO(OSSERVATORI) CHE IL PARAMETRO NON SFORI IL RANGE
41 @ public void setValoreParametro(Double val_ParInput, String Tipo_Comportamento, Oggetti_Casa_Domotica Ogg) {
42
43     if( (Tipo_Comportamento.contains("INCREMENTA") && Ogg.getStato() == 0) || (Tipo_Comportamento.contains("DECREMENTA") && Ogg.getStato() == 1)){
44         ValoreParametro=this.getParametro().getValore()+val_ParInput;
45         if(ValoreParametro >= 10000){
46             ValoreParametro=10000.0;
47         }
48     }
49     else if( (Tipo_Comportamento.contains("DECREMENTA") && Ogg.getStato() == 0) || (Tipo_Comportamento.contains("INCREMENTA") && Ogg.getStato() == 1)){
50         ValoreParametro=this.getParametro().getValore()-val_ParInput;
51         if(ValoreParametro <= 0){
52             ValoreParametro=0.0;
53         }
54     }
55     this.Parametro.setValore(ValoreParametro);
56     notifyObservers();
57
58 }

```

```

60
61 //AGGIUNGE UN NUOVO OSSERVATORE
62 public void addObserver(Observer observer) { addObserverList().add(observer); }
63
64
65
66 //RIMUOVE UN OSSERVATORE
67 public void removeObserver(Observer observer) { addObserverList().remove(observer); }
68
69
70
71 //NOTIFICA LO STATO DEL PARAMETRO A TUTTI I SENSORI E IN CASO DI SFORAMENTO RANGE LI INSERISCE IN UNA CODA
72 public void notifyObservers() {
73     String Allarme_Sensore;
74     for(Observer obj: addObserverList()){
75         Allarme_Sensore=obj.update(this.getValoreParametro());
76
77         if(Allarme_Sensore != null){
78             Coda_Sensori.add(obj);
79         }
80     }
81 }
82 }

```

## Interfaccia per gli osservatori

```

3 public interface Observer {
4     String update(Double Valore);
5 }

```

## Classe di Osservatori (Sensori Monitoraggio) che implementa l'interfaccia Observer:

```
5 //CLASSE DI SENSORI MONITORAGGIO OSSERVATORI CHE SI OCCUPERANNO DI MONITORARE IL PARAMETRO
6 public class SensoreMonitoraggioObserver implements Observer{
7
8     SensoreMonitoraggio sensoreMonitoraggio;
9     private ParametroObservable Parametro;
10
11 @
12     public SensoreMonitoraggioObserver(SensoreMonitoraggio M,ParametroObservable P){
13         this.sensoreMonitoraggio=M;
14         Parametro=P;
15         System.out.println("Nuovo Osservatore per il Parametro: "+P.getParametro().getParametro());
16     }
17
18     public SensoreMonitoraggio getSensoreMonitoraggio() { return sensoreMonitoraggio; }
19
20
21 //CONTROLLA SE IL RANGE DEL SENSORE E RISPETTATO
22 public Boolean isRange(Double valore) {
23     int Val_Pos = sensoreMonitoraggio.getValore_Positivo();
24     int Val_Crit = sensoreMonitoraggio.getValore_Critico();
25
26     return (valore > Val_Crit && Val_Crit < Val_Pos) || (valore < Val_Crit && Val_Crit > Val_Pos);
27 }
28
29 //METODO EREDITATO DALL'INTERFACCIA 'OBSERVER' PER FAR CONTROLLARE AI SENSORI MONITORAGGIO CHE IL VALORE DEI PARAMETRI NON SFORI IL RANGE
30 @Override
31 public String update(Double Valore) {
32     if(!isRange(Valore) && this.getSensoreMonitoraggio().getStato()!=0) {
33         System.out.println(getSensoreMonitoraggio().allarmeSuonato());
34         return this.getSensoreMonitoraggio().allarmeSuonato();
35     }
36     return null;
37 }
38 }
```

## Gestione Eccezioni

Oltre le classiche eccezioni lanciate dal programma stesso, sono state implementate varie eccezioni personalizzate che possono essere lanciate in più punti a seconda della situazione con un determinato messaggio d'errore.

Classe per le Eccezioni legate ai problemi dell'account:

```
3 public class AccountException extends Exception {
4
5     String Err;
6     public AccountException(String Error){
7         super("Errore Account: ");
8         Err=Error;
9     }
10
11     @Override
12     public String toString() { return getMessage()+ Err; }
13
14 }
15
```

Classe per le Eccezioni legati ai problemi di gestione della casa domotica:

```
3 public class CasaException extends Exception{
4     String Err;
5     public CasaException(String Error){
6         super("Errore Casa Domotica : ");
7         Err=Error;
8     }
9
10     @Override
11     public String toString() { return getMessage()+ Err; }
12
13 }
14
```

Classe per le eccezioni legate ai problemi di gestione dei sensori:

```
3      public class SensoriException extends Exception{
4
5          String Err;
6      public SensoriException(String Error){
7          super("Errore Gestione Sensori: ");
8          Err=Error;
9      }
10
11      @Override
12      public String toString() { return getMessage()+ Err; }
15  }
```

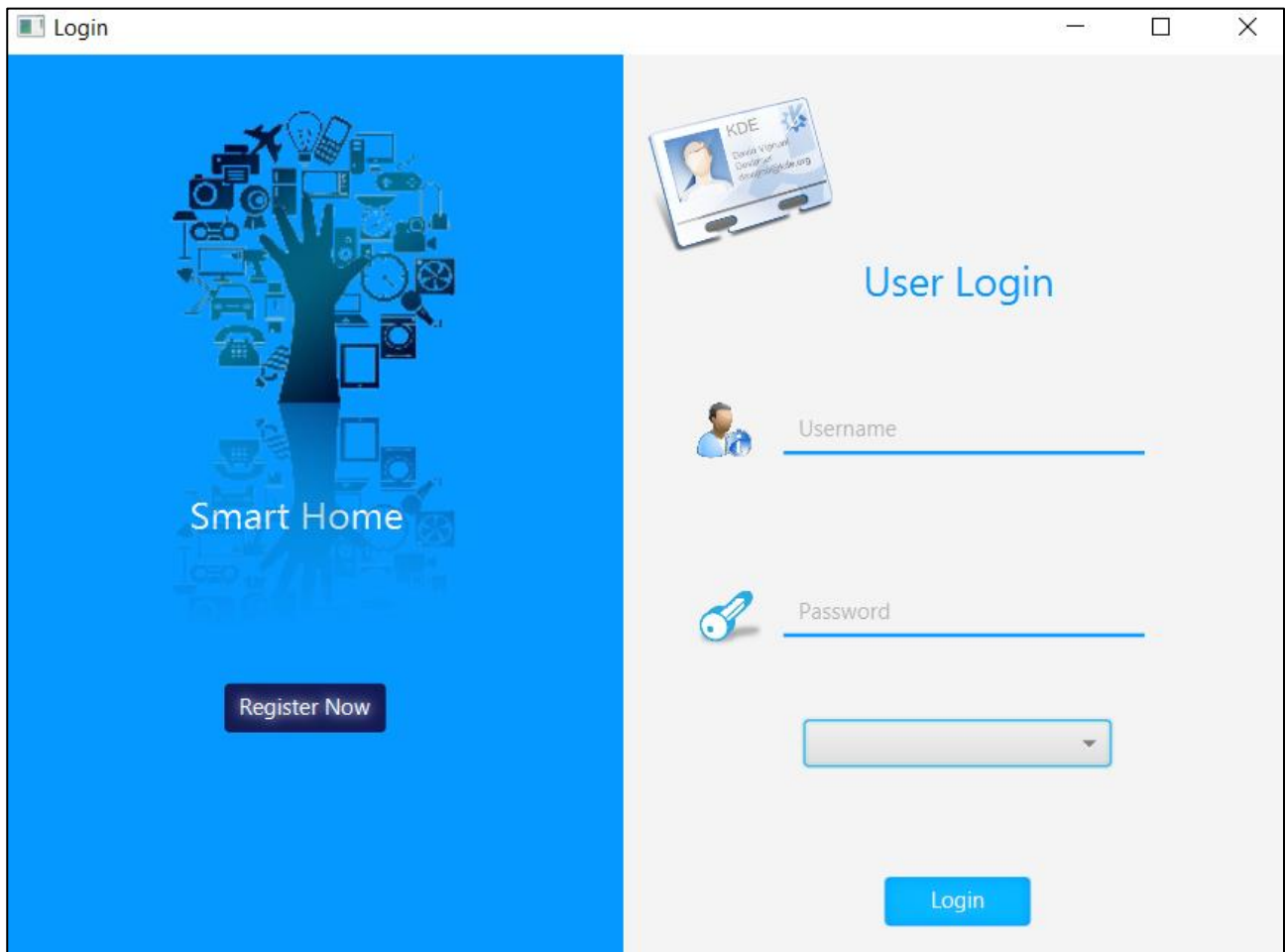
In questo caso, l'eccezione legata ai **problemi dell'account** viene lanciata a seguito di un errato inserimento delle credenziali di accesso dell'account oppure di una mancata scelta della tipologia di account:

```
    } else if (ChoiceAccount.getValue() == null) {
        throw new AccountException("Scegliere un account dal menu' a tendina");
    } else {
        throw new AccountException("Login: User o Password non corretti");
    }
} catch (Exception ex) {
    if (ex instanceof AccountException) {
        ErrorLabel1.setTextFill(Paint.valueOf("#800517"));
        this.ErrorLabel1.setText(ex.toString());
        System.err.println(ex.getMessage());
        ex.printStackTrace();
    }
    else{
        System.err.println(ex.getMessage());
        ex.printStackTrace();
    }
}
```



## Esecuzione

All'avvio del programma ci ritroveremo la seguente schermata dove potremo scegliere se accedere con un account di tipo **collaudo** o con un account di tipo **Attivato**, in alternativa, potremo registrarci ad una delle due tipologie di account tramite l'interfaccia dedicata alla **registrazione**.



Vediamo ora i vari casi che possiamo avere a partire da questa schermata.

## Caso Registrazione:

Register

Account Attivato

Name

Last Name

Email

Username

Password

Register

Back

Account Collaudo


Name

Last Name

Email

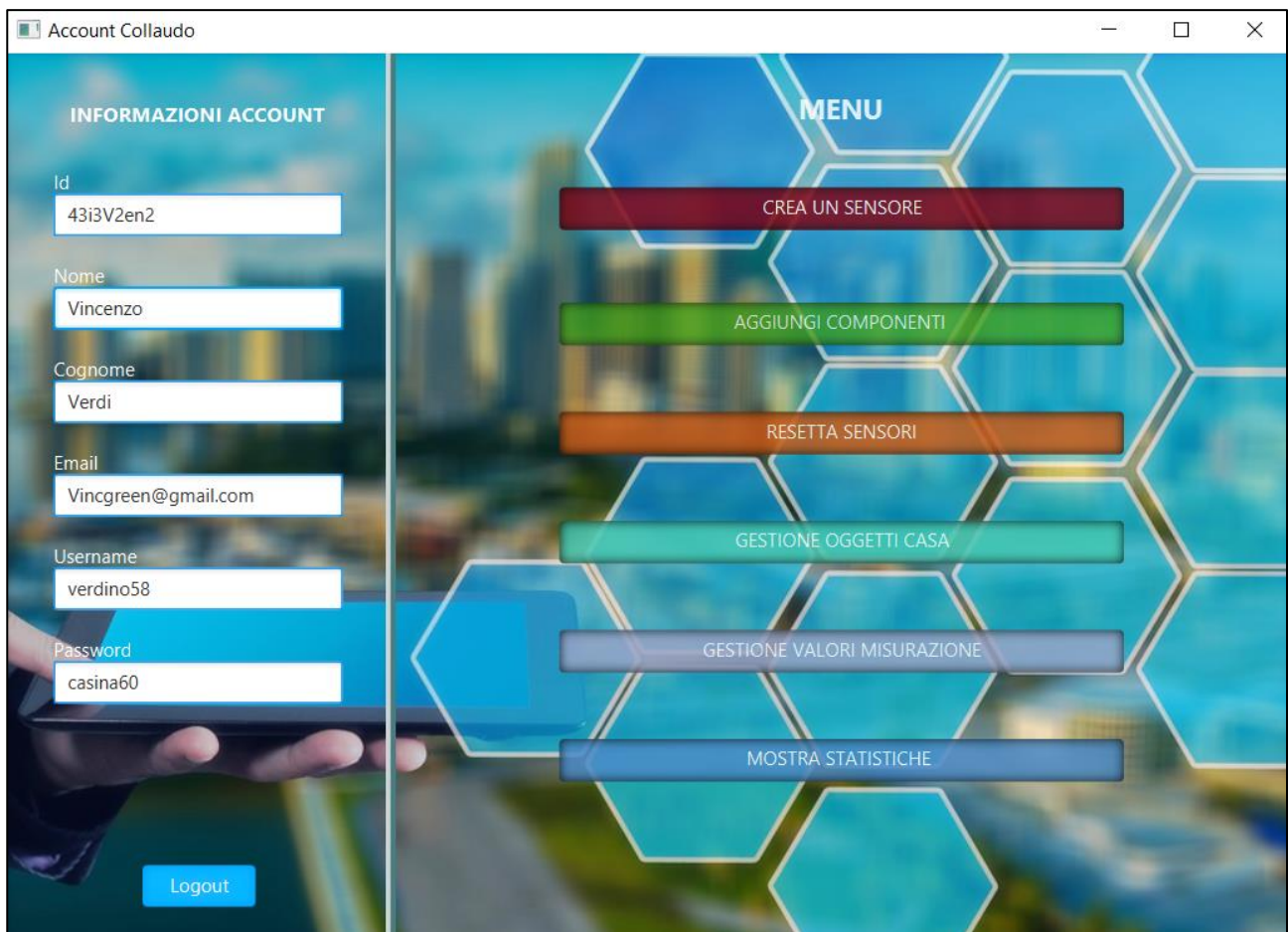
Password

Register



In questo caso potremo registrarci con una delle due tipologie di account inserendo delle credenziali di accesso valide.

## MODALITA' COLLAUDO



Ci ritroveremo con questa schermata con i dettagli dell'account e l'opzione di logout sul lato sinistro e le varie opzioni disponibili per gestire la casa domotica sulla destra.

## INSTALLAZIONE SENSORI

Se scegliamo **Crea un Sensore** , ci comparirà una finestra con due scelte possibili, ovvero se installare un sensore monitoraggio o un sensore intervento o in alternativa , un pulsante d'uscita:



### Scelta Sensore Monitoraggio:

The screenshot shows a window titled "Installazione Sensore Monitoraggio". The main heading is "SENSORE MONITORAGGIO". There are input fields for "NOME SENSORE", "PARAMETRO" (a dropdown), "VALORE POSITIVO" (a spinner set to 0), "VALORE MEDIO" (a spinner set to 0), and "VALORE CRITICO" (a spinner set to 0). There is a "CREA" button. Below these is a table with the following data:

NOME	STATO	STRINGA_STATO	DATA CREAZIONE	DATA ALLARME	PARAMETRO_RIF	Val_Pos	Val_Med	Val_Crit
FUGA_GAS	1	ACCESO	2021-02-08 04:35:49		GAS	0	500	1000
INCENDIO	1	ACCESO	2021-02-08 04:34:41	2021-02-08 22:38:10	TEMPERATURA	20	60	90
SOVRACCARICO	1	ACCESO	2021-02-08 04:35:06	2021-02-08 22:38:08	ELETTRICITA	0	1500	3000
TEMPERATURE_BASSE	1	ACCESO	2021-02-08 04:36:45		TEMPERATURA	20	15	0

Below the table are buttons "ELIMINA SENSORE" and "INTERRUTTORE". Below that is a section titled "SENSORI INTERVENTO COLLEGATI(SELEZIONARE UN SENSORE DALLA PRIMA TABELLA)". It contains a table with the following data:

NOME	STATO	STRINGA_STATO	DATA CREAZIONE	DATA ALLARME	PARAMETRO_RIF	DESCRIZIONE AZIONE	COMPORTAMENTO AZIONE
Nessun contenuto nella tabella							

At the bottom are buttons "ESCI" and "SCOLLEGA SENSORE".

In questa schermata potremo creare un nuovo sensore addetto al monitoraggio della casa domotica, inserendo un Nome Sensore, un Parametro di riferimento ed un range di valori delimitato da un valore positivo ed un valore critico ,in modo tale che il sensore possa rilevare eventuali criticità nella casa domotica, infine basta cliccare sul pulsante **Crea** per creare il sensore.

Abbiamo anche a disposizione un pulsante per **Eliminare** un sensore ed un pulsante **interruttore** per spegnere o accendere un sensore.

Ad ogni elemento selezionato dalla prima tabella verranno caricati tutti i sensori intervento collegati a quel sensore e sarà disponibile un pulsante per scollegare i due sensori selezionati dalle due tabelle.

### Scelta Sensore Intervento:

Installazione Sensore Intervento

SENORE INTERVENTO

NOME SENSORE

DESCRIZIONE AZIONE ...

PARAMETRO

COMPORTAMENTO AZIONE

CREA

NOME	STATO	STRINGA_STATO	DATA CREAZIONE	DATA ALLARME	PARAMETRO_RIF	DESCRIZIONE AZIONE	COMPORTAMENTO AZIONE
BLACKOUT	0	SPENTO	2021-02-08 04:37:19	2021-02-08 22:38:08	ELETTRICITA	EVITA DI SUPERARE I 3000 WATT	DECREMENTA IL PARAMETRO
GETTO ACQUA	0	SPENTO	2021-02-08 04:37:53	2021-02-08 22:38:10	TEMPERATURA	RAFFREDDA LA CASA O SPEGNE L'INCENDIO	DECREMENTA IL PARAMETRO
GETTO ARIA	0	SPENTO	2021-02-08 04:38:10	2021-02-08 22:38:10	TEMPERATURA	RAFFREDDA LA CASA	DECREMENTA IL PARAMETRO
RISCALDA LA CASA	0	SPENTO	2021-02-08 04:38:53		TEMPERATURA	RISCALDA LA CASA IN CASO DI TEMPERATUR...	INCREMENTA IL PARAMETRO

ELIMINA SENSORE

INTERRUTTORE

SENSORI MONITORAGGIO DA COLLEGARE (SELEZIONARE UN SENSORE DALLA PRIMA TABELLA)

NOME	STATO	STRINGA_STATO	DATA CREAZIONE	DATA ALLARME	PARAMETRO_RIF	Val_Pos	Val_Med	Val_Crit
Nessun contenuto nella tabella								

ESCI

COLLEGA SENSORE

In questa schermata potremo creare un nuovo sensore addetto agli interventi della casa domotica, inserendo un Nome Sensore, un Parametro di riferimento, un comportamento del sensore che potrà essere l'incremento o il decremento di un determinato parametro al fine di riportare la casa ai valori di base ed una descrizione dell'azione eseguita dal sensore

(Es: Getta acqua per spegnere un incendio o raffreddare la casa per le temperature eccessive) .

infine basta cliccare sul pulsante **Crea** per creare il sensore.

Abbiamo anche a disposizione un pulsante per **Eliminare** un sensore ed un pulsante **interruttore** per spegnere o accendere un sensore.

Ad ogni elemento selezionato dalla prima tabella verranno caricati tutti i sensori Monitoraggio ancora da collegare a quel sensore e sarà disponibile un pulsante per collegare i due sensori selezionati dalle due tabelle.

#### AGGIUNGI COMPONENTI

Ci comparirà un finestra dove potremo creare un nuovo componente da collegare ad una delle due tipologie di sensori:

The screenshot shows a window titled "Aggiungi Componenti" with a teal background. At the top, the word "COMPONENTE" is written in red. Below it, there are two input fields: "NOME COMPONENTE" and "DESCRIZIONE AZIONE". A "CREA" button is positioned between them. Below the "CREA" button is a table with three columns: "NOME COMPONENTE", "DATA CREAZIONE", and "DESCRIZIONE AZIONE". The table contains three rows of data. Below the table is an "ELIMINA COMPONENTE" button. At the bottom of the window, there are two sections: "COLLEGA O SCOLLEGA UN COMPONENTE AD UN SENSORE(SELEZIONA DALLA PRIMA TABELLA)" and "SENSORI INTERVENTO". Each section has a "COLLEGA COMPONENTE" button, a dropdown menu, and a "SCOLLEGA COMPONENTE" button. An "ESCI" button is located at the bottom left.

NOME COMPONENTE	DATA CREAZIONE	DESCRIZIONE AZIONE
LUCE DI EMERGENZA1	2021-02-07 20:15:15	ILLUMINA
TELECAMERA1	2021-02-07 20:09:48	FURTI
TELECAMERA2	2021-02-07 20:10:13	MONITORAGGIO BAMBINI



In questa schermata potremo creare un nuovo Componente da collegare ad un sensore , inserendo un Nome Componente ed una breve descrizione.

infine basta cliccare sul pulsante **Crea** per creare il nuovo componente, abbiamo anche a disposizione un pulsante per **Eliminare** un Componente.

Nella zona sottostante sarà possibile scegliere i sensori (monitoraggio o intervento) ancora non collegati e collegarli al componente selezionato nella prima tabella, inoltre sarà possibile scollegare i sensori già collegati al suddetto componente.

#### RESETTA SENSORI

Ci comparirà una finestra dove avremo due opzioni disponibili, ovvero quella di resettare i sensori monitoraggio azzerando l'ultimo allarme scattato e accendendoli tutti come impostazione di creazione ed infine quella di resettare tutti i sensori intervento azzerando l'ultimo allarme scattato e spegnendoli tutti come da impostazione di creazione:



Ci comparirà una finestra dove potremo scegliere di creare degli oggetti per la nostra casa domotica e dove potremo creare dei comportamenti per i suddetti oggetti:

Creazione Oggetto e Comportamenti

### CREAZIONE OGGETTO

NOME OGGETTO

MARCA OGGETTO

SERIALE	NOME	MARCA	STATO	STRINGA STATO
04229187-1AA8-...	CLIMATIZZATORE	VORTICE	0	SPENTO
8BE46DE3-9B77-...	COMPUTER1	HP	0	SPENTO
FAC88448-6AC5-...	FRULLATORE1	KENWOOD	0	SPENTO
21AB5BCE-E7E3-...	LAVATRICE	HOOVER	0	SPENTO
05DFFB7C-3B59-...	PLAYSTATION 4	SONY	0	SPENTO
ED150E9C-C823-...	SCALDINO1	DELONGHI	0	SPENTO
2AD1B875-C53F-...	SCALDINO2	DELONGHI	0	SPENTO
F467BD4A-7A69-...	TELEVISORE1	SAMSUNG	0	SPENTO
02B13C7E-A012-...	TELEVISORE2	SHARP	0	SPENTO
3A19B70A-C94C-...	TERMOFONE1	IRSAP	0	SPENTO

### CREAZIONE COMPORTAMENTO

(SELEZIONA UN OGGETTO DALLA PRIMA TABELLA)

PARAMETRO OGGETTO

COMPORTAMENTO

VALORE DI CONSUMO

SERIALE OGGETTO	PARAMETRO	COMPORTAMENTO	CONSUMO
Nessun contenuto nella tabella			

In questa schermata potremo creare un nuovo Oggetto , inserendo un Nome Oggetto ed una Marca Oggetto.

infine basta cliccare sul pulsante **Crea** per creare il nuovo oggetto, abbiamo anche a disposizione un pulsante per **Eliminare** un oggetto (Cancella la riga selezionata).

Ad ogni elemento selezionato dalla prima tabella verranno caricati tutti i comportamenti dell'oggetto, inoltre sarà possibile creare un nuovo comportamento inserendo un Parametro di riferimento, una Tipologia di Comportamento(Incremento o Decremento) di un parametro ed il valore di consumo



(Es: Scaldino della Delonghi incrementa il consumo di elettricità di 1500 watt, inoltre incrementa la temperatura della casa domotica di circa 10 gradi celsius in un'ora). Sarà possibile creare per un oggetto solo un comportamento per ogni parametro.

## GESTIONE VALORI MISURAZIONE

Ci comparirà una finestra dove potremo scegliere di creare dei parametri per la nostra casa domotica:

PARAMETRO	UNITA' DI MISURA	VALORE
ELETTRICITA'	WATT	0.0
GAS	CM^3/H	0.0
PRESSIONE	BAR	0.0
TEMPERATURA	CELSIUS	20.0

In questa schermata potremo creare un nuovo Parametro, inserendo un Nome Parametro, un'Unità di Misura ed un Valore Iniziale.

infine basta cliccare sul pulsante **Crea** per creare il nuovo parametro, abbiamo anche a disposizione un pulsante per **Eliminare** un parametro (Cancella la riga selezionata).

## MOSTRA STATISTICHE

Ci comparirà una finestra dove potremo tenere sotto controllo tutti i sensori della casa domotica con tutte le date degli ultimi allarmi, tutti gli oggetti, i componenti e i parametri della casa:

Mostra Statistiche									
SENSORI MONITORAGGIO									
NOME	STATO	STRINGA_STATO	DATA CREAZIONE	DATA ALLARME	PARAMETRO_RIF	Val_Pos	Val_Med	Val_Crit	+
FUGA_GAS	1	ACCESO	2021-02-08 04:35:49		GAS	0	500	1000	
INCENDIO	1	ACCESO	2021-02-08 04:34:41	2021-02-08 22:38:10	TEMPERATURA	20	60	90	
SOVRACCARICO	1	ACCESO	2021-02-08 04:35:06	2021-02-08 22:38:08	ELETTRICITA	0	1500	3000	
TEMPERATURE_BASSE	1	ACCESO	2021-02-08 04:36:45		TEMPERATURA	20	15	0	
SENSORI INTERVENTO									
NOME	STATO	STRINGA_STATO	DATA CREAZIONE	DATA ALLARME	PARAMETRO_RIF	DESCRIZIONE AZIONE	COMPORTAMENTO AZIONE		+
BLACKOUT	0	SPENTO	2021-02-08 04:37:19	2021-02-08 22:38:08	ELETTRICITA	EVITA DI SUPERARE I 3000 WATT	DECREMENTA IL PARAMETRO		
GETTO ACQUA	0	SPENTO	2021-02-08 04:37:53	2021-02-08 22:38:10	TEMPERATURA	RAFFREDDA LA CASA O SPEGNE L'INCENDIO	DECREMENTA IL PARAMETRO		
GETTO ARIA	0	SPENTO	2021-02-08 04:38:10	2021-02-08 22:38:10	TEMPERATURA	RAFFREDDA LA CASA	DECREMENTA IL PARAMETRO		
RISCALDA LA CASA	0	SPENTO	2021-02-08 04:38:53		TEMPERATURA	RISCALDA LA CASA IN CASO DI TEMPERATUR...	INCREMENTA IL PARAMETRO		
COMPONENTI					OGGETTI				
NOME COMPONENTE	DATA CREAZIONE	DESCRIZIONE AZIONE		+	SERIALE	NOME	MARCA	STATO	STRINGA STATO
LUCE DI EMERGENZA1	2021-02-07 20:15:15	ILLUMINA			04229187-1AA...	CLIMATIZZA...	VORTICE	0	SPENTO
TELECAMERA1	2021-02-07 20:09:48	FURTI			8BE46DE3-9B7...	COMPUTER1	HP	0	SPENTO
TELECAMERA2	2021-02-07 20:10:13	MONITORAGGIO BAMBINI			FAC88448-6AC...	FRULLATORE1	KENWOOD	0	SPENTO
PARAMETRI									
PARAMETRO	UNITA' DI MISURA	VALORE		+					
ELETTRICITA	WATT	0.0							
GAS	CM^3/H	0.0							

## MODALITA' INTERVENTO

Account Attivato

**INFORMAZIONI ACCOUNT**  
Id  
32t2R2Rb2  
  
Nome  
Roberto  
  
Cognome  
Rossi  
  
Email  
Robred@yahoo.it  
  
Username  
casetta  
  
Password  
casina58  
  
Logout

**CASA DOMOTICA**  
**CONTROLLO OGGETTI**

SERIALE	NOME	MARCA	STATO	STRINGA STATO
04229187-1AA8...	CLIMATIZZATORE	VORTICE	0	SPENTO
8BE46DE3-9B77...	COMPUTER1	HP	0	SPENTO
FAC88448-6AC5...	FRULLATORE1	KENWOOD	0	SPENTO
21AB5BCE-E7E3...	LAVATRICE	HOOVER	0	SPENTO
05DFFB7C-3B59...	PLAYSTATION 4	SONY	0	SPENTO
ED150E9C-C823...	SCALDINO1	DELONGHI	0	SPENTO
2AD18B75-C53F...	SCALDINO2	DELONGHI	0	SPENTO
F467BD4A-7A69...	TELEVISORE1	SAMSUNG	0	SPENTO
02B13C7E-A012...	TELEVISORE2	SHARP	0	SPENTO

  
INTERUTTORE

**ALLARMI SENSORI SCATTATI**  
  
CLEAR

**CONTROLLO PARAMETRI CASA**

PARAMETRO	UNITA' DI MISURA	VALORE
ELETTRICITA	WATT	0.0
GAS	CM^3/H	0.0
PRESSIONE	BAR	0.0
TEMPERATURA	CELSIUS	20.0

**AGGIORNAMENTO VALORI**  
  
CLEAR

Ci ritroveremo con questa schermata con i dettagli dell'account e l'opzione di logout sul lato sinistro, mentre sul lato destro ci ritroveremo con l'ambiente di testing della casa domotica:

1. Potremo accendere o spegnere gli oggetti e controllare i parametri della casa nel riquadro inferiore.
2. Potremo monitorare tutti gli allarmi dei sensori monitoraggio e tutte le attivazioni dei sensori intervento nella finestra di log: **Allarmi Sensori Scattati**.
3. Potremo monitorare l'eventuale decremento o incremento di un parametro a causa di un sensore scattato e quindi dell'effetto dei suoi sensori intervento nella finestra di log: **Aggiornamento Valori**.