Air Analyzer

Versione: 1.0.1

Data di rilascio: 18/12/2020

**Realizzato da**

Davide Palladino

[me@davidepalladino.com](mailto:me@davidepalladino.com)

[www.davidepalladino.com](http://www.davidepalladino.com)

Indice

1. Product Backlog 3

1.1 Introduzione 3

1.2 Contesto di business 3

1.3 Stakeholder 3

1.3.1 Scrum Master 3

1.3.2 Product Owner 3

1.3.3 Developer 3

1.4 Item funzionali 3

1.4.1 IF-1 Visualizzazione della temperatura su monitor 3

1.4.2 IF-2 Visualizzazione dell’umidità su monitor 3

1.4.3 IF-3 Visualizzazione esterna dei valori memorizzati 3

1.4.4 IF-4 Visualizzazione dello percentuale della batteria 3

1.5 Item informativi 4

1.5.1 II-1 Temperatura 4

1.5.2 II-2 Umidità 4

1.5.3 II-3 Batteria 4

1.6 Item di interfaccia 4

1.6.1 IU-1 Pagina principale 4

1.7 Altri Item 4

1.7.1 AI-1 Memorizzazione esterna 4

1.7.2 AI-2 ID del dispositivo 4

2. Sprint Report 5

2.1 Sprint Backlog 5

2.2 Product Requirement Specification 6

2.2.1 Diagramma dei Casi d’uso 6

2.3 Detailed Product Design 6

2.3.1 Diagramma delle Classi 6

2.3.2 Specifiche delle Classi 6

2.4 Data modeling and design 7

2.4.1 Modello logico del Database 7

2.4.2 Struttura fisica del Database 7

3. Glossario 8

3.1 Definizioni 8

# Product Backlog

## Introduzione

Air Analyzer è un sistema che nasce per monitorare la qualità dell’aria in temperatura e umidità, espressi rispettivamente in gradi Celsius ed in percentuale. I valori saranno sia visualizzati in real-time che memorizzati all’interno di un database.

## Contesto di business

Air Analyzer vuole essere una soluzione ai semplici sistemi correlati, i quali sono limitati alla sola visualizzazione del corrente stato. Mediante questo sistema è possibile memorizzare i dati su di una struttura esterna quale un database. Quest’ultimo, infatti, è accessibile esternamente al fine di effettuare un’analisi dei dati e poter, quindi, prendere provvedimenti per il miglioramento della qualità dell’aria.

## Stakeholder

### Scrum Master

Davide Palladino

### Product Owner

Davide Palladino

### Developer

Davide Palladino

## Item funzionali

### IF-1 Visualizzazione della temperatura su monitor

Come utente voglio visualizzare la temperatura percepita su schermo ed al momento, così che possa migliorarla.

### IF-2 Visualizzazione dell’umidità su monitor

Come utente voglio visualizzare l’umidità percepita su schermo ed al momento, così che possa migliorarla.

### IF-3 Visualizzazione esterna dei valori memorizzati

Come utente voglio accedere esternamente ai valori precedentemente percepiti così che possa fare analisi e migliorare la qualità dell’aria.

### IF-4 Visualizzazione della percentuale della batteria

Come utente voglio visualizzare la percentuale di carica della batteria cosi che possa cambiarla all’evenienza.

## Item informativi

### II-1 Temperatura

La temperatura sarà caratterizzata da un valore espresso in gradi Celsius e dalla data di rilevazione.

### II-2 Umidità

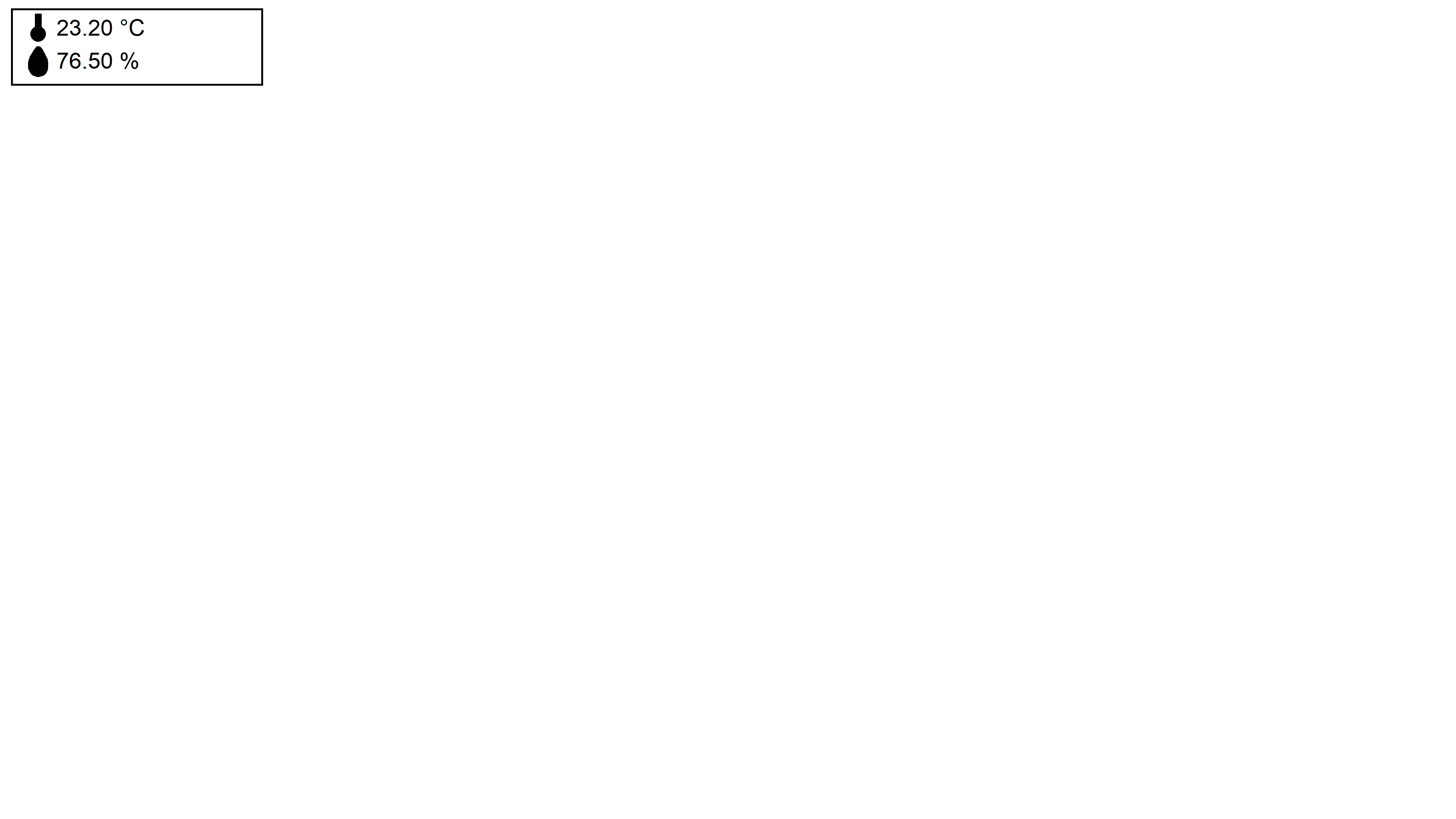
L’umidità sarà considerata relativa e sarà caratterizzata da un valore espresso in percentuale e dalla data di rilevazione.

### II-3 Batteria

La batteria, collegata al dispositivo, verrà indicata con un valore espresso in percentuale.

## Item di interfaccia

### IU-1 Pagina principale

****

## Altri Item

### AI-1 Memorizzazione esterna

I valori di temperatura e umidità letti saranno memorizzati in un database, per una successiva visione ed analisi. Ogni coppia di valori avrà la data e l’ora di rilevazione. Se la batteria è al di sotto del 30%, i valori non saranno memorizzati.

### AI-2 ID del dispositivo

Ogni dispositivo ha un ID univoco ed un nome, per la memorizzazione delle informazioni all’interno del database.

### AI-3 Batteria

Data la necessità di una connettività costante, si necessità come implicazione una batteria con un amperaggio ad ore superiore a quelle in commercio.

# Sprint Report

## Sprint Backlog

Tabella di riepilogo che indica, per ognuno degli Sprint successivi allo Sprint n.0, la lista degli item del Product Backlog, evidenziando quelli che verranno implementati nell’ambito dello sprint corrente unitamente ad una descrizione esplicativa.

Per semplificare l’esposizione e salvaguardare la tracciabilità tra semilavorati si è proceduto alle seguenti assunzioni:

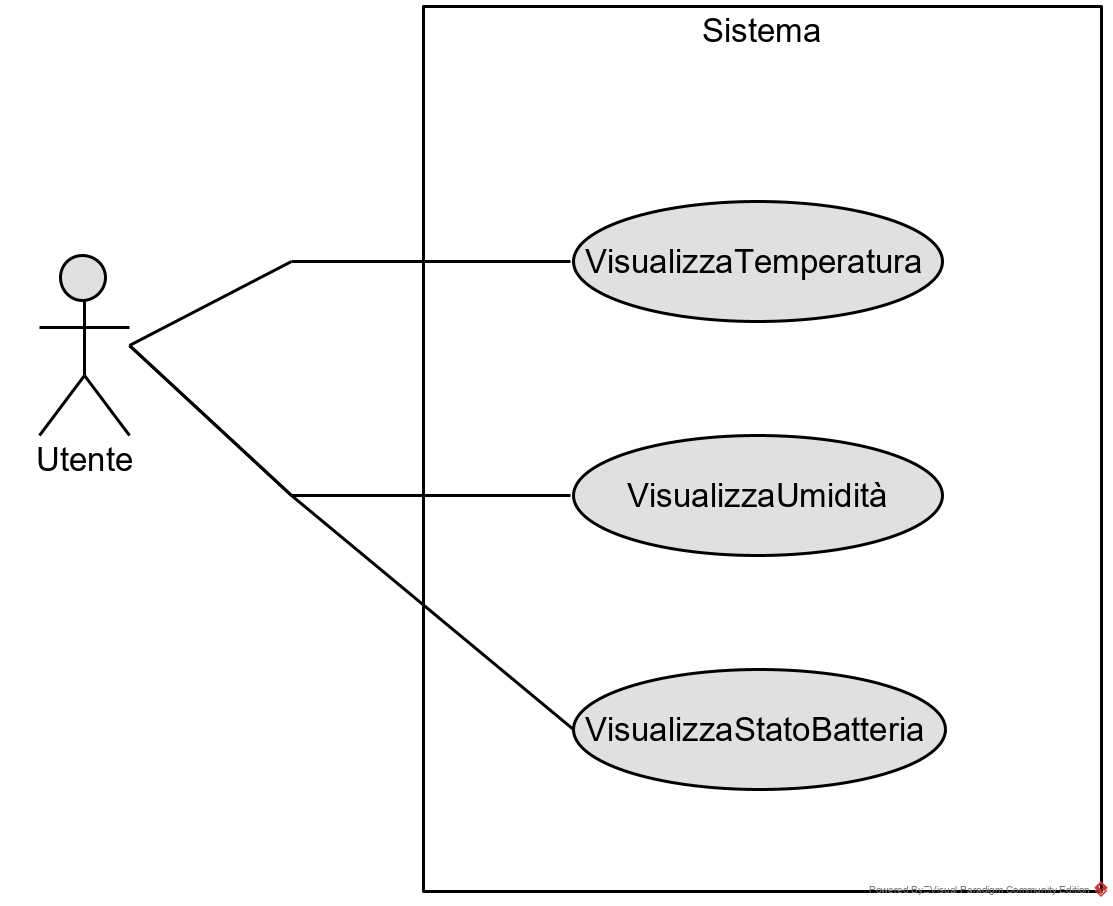
* All’interno di uno Sprint sono implementati un sottoinsieme di item tra quelli specificati nel Product Backlog
* Lo Sprint Backlog relativo allo sprint corrente contiene pertanto l’insieme degli item del Product Backlog in corso di implementazione
* Gli Item funzionali, ovvero le User Stories dovranno essere tracciabili uno ad uno, auspicabilmente seppur non necessariamente, con i casi d’uso
* Ad ogni caso d’uso dovrà essere associato uno scenario di base più gli eventuali scenari alternativi. Lo scenario in prima istanza viene redatto a partire dalla specifica della User Story riportata nel Product Backlog
* Ad ogni caso d’uso dovrà essere associato un diagramma di sequenza.

Ogni sprint deve necessariamente produrre in output del codice funzionante. L’unica eccezione è rappresentata dallo Sprint n°0 che deve essere utilizzato per disegnare la macro-architettura del sistema con le sue componenti e le sue interfacce, e che sarà utilizzata come roadmap per gli sprint successivi andando a chiarire dove si colloca quanto realizzato in ciascuno di essi.

|  |  |  |
| --- | --- | --- |
| **Numero Sprint** | **Codice Item** | **Note** |
| Sprint 1 | IF-1 | Visualizzazione della temperatura su monitor |
| IF-2 | Visualizzazione dell’umidità su monitor |
| Sprint 2 | IF-3 | Visualizzazione esterna dei valori memorizzati |

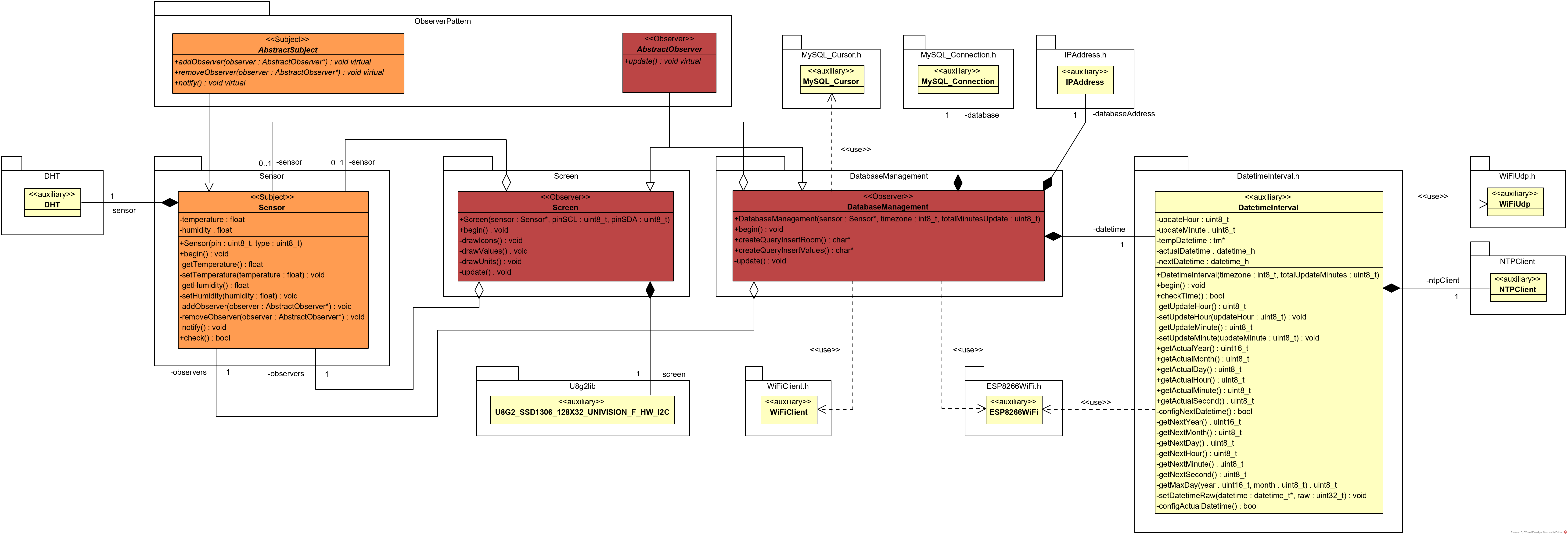
## Product Requirement Specification

### Diagramma dei Casi d’uso



## Detailed Product Design

### Diagramma delle Classi

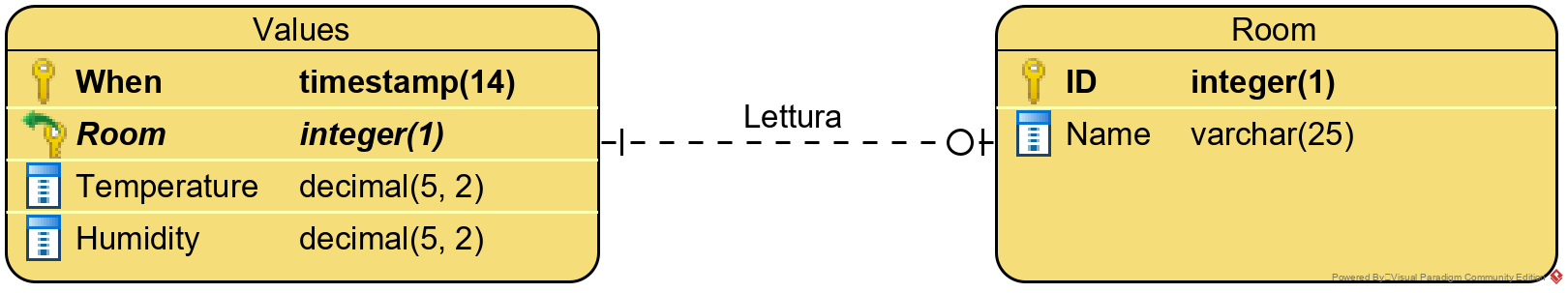


### Specifiche delle Classi

|  |  |  |
| --- | --- | --- |
| **Ruolo** | **Classe** | **Comportamento** |
| Subject | AbstractSubject | Questa classe astratta serve allo sviluppo delle sottoclassi annesse. |
| Observer | AbstractObserver |
| Subject | Sensor | Questa classe permette di leggere i valori di temperatura e umidità, ed usa la classe DHT per tali operazioni. |
| Observer | Screen | Questa classe permette la visualizzazione su monitor dei valori letti dalla classe Sensor. Utilizza la classe U8G2\_SSD1306\_128X32\_UNIVISION\_F\_HW\_I2C per tali operazioni. |
| DatabaseManagement | Questa classe permette di memorizzare, all’interno di un database relazionale MySQL, i valori letti insieme alla data ed il codice della stanza. |
| Auxiliary | DHT | Questa classe fornisce i servizi richiesti dalla classe Sensor. |
| U8G2\_SSD1306\_128X32\_  UNIVISION\_F\_HW\_I2C | Questa classe fornisce i servizi richiesti dalla classe Screen. |
| WiFiClient | Questa classe fornisce i servizi richiesti dalla classe DatabaseManagement. |
| MySQL\_Connection |
| MySQL\_Cursor |
| IPAddress |
| DatetimeInterval |
| ESP8266WiFi | Questa classe fornisce i servizi richiesti dalla classe DatabaseManagement e DatetimeInterval. |
| WiFiUdp | Questa classe fornisce i servizi richiesti dalla classe DatetimeInterval. |
| NTPClient |

## Data modeling and design

### Modello logico del Database



### Struttura fisica del Database

Il diagramma UML precedentemente illustrato è basato sul modello relazionale, gestito dal DBMS MySQL alla versione 8.0.22.

# Glossario

## Definizioni

**Utente** – colui che usufruisce delle informazioni elaborate dal sistema, come la temperatura e l’umidità percepita nell’aria sia al momento che in precedenza.

**Temperatura** – proprietà fisica, definita per mezzo di una grandezza scalare, che esprime lo stato termico dell’aria. Essa sarà espressa in gradi Celsius.

**Umidità** – la quantità di acqua (o vapore acqueo) contenuta nell’atmosfera. Essa sarà di tipo relativa, quindi espressa in percentuale. In particolare, sarà il rapporto tra umidità assoluta ed umidità massima possibile.

**Batteria** – dispositivo esterno che alimenta il sistema, dotato di un voltaggio pari a 9V.