Specifica tecnica

A.A. 2022-2023

Componenti del gruppo:

Andrea Crocco, matr. 1226135
Elena Fabris, matr. 2008072
Sonia Franco, matr. 1224437
Andrea Stecca, matr. 2016104
Filippo Tonini, matr. 2008080
Enrico Zangrando, matr. 2000547
e-mail: *next.team.swe@gmail.com*

Università
degli Studi
di Padova

Indice

Registro delle modifiche

Versione	Descrizione	Verificatore	Autore	Data
1.0	Approvazione	Non richiesto	Andrea Stecca	2023-08-15
0.4	Aggiunti i diagrammi UML del frontend e relativa descrizione	Sonia Franco	Enrico Zangrando	2023-08-14
0.3	Aggiunti i diagrammi UML del backend e relativa descrizione	Filippo Tonini	Enrico Zangrando	2023-07-29
0.2	Aggiornati i requisiti implementati	Filippo Tonini	Enrico Zangrando	2023-07-28
0.1	Aggiunti i requisiti soddisfatti e non	Filippo Tonini	Enrico Zangrando	2023-06-20

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di fornire una descrizione dell'architettura del prodotto, delle scelte architetturali adottate dal team e delle tecnologie utilizzate. La specifica tecnica indica inoltre quanti requisiti sono stati soddisfatti.

1.2 Descrizione del progetto

L'obiettivo del capitolato è la creazione di un sistema di illuminazione con rilevamento della presenza di persone attraverso l'utilizzo di *sensori*$_G$ e lo sviluppo di un'applicazione web *responsive*$_G$ in grado di monitorare un sistema di illuminazione pubblico. In particolare, il sistema deve essere in grado di:

- Rilevare la presenza di persone in prossimità della fonte luminosa
- Aumentare o ridurre l'intensità luminosa di un singolo lampione o di intere aree
- Rilevare automaticamente il guasto di un *impianto di illuminazione*$_G$
- Permettere la segnalazione manuale dei guasti di un impianto di illuminazione
- Permettere l'inserimento e la gestione di un impianto luminoso

2 Architettura

L'applicazione sfrutta un'architettura client-server.

2.1 Back-end

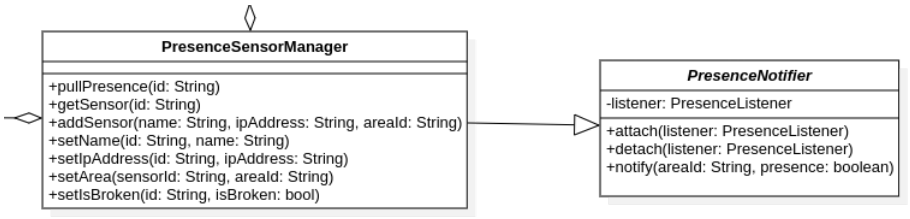
Il backend ha lo scopo di:

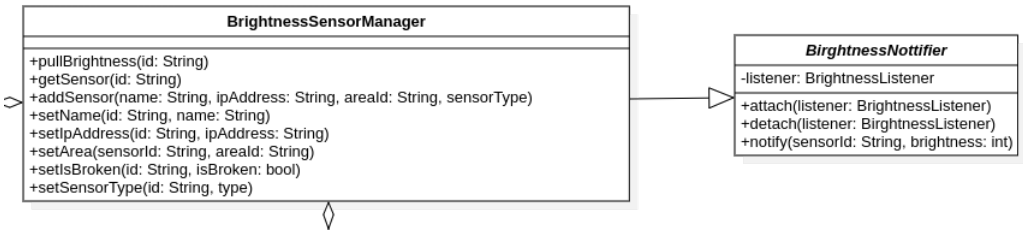
- Controllare l'accensione e lo spegnimento delle luci;
- Gestire le liste di luci, aree e sensori;
- Gestire la lista dei dispositivi guasti;
- Intervenire rispetto a ciò che viene segnalato dai sensori;
- Fornire servizi per visualizzare e modificare i dati riguardanti le liste sopracitate ai client collegati;
- Gestire l'autenticazione e registrazione degli utenti;

Il backend è stato realizzato con Nodejs utilizzando il linguaggio TypeScript.

2.1.1 Design pattern

Nel backend viene utilizzato il pattern observer, in modo che i sensori siano in grado di segnalare ciò che rilevano alla classe che si occupa della gestione delle luci.





2.2 Front-end

Il frontend è una webapp che deve:

- Consentire l'accesso e registrazione degli utenti;
- Consentire agli utenti di:
  - Visualizzare, inserire, rimuovere o modificare aree, luci, sensori;
  - Visualizzare la lista dei dispositivi guasti e poter inserire o rimuovere dispositivi ad essa;
  - Accendere e spegnere delle aree manualmente;
  - Gestire manualmente la luminosità e l'accensione di un'area;

La webapp è sviluppata utilizzando il framework Flutter e linguaggio Dart.

2.2.1 Pattern architetturale e di design

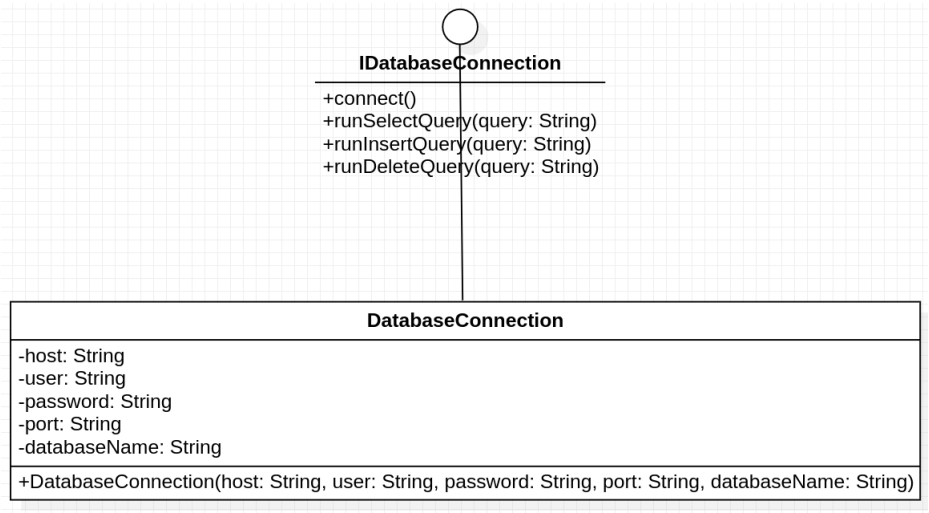
La webapp utilizza un'architettura model-view-controller.

3 Diagrammi delle classi

Di seguito vengono riportati i diagrammi delle classi che compongono backend e frontend.

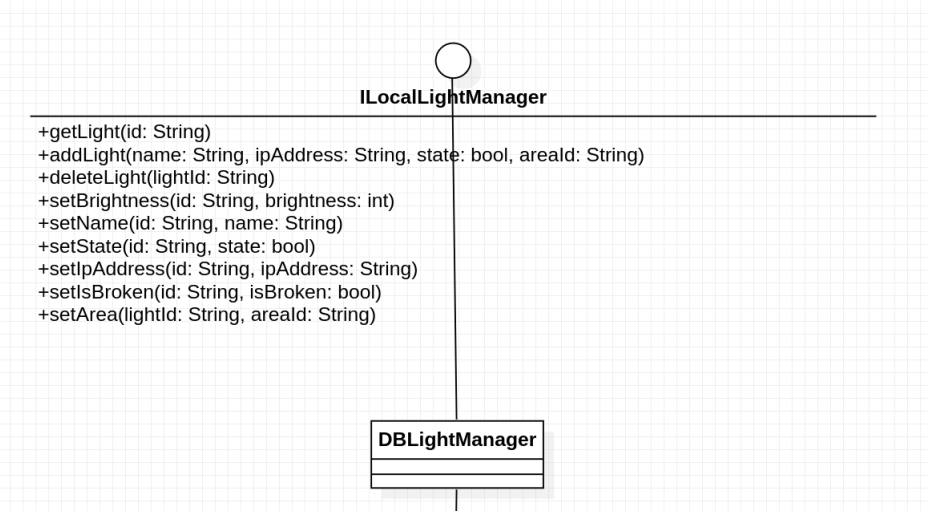
3.1 Backend

DatabaseConnection



La classe *DatabaseConnection* ha il compito di gestire la connessione al database ed eseguire le query richieste da altre classi.

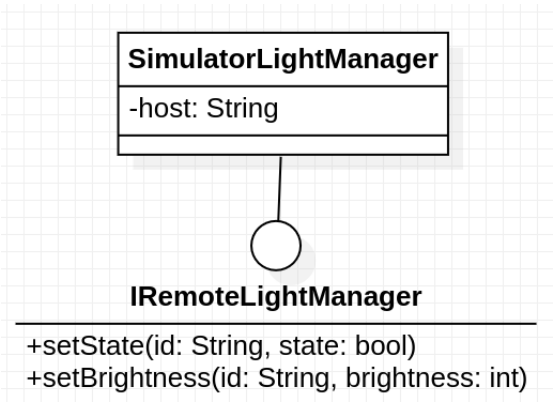
LocalLightManager



ILocalLightManager rappresenta il contratto che devono rispettare le classi che si prospettano di gestire localmente i dati dei dispositivi d'illuminazione. Nel nostro caso l'interfaccia viene implementata dalla classe *DBLightManager* la quale contiene un'istanza della classe *DatabaseConnection* poiché la gestione locale dei dati avviene, nel nostro caso, per mezzo di un database.

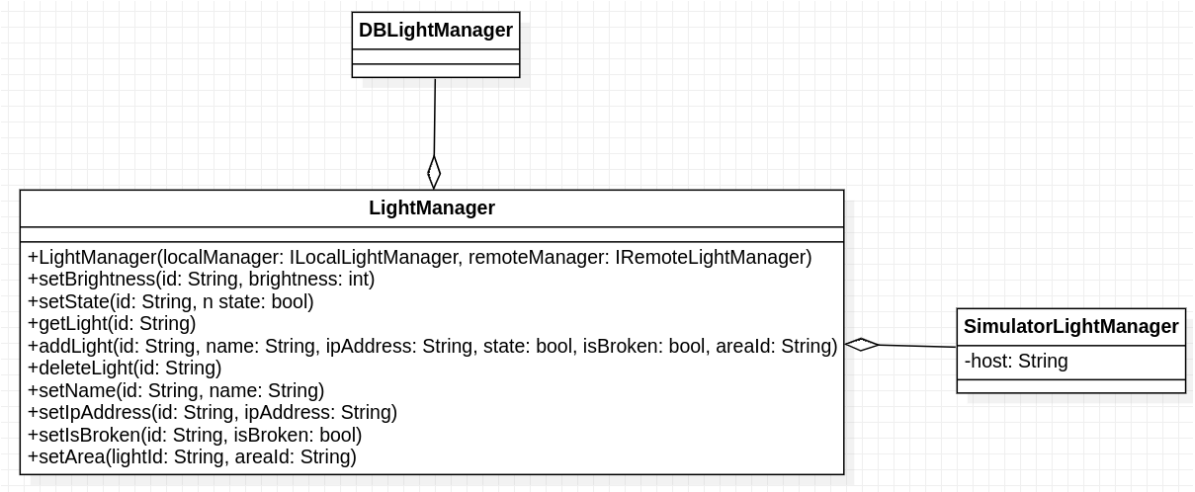
RemoteLightManager

IRemoteLightManager rappresenta il contratto che devono rispettare le classi che si prospettano di gestire i dispositivi d'illuminazione remoti. Esso prevede due metodi: *setState* per accendere un dispositivo e *setBrightness* per impostarne la luminosità. Nel nostro caso l'interfaccia viene implementata dalla classe



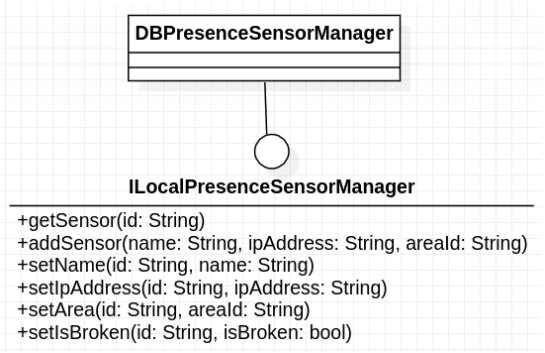
SimulatorLightManager che implementa i metodi di accensione e luminosità tramite richieste HTTP ad un simulatore.

LightManager



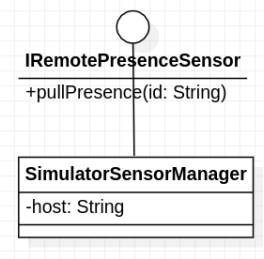
La classe *LightManager* offre dei metodi per la gestione generale dei dispositivi d'illuminazione. Essa contiene un'istanza di *ILocalLightManager* ed una di *IRemoteLightManager* in modo da poter effettuare le operazioni sui dispositivi remoti ed in caso di successo modificare i relativi dati localmente.

LocalPresenceSensor



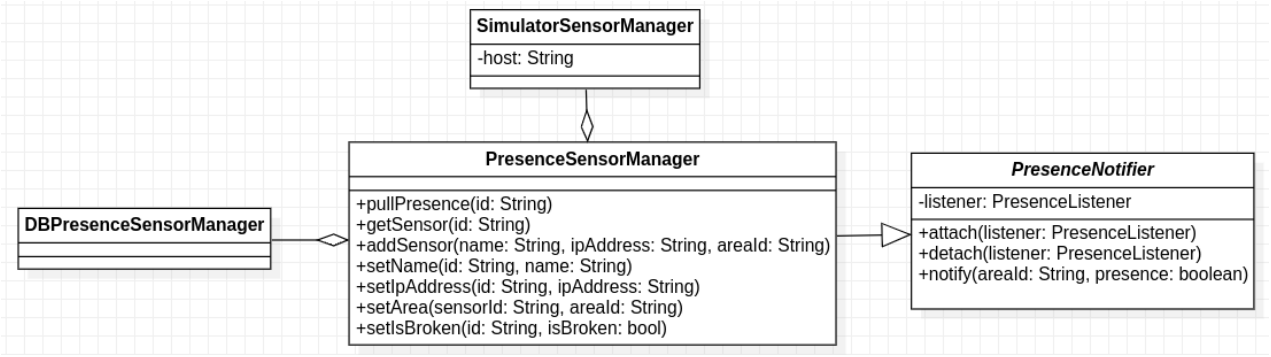
ILocalPresenceSensorManager rappresenta il contratto che devono rispettare le classi che si prospettano di gestire localmente i dati dei sensori di presenza. Nel nostro caso l'interfaccia viene implementata dalla classe *DBPresenceSensorManager* che contiene un'istanza della classe *DatabaseConnection* poiché la gestione locale dei dati avviene per mezzo di un database.

RemotePresenceSensorManager



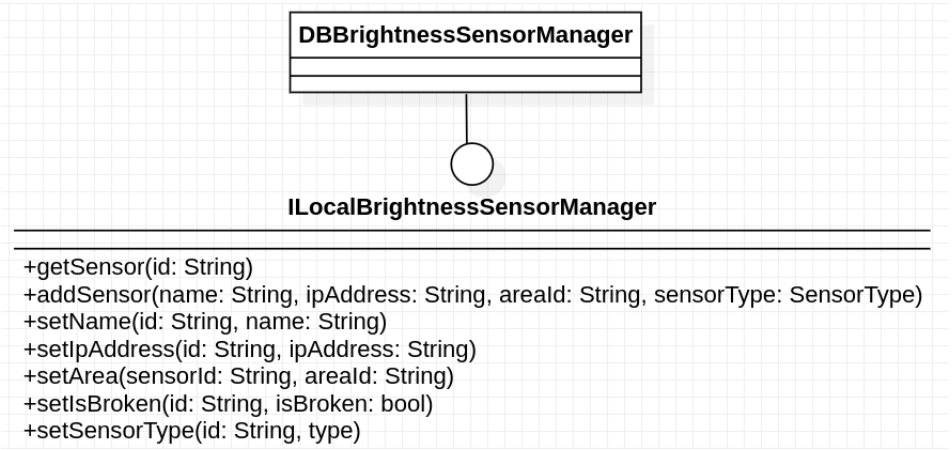
IRemotePresenceSensor è il contratto che devono rispettare le classi che si prospettano di gestire i sensori di presenza in remoto, esse dovranno implementare un metodo per poter leggere se il sensore di presenza ha rilevato oppure no una presenza. Nel nostro caso l'interfaccia viene implementata dalla classe *SimulatorSensorManager* che si occupa di ottenere il valore di presenza tramite chiamate HTTP ad un simulatore.

PresenceSensorManager



PresenceSensorManager offre dei metodi per la gestione generale dei sensori di presenza. Contiene un'istanza di *IRemotePresenceSensor* ed una di *ILocalPresenceSensorManager*. La classe è anche un *PresenceNotifier* in grado di notificare ad eventuali ascoltatori (*listener*) la rilevazione di presenze in una determinata area.

LocalBrightnessSensorManager

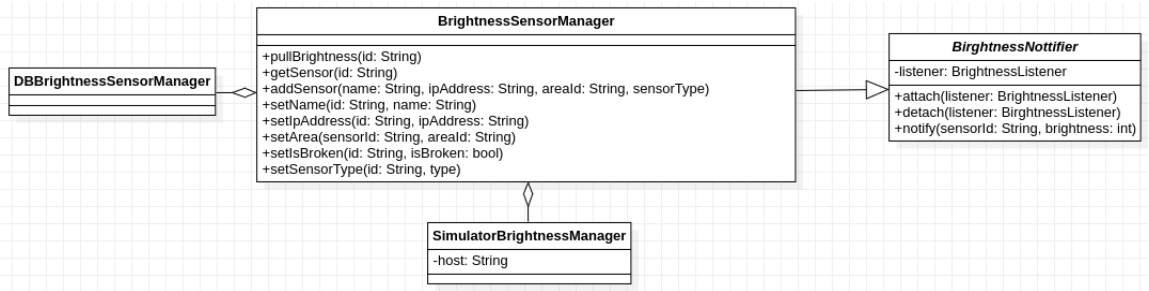


ILocalBrightnessSensorManager si occupa di gestire localmente i dati relativi ai sensori di luminosità. Nel nostro caso l'interfaccia viene implementata dalla classe *DBBrightnessSensorManager* poiché i dati vengono gestite tramite un database.

RemoteBrightnessSensorManager

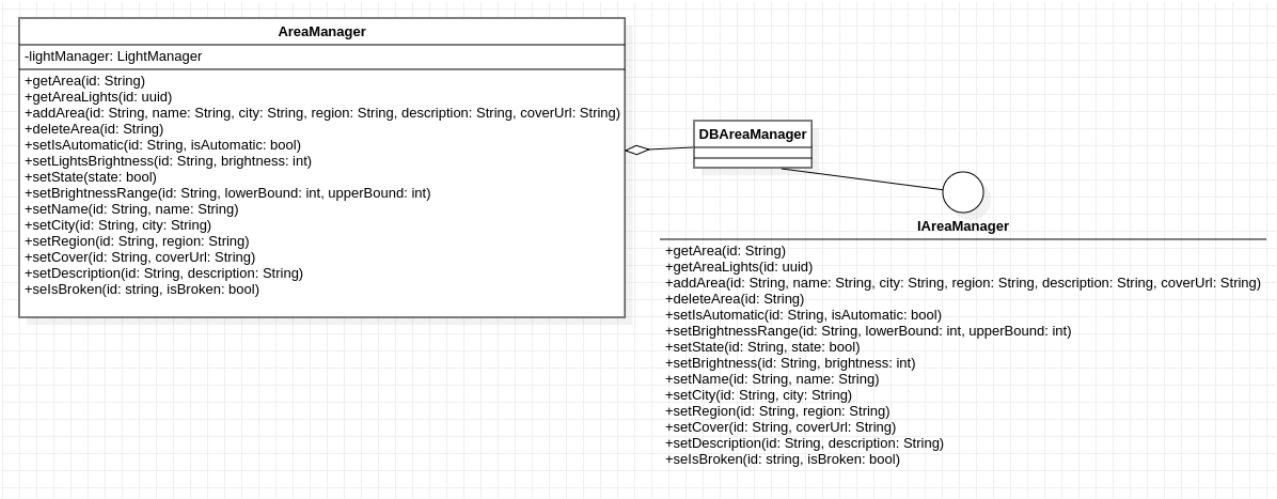
IRemoteBrightnessSensor rappresenta il contratto che deve essere rispettato dalle classi che si prospettano di gestire i sensori d'intensità luminosa remoti; prevede l'implementazione di un metodo che consenta di ottenere l'intensità luminosa rilevata dal sensore. Nel nostro caso l'interfaccia viene implementata dalla classe *SimulatorBrightnessManager* che si occupa di ottenere l'intensità rilevata tramite richieste HTTP ad un simulatore.

BrightnessSensorManager



BrightnessSensorManager offre dei metodi per la gestione generale dei sensori di luminosità. Contiene un'istanza di *IRemoteBrightnessSensor* e una di *ILocalBrightnessSensorManager*. La classe è anche un *BrightnessNotifier* in grado di notificare ad eventuali ascoltatori (*listener*) la luminosità rilevata.

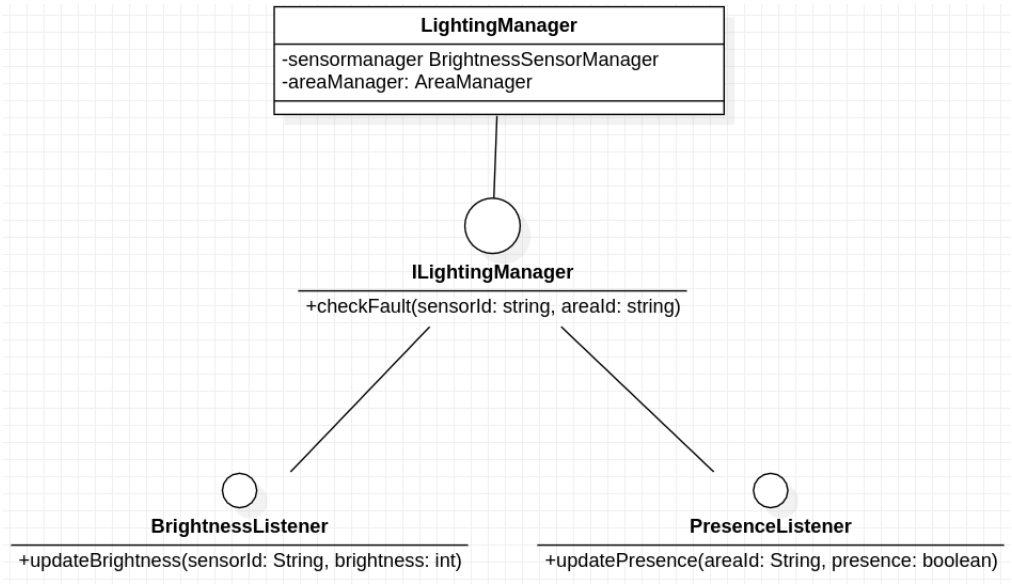
AreaManager



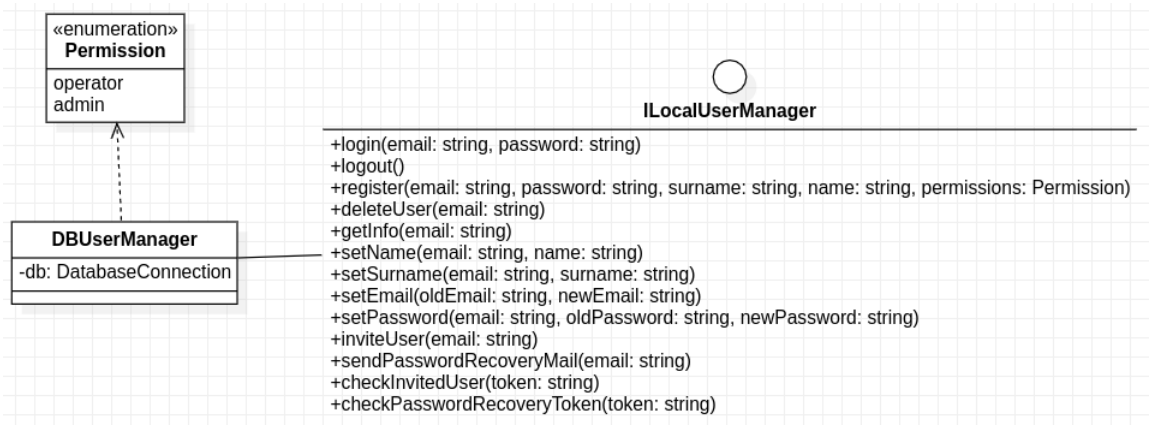
IAreaManager è il contratto che deve essere implementato da ogni classe che si prospetta di gestire i dati dell'area. Nel nostro caso viene implementato dalla classe *DBAreaManager* che contiene un'istanza di *DBConnection*. La classe *AreaManager* contiene un'istanza di *LightManager* necessaria per gestire l'accensione e lo spegnimento dell'intera area ed una di *DBAreaManager*, necessaria per gestire i dati relativi ad ogni area.

LightingManager

ILightingManager rappresenta il contratto che devono implementare tutte le classi che si prospettano di gestire l'accensione delle luci. *LightingManager* si occupa di accendere le luci quando viene rilevata una presenza e quando non c'è abbastanza luce naturale, per questo è un *BrightnessListener* ed un *PresenceListener*. Dopo aver accesso le luci si occupa di verificare che non ci siano guasti, operazione per la quale è necessario prelevare la luminosità, per questo contiene anche un'istanza di *BrightnessSensorManager*. L'istanza di *AreaManager* è necessaria per poter accendere l'area.



UserManager



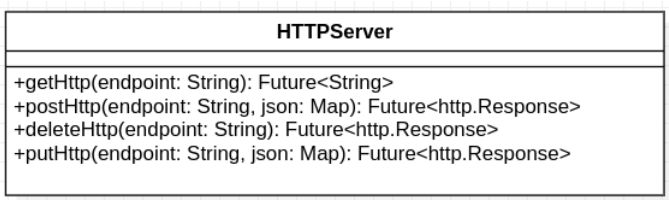
ILocalUserManager è il contratto che deve rispettare una classe che si prospetta di gestire i dati degli utenti ed eventuali verifiche su di essi. Per gestire i dati, nel nostro caso, viene utilizzato un database per cui l'interfaccia viene implementata dalla classe *DBUserManager*.

3.2 Frontend

Per lo sviluppo del frontend si è deciso di utilizzare il design pattern *Model View Controller*.

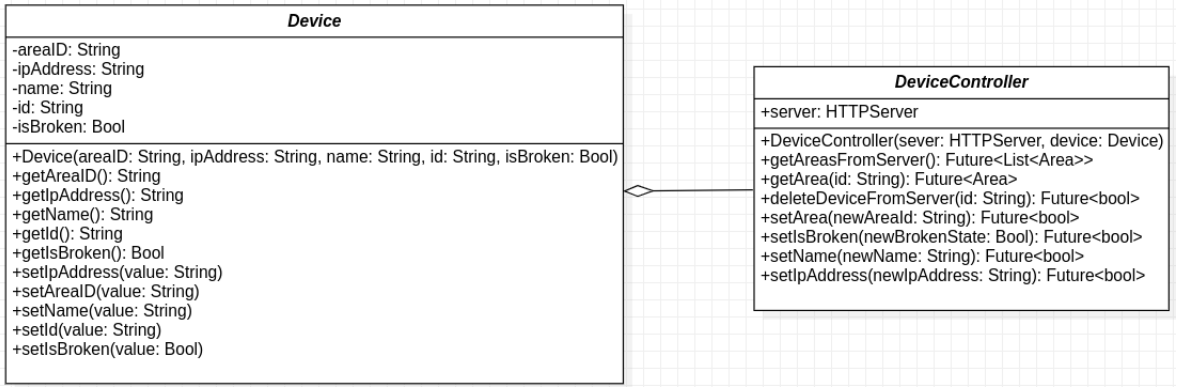
HttpServer

La classe *HttpServer* viene utilizzata dai controller per comunicare con il server, in questo caso la comunicazione avviene tramite chiamate Http alle API del backend.



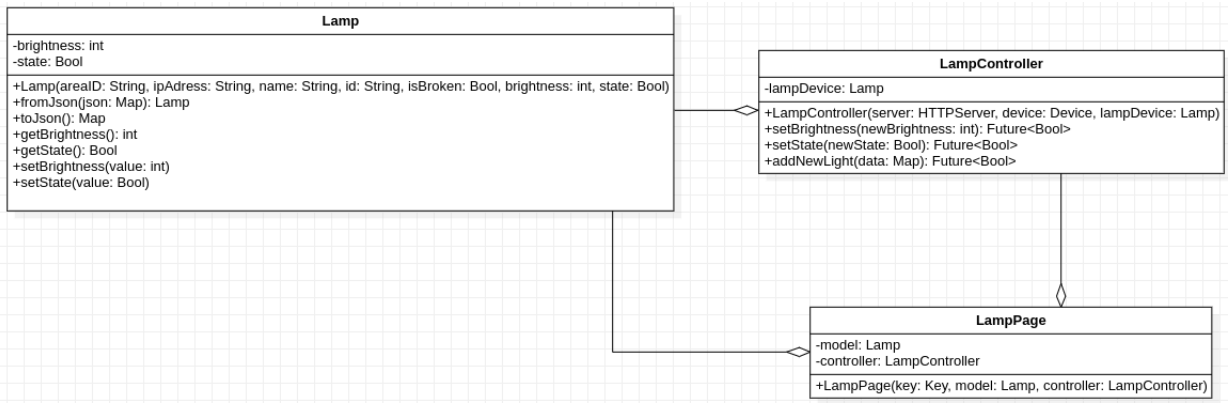
Device

Device è la classe astratta che viene ereditata dai modelli utilizzati per i dispositivi. *DeviceController* è la classe astratta dalla quale ereditano tutti i controller dei dispositivi.



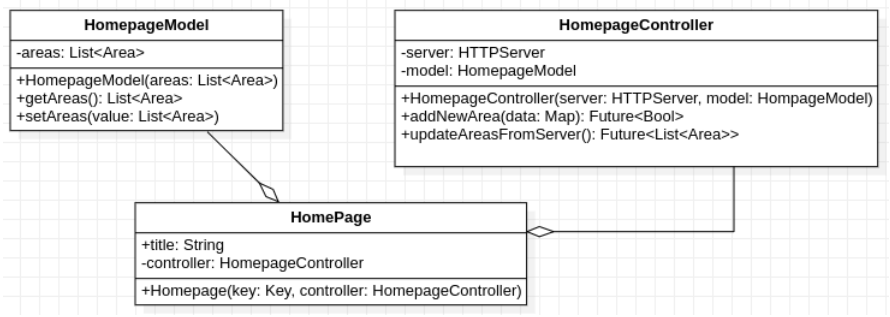
Lamp

LampPage è la *view* per i dispositivi di illuminazione contiene un *LampController* ed un modello *Lamp*.



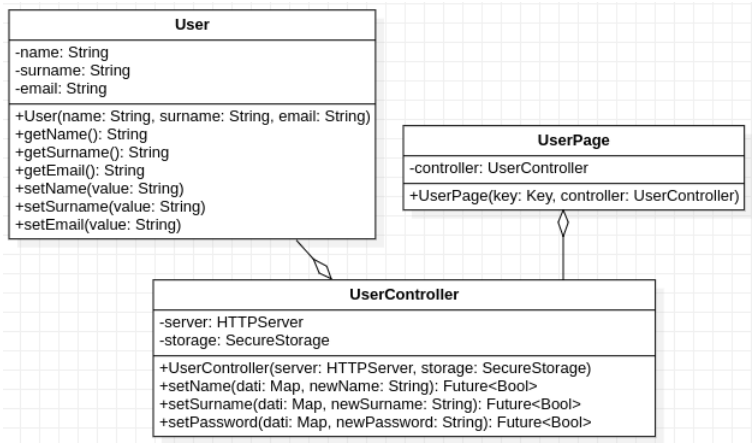
Homepage

La pagina principale della webapp ha lo scopo di mostrare una lista delle aree.



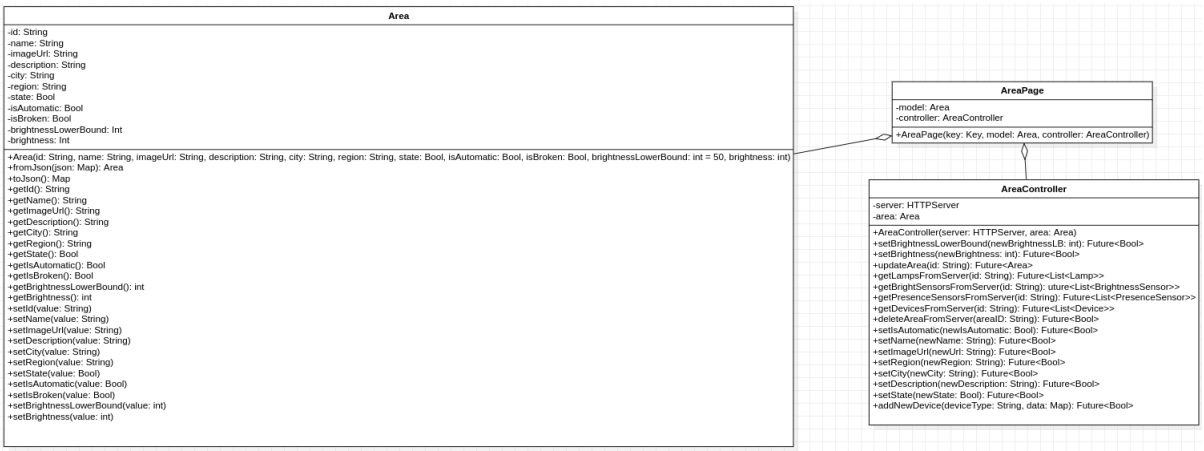
User

La pagina per gestire i dati riguardanti l'utente è *UserPage* che contiene il modello *User* e lo *UserController*.



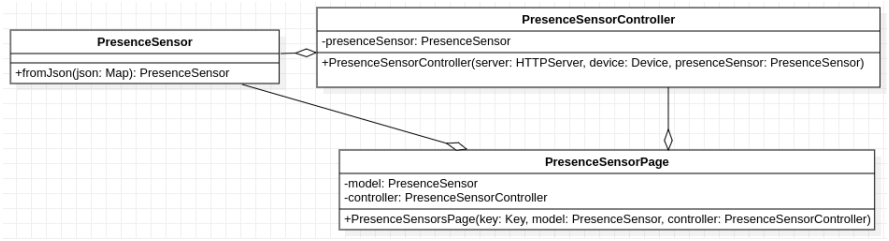
Area

AreaPage è la *view* per le aree che contiene un *AreaController* ed un modello *Area*. Da questa pagina è possibile regolare la modalità dell'area, visualizzare i dati relativi ad un'area e tutti i dispositivi appartenenti all'area.



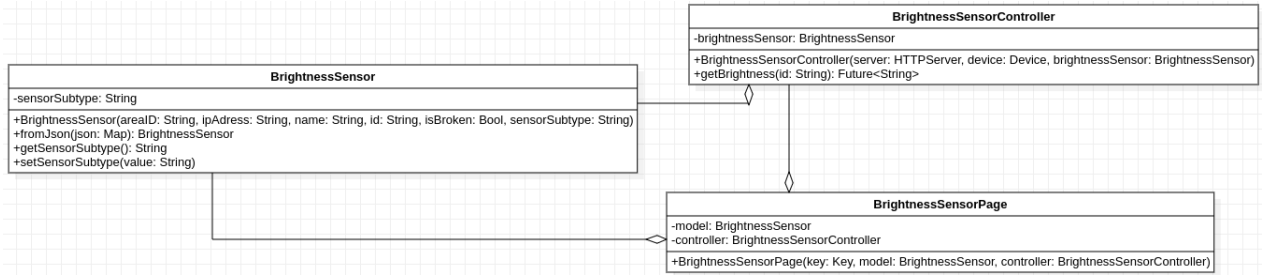
PresenceSensor

Per i sensori di presenza viene utilizzata la *PresenceSensorPage* che contiene un modello *PresenceSensor* e dun controller *PresenceSensorController*.



BrightnessSensor

I sensori di luminosità vengono gestiti nella *BrightnessSensorPage* che contiene il controller *BrightnessSensorController* e il modello *BrightnessSensor*.



4 Requisiti implementati

4.1 Requisiti di funzionalità

Codice	Requisito	Implementato
R1F2	Il gestore deve poter invitare nuovi utenti ad utilizzare il sistema	Sì
R2F2	Il gestore deve poter inserire l'indirizzo e-mail dell'utente che vuole invitare	Sì
R3F2	Un nuovo utente deve poter essere in grado di registrarsi nel sistema	Sì
R4F0	Un nuovo utente deve poter inserire il proprio nome per registrarsi	Sì
R5F0	Un nuovo utente deve poter inserire il proprio cognome per registrarsi	Sì
R6F2	Un nuovo utente deve poter inserire il proprio indirizzo mail per registrarsi	Sì
R7F2	Il sistema deve poter rifiutare la richiesta di registrazione se l'indirizzo mail inserito non rispetta i vincoli imposti	Sì
R8F2	Un nuovo utente deve poter inserire una password per registrarsi	Sì
R9F2	Il sistema deve poter rifiutare la richiesta di registrazione se la password inserita non rispetta i vincoli imposti	Sì
R10F2	Un nuovo utente deve poter reinserire la password per confermarla per registrarsi	Sì
R11F2	Il sistema deve poter rifiutare la richiesta di registrazione se la password e la conferma della password non corrispondono	Sì
R12F2	Il sistema deve poter inviare una email all'utente sconosciuto contenente un codice di conferma dell'indirizzo unico e privato	Sì
R13F2	Un nuovo utente deve poter confermare la propria email inserendo il codice inviatogli sulla casella di posta	Sì
R14F2	Il sistema deve poter rifiutare la richiesta di registrazione se il codice inviato e il codice inserito non corrispondono	Sì
R15F2	Ogni utente registrato deve poter effettuare il login	Sì
R16F2	Ogni utente registrato deve poter inserire il proprio indirizzo email per effettuare il login	Sì
R17F2	Ogni utente registrato deve poter inserire la propria password per effettuare il login	Sì
R18F2	Il sistema deve poter rifiutare la richiesta di autenticazione se la combinazione email e password non è presente nel database	Sì
R19F1	Ogni utente registrato deve poter recuperare la sua password	Sì

R20F1	Ogni utente registrato deve poter inserire il proprio indirizzo email per recuperare la sua password	Sì
R21F1	L'applicazione deve poter permettere all'utente di reimpostare la sua password tramite il link ricevuta dall'utente sulla sua casella email personale	Sì
R22F1	Ogni utente registrato deve poter inserire una nuova password	Sì
R23F1	Il sistema deve poter rifiutare la richiesta di reimpostazione della password se questa non rispetta i vincoli	Sì
R24F1	Ogni utente registrato deve poter reinserire la nuova password per confermarla	Sì
R25F1	Il sistema deve poter rifiutare la richiesta di reimpostazione della password se la nuova password e la conferma della password non corrispondono	Sì
R26F2	Ogni utente deve poter effettuare il logout	Sì
R27F2	Il sistema deve essere in grado di rilevare la presenza di persone all'interno di un'area illuminata	Sì
R28F2	Il sistema deve poter aumentare l'intensità luminosa dell'impianto interessato per un periodo di tempo prefissato	Sì
R29F2	Ogni utente deve poter visualizzare l'elenco delle aree illuminate	Sì
R30F2	Ogni utente deve poter avere la possibilità di creare una nuova area luminosa	Sì
R31F2	Ogni utente deve poter inserire il nome della nuova area che vuole creare	Sì
R32F2	Il sistema deve poter rifiutare la richiesta di creazione di una nuova area se il nome inserito non rispetta i vincoli imposti	Sì
R33F2	Ogni utente deve poter rimuovere un'area illuminata dal sistema	Sì
R34F2	Ogni utente deve poter selezionare l'area da rimuovere	Sì
R35F2	Il sistema deve poter rifiutare la richiesta di rimozione dell'area se questa non rispetta i vincoli imposti	Sì
R36F2	Il sistema deve poter permettere di cambiare la modalità di regolazione dell'intensità luminosa	Sì
R37F2	L'utente deve poter selezionare l'area di cui vuole cambiare la regolazione dell'intensità luminosa	Sì
R38F2	L'utente deve poter impostare la nuova modalità di regolazione dell'intensità luminosa per l'area di interesse	Sì
R39F2	L'utente deve poter regolare manualmente l'intensità di luce emessa selezionando il valore desiderato	Sì
R40F2	L'intensità luminosa degli impianti dell'area deve poter essere cambiata in base al valore inserito	Sì
R41F2	Il sistema deve poter rilevare l'intensità luminosa di una zona tramite il *sensore*$_{G}$	Sì

R42F2	Il sistema deva poter regolare automaticamente l'intensità di luce emessa dall'impianto in base alla luminosità rilevata	Sì
R43F2	Ogni utente deve poter selezionare l'area per la gestione dei dispositivi al suo interno	Sì
R44F2	Ogni utente deve poter visualizzare l'elenco dei dispositivi presenti in un'area specifica	Sì
R45F2	Ogni utente deve poter inserire un nuovo dispositivo nel sistema	Sì
R46F2	L'utente deve poter inserire l'identificativo per il dispositivo da inserire	Sì
R47F2	Il sistema deve poter rifiutare la richiesta di inserimento del nuovo dispositivo se l'identificativo inserito non rispetta i vincoli imposti	Sì
R48F2	L'utente deve poter selezionare la tipologia del nuovo dispositivo	Sì
R49F2	Ogni utente deve poter rimuovere un dispositivo dal sistema	Sì
R50F2	L'utente deve poter selezionare il dispositivo da rimuovere dal sistema	Sì
R51F2	Ogni dispositivo deve poter essere spostato ad un'altra area di illuminazione	Sì
R52F2	L'utente deve poter selezionare i dispositivi da spostare	Sì
R53F2	L'utente deve poter selezionare l'area un cui spostare i dispositivi	Sì
R54F2	Ogni utente deve poter visualizzare l'elenco dei dispositivi guasti	Sì
R55F2	Ogni utente deve poter visualizzare l'elenco delle aree al cui interno sono presenti uno o più dispositivi guasti	Sì
R56F2	I dispositivi guasti devono poter essere segnalati manualmente	Sì
R57F2	L'utente deve poter selezionare il dispositivo guasto	Sì
R58F2	I dispositivi guasti devono poter essere rimossi dall'elenco guasti quando riparati	Sì
R59F2	L'utente deve poter selezionare il dispositivo da rimuovere dall'elenco dei dispositivi guasti	Sì
R60F2	Le aree con dispositivi guasti devono poter essere rimosse dall'elenco delle aree guaste	Sì
R61F2	L'utente deve poter selezionare l'area da rimuovere dall'elenco delle aree guaste	Sì
R62F2	Il sistema deve poter rilevare un guasto tramite il $sensore_G$ di intensità luminosa	Sì
R63F1	Il sistema deve poter comunicare all'utente del guasto rilevato	No
R64F0	Ogni utente deve poter modificare il proprio nome	Sì

R65F0	L'utente deve poter inserire il nuovo nome	Sì
R66F0	Ogni utente deve poter modificare il proprio cognome	Sì
R67F0	L'utente deve poter inserire il nuovo cognome	Sì
R68F1	Ogni utente deve poter modificare il proprio indirizzo email	No
R69F1	L'utente deve poter inserire il nuovo indirizzo email	No
R70F1	Il sistema deve poter rifiutare l'email se è già registrata e notificare l'utente del problema	Sì
R71F1	L'utente deve poter confermare il nuovo indirizzo email entra un determinato intervallo di tempo	No
R72F1	Il sistema deve poter rifiutare la richiesta di modifica l'indirizzo email non è stato confermato	No
R73F1	Ogni utente deve poter modificare la propria password	Sì
R74F1	L'utente deve poter inserire la password precedente per modificarla	No
R75F1	Il sistema deve poter rifiutare la richiesta di modifica se la password non è corretta	No
R76F1	L'utente deve poter inserire la nuova password	Sì
R77F1	Il sistema deve poter rifiutare la richiesta di modifica se la password non rispetta i vincoli imposti	Sì
R78F1	L'utente deve poter reinserire la nuova password per confermarla	Sì
R79F1	Il sistema deve poter rifiutare la richiesta di modifica se le password non corrispondono	Sì
R80F0	Il gestore deve poter modificare l'indirizzo email di un altro utente	No
R81F0	Il gestore deve poter inserire il nuovo indirizzo email del profilo che vuole modificare	No
R82F2	Ogni dispositivo (*impianto di illuminazione*$_G$, *sensore*$_G$ di intensità luminosa, sensore di presenza, sensore di rilevazione guasti) deve essere appartenere ad un'area	Sì
R83F2	È necessario che il server sia dotato di un'interfaccia grafica tramite cui gli utenti possano interagire che consisterà in una web-app	Sì
R84F2	Ogni area ha un nome univoco	Sì
R85F2	Ogni dispositivo ha un codice identificativo univoco	Sì
R86F2	Ogni utente ha un indirizzo email univoco	Sì
R87F1	Il front-end deve essere sviluppato tramite *Flutter*$_G$	Sì
R88F1	Per la parte server verrà usato *Node.js*$_G$	Sì
R89F2	L'operatore deve poter visualizzare informazioni dettagliate su un area di illuminazione	Sì
R90F2	L'operatore deve poter visualizzare informazioni dettagliate su un impianto di illuminazione	Sì

R91F2	L'operatore deve poter regolare la soglia dell'area sotto la quale viene rilevato automaticamente un guasto	Sì

Requisiti obbligatori implementati: 61/61
Requisiti opzionali implementati: 15/22
Requisiti desiderabili implementati: 6/8

4.2 Requisiti di vincolo

Codice	Requisito	Implementato
R1V2	Il sistema dovrà funzionare sul *browser*$_G$ *Chrome* dalla versione più recente (110.0.5481.178)	Sì
R2V2	Il sistema dovrà funzionare sul browser *Firefox* dalla versione più recente (102.8.0esr)	Sì
R3V2	Il sistema dovrà funzionare sul browser *Microsoft Edge* dalla versione più recente (110.0.1587.41)	Sì
R4V2	Il sistema dovrà funzionare sul browser *Safari* dalla versione più recente (5.1.7)	Sì

Requisiti obbligatori implementati: 4/4

4.3 Requisiti di qualità

Codice	Requisito	Implementato
R1Q2	Il sistema dovrà essere sviluppato secondo quanto espresso nel documento *Norme di Progetto*	Sì
R2Q1	Il codice sorgente della piattaforma sarà reperibile su *GitHub*$_G$	Sì
R3Q2	Le funzionalità del sistema utilizzabili dall'utente dovranno essere documentate nel *Manuale utente*	Sì

Requisiti obbligatori implementati: 2/2
Requisiti opzionali implementati: 1/1