Domotizzazione dell'aula

Indice

1 Introduzione

1.1 Informazioni sul progetto

Questo progetto è stato assegnato dal docente Francesco Mussi il 13.02.2019 agli allievi del terzo anno della scuola arti e mestieri Trevano. Mattia Ruberto, Giulio Bosco e Paolo Gübeli sono i membri del gruppo che si occupano del progetto. La scadenza del progetto è fissata per il 17.05.2019.

1.2 Abstract

This project started from the fact that the lighting switches of the school of arts and crafts of Trevano are placed in the classroom in clumsy and uncomfortable places so we decided to implement an application that would allow you to manage lights, curtains and beamer through a website that allows you to adjust the various components of the classroom. In short, it is a domotization of the classrooms that in this way can be controlled remotely.

1.3 Scopo

Lo scopo del progetto è quello di domotizzare le aule della scuola arti e mestieri Trevano. Per domotizzazione delle aule si intende la possibilità di poter gestire le luci, il beamer e le tende delle aule tramite un sito web in cui vengono rappresentate le aule con tutti i valori modificabili. In questo sito web ci sarà un sistema di login da cui potranno accedere solo i docenti.

1.4 Analisi del dominio

Questa applicazione permette di controllare a distanza luci, tende e beamer delle aule delle scuole. Infatti il suo contesto principale di utilizzo sono proprio le scuole dato che è un sistema molto semplice e a basso costo che fa proprio caso alle scuole. Infatti l'idea è di implementare e testare quest'applicazione inizialmente in una singola aula dopdichè se funziona bene si può implementarlo in tutte le aule della scuola. Prodotti simili esistono già ma non specifici per le aule del CPT, inoltre il nostro sistema va ad interagire direttamente con l'Active Directory Domain Service della scuola per controllare se chi fa il login è un docente e quindi se può accedere al pannello di controllo o no. Senza il nostro prodotto gli utenti delle varie aule, docenti e allievi tribulano di continuo con i pulsanti delle luci che funzionano a caso e sono in posizioni difficili da raggiungere.

1.5 Analisi e specifica dei requisiti

Questi sono i requisiti iniziali, dato che all'inizio avevamo programmato di riuscire a fare tutto. Ma durante lo sviluppo del progetto ci siamo accorti che le cose da fare erano tante di cui la maggior parte cose che non avevamo mai visto quindi abbiamo deciso di cambiare le priorità di alcuni requisiti.

Requisiti cambiati:

- Simulazione tende
  - Priorità: 1 => 3
- Modulo luci
  - Priorità: 1 => 3
- Modulo tende
  - Priorità: 1 => 3
- Modulo beamer
  - Priorità: 1 => 3

	ID: REQ-01
Nome:	Pagina di login
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Input username e password

	ID: REQ-02
Nome:	Pagina gestionale aule
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Modificare lo stato delle luci
002:	Modificare lo stato delle tende
003:	Modificare lo stato del beamer

	ID: REQ-03
Nome:	Web Server
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Gestione dalla parte web

	ID: REQ-04
Nome:	Server in Python
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Gestione dall'automatizzazione

	ID: REQ-05
Nome:	Ricavare dati dal database
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Tramite java

	ID: REQ-06
Nome:	Verificare i permessi degli utenti
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Verificare sul database della scuola che l'utente che sta cercando di fare il login sia un docente.

	ID: REQ-07
Nome:	Comunicazione parte web con l'Arduino
Priorità:	1
Versione:	1.0
Note:	

	ID: REQ-08
Nome:	Simulazione luci
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Controllare delle lampadine da 220v tramite i relay

	ID: REQ-09
Nome:	Simulazione tende
Priorità:	3
Versione:	1.0
Note:	
	Sotto requisiti
001:	Controllo dei motori tramite l'Arduino

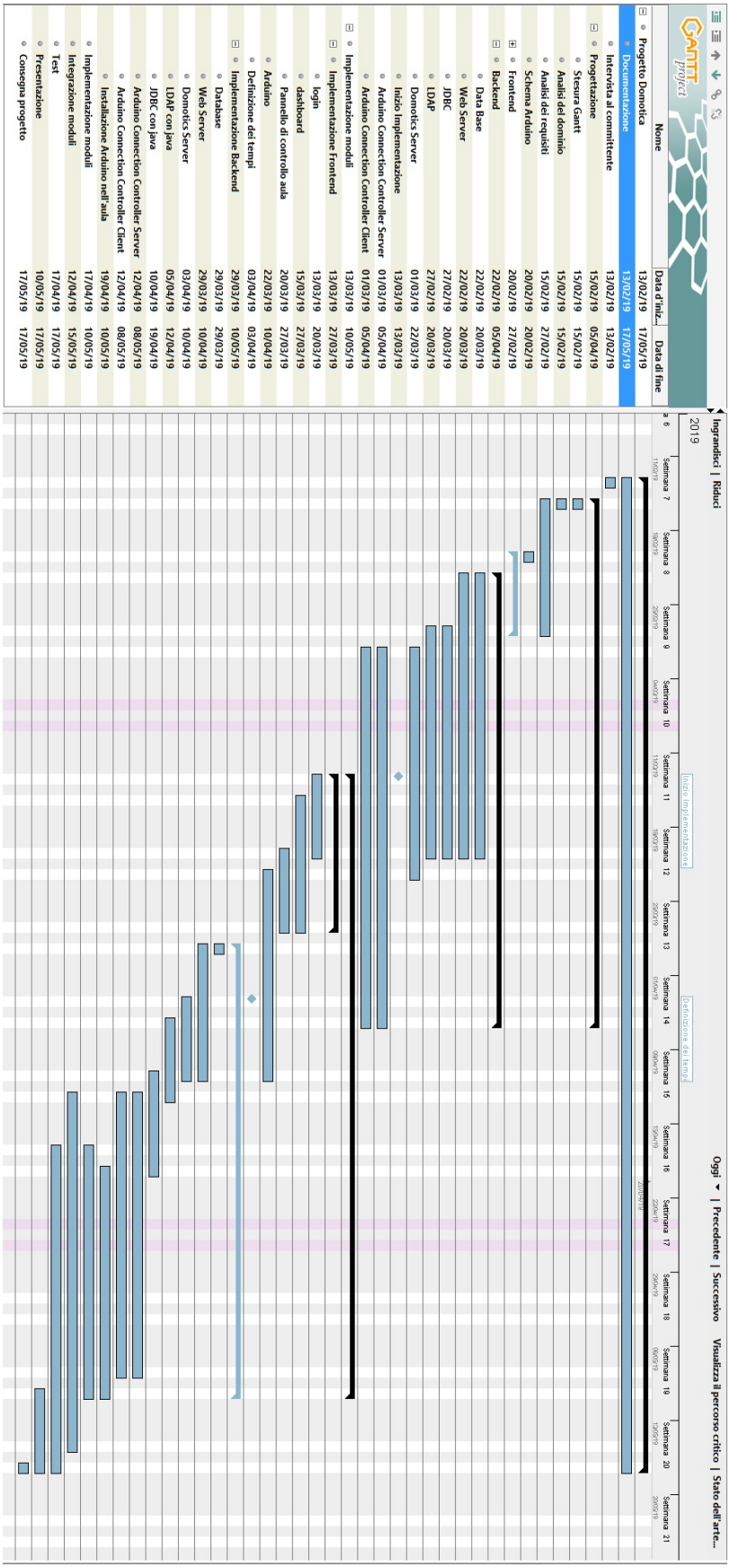
	ID: REQ-10
Nome:	Modulo luci
Priorità:	3
Versione:	1.0
Note:	
	Sotto requisiti
001:	Implementazione modulo fisico delle luci

	ID: REQ-11
Nome:	Modulo tende
Priorità:	3
Versione:	1.0
Note:	
	Sotto requisiti
001:	Implementazione modulo fisico delle tende.

	ID: REQ-12
Nome:	Modulo beamer
Priorità:	1
Versione:	1.0
Note:	
	Sotto requisiti
001:	Implementazione modulo fisico del beamer

	ID: REQ-13
Nome:	Guide di utilizzo
Priorità:	2
Versione:	1.0
Note:	
	Sotto requisiti
001:	Documentazione di tutti i moduli che compongono il progetto

1.6 Pianificazione


GANTT project
Ingrandisci | Riduci
Oggi | Precedente | Successivo | Visualizza il percorso critico | Stato dell'arte...
2019
Inizio implementazione
definizione dei tempi
Progetto Domotica 13/02/19 17/05/19
Documentazione 13/02/19 17/05/19
Intervista al committente 13/02/19 13/02/19
Progettazione 15/02/19 05/04/19
Stesura Gantt 15/02/19 15/02/19
Analisi del dominio 15/02/19 15/02/19
Analisi dei requisiti 15/02/19 27/02/19
Schema Arduino 20/02/19 20/02/19
Frontend 27/02/19 05/04/19
Backend 22/02/19 05/04/19
Data Base 22/02/19 20/03/19
Web Server 22/02/19 20/03/19
JDBC 27/02/19 20/03/19
LDAP 27/02/19 20/03/19
Domotics Server 01/03/19 22/03/19
Inizio Implementazione 13/03/19 13/03/19
Arduino Connection Controller Server 01/03/19 05/04/19
Arduino Connection Controller Client 01/03/19 05/04/19
Implementazione moduli 13/03/19 10/05/19
Arduino Connection Controller Client 13/03/19 10/05/19
Implementazione Frontend 13/03/19 27/03/19
login 13/03/19 20/03/19
dashboard 15/03/19 27/03/19
Pannello di controllo aula 20/03/19 27/03/19
Arduino 22/03/19 10/04/19
Definizione dei tempi 03/04/19 03/04/19
Implementazione Backend 29/03/19 10/05/19
Database 29/03/19 29/03/19
Web Server 29/03/19 10/04/19
Domotics Server 03/04/19 10/04/19
LDAP con java 05/04/19 12/04/19
JDBC con java 10/04/19 19/04/19
Arduino Connection Controller Server 12/04/19 08/05/19
Arduino Connection Controller Client 12/04/19 08/05/19
Installazione Arduino nell'aula 19/04/19 10/05/19
Implementazione moduli 17/04/19 10/05/19
Integrazione moduli 12/04/19 15/05/19
Test 17/04/19 17/05/19
Presentazione 10/05/19 17/05/19
Consegna progetto 17/05/19 17/05/19

Il progetto ci è stato commissionato il 13 Febbraio 2019 programmiamo di completarlo il 17 Maggio 2019. Questo è il gantt che rappresenta la nostra pianificazione iniziale. È strutturato in modo tale che la documentazione viene portata a pari passo con tutto il progetto, poi dal 15 Febbraio al 29 Marzo arriva la progettazione dove viene fatta la stesura del gantt, viene fatta l'analisi del dominio, dei requisiti, e si inizia a fare la progettazione di come sarà l'intera rete, come sarà il design del frontend, il design del backend e come sarà lo schema dell'arduino. Dopo la progettazione incomincia la parte di implementazione che va dal 13 Febbraio al 8 Maggio infatti essendo un gruppo da tre quando qualcuno finirà la progettazione qualcuno potrà già iniziare con l'implementazione dei vari moduli. Nel gantt viene mostrato come inizieremo ad implementare prima il frontend, ci occuperemo dell'arduino ed infine ci concentreremo sull'implementazione del backend. Dopo l'implementazione ci sarà l'integrazione di tutti i moduli dal frontend che dovrà comunicare con il backend che comunicherà a sua volta con l'arduino. Dopodiché verranno effettuati i test ed infine viene fatta la presentazione.

1.7 Analisi dei mezzi

1.7.1 Software

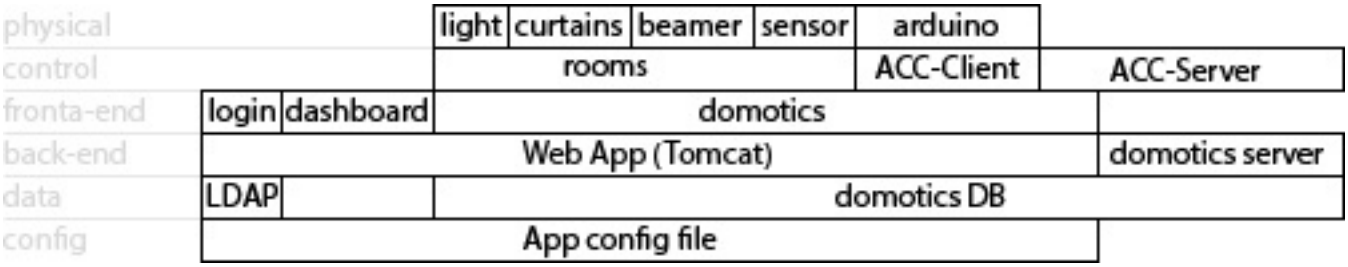
- JetBrains WebStorm 2018.3.5
- JetBrains CLion 2018.3.4
- IntelliJ IDEA 2018.3.4/2019.1
- Visual Studio Code
- Sublime Text 3
- StarUML
- Oracle VM VirtualBox
- Fritzing.exe
- Arduino
- Atom
- GitHub Desktop
- NetBeans IDE 8.2

1.7.2 Hardware

- Mattia Ruberto: Asus ROG Intel(R) Core(TM) i7-7700HQ CPU 2.8GHz RAM 16GB
- Giulio Bosco: Mac Book Pro 2018 i7 CPU 3.1GHz RAM 16GB
- Paolo Gübeli: HP Omen 17 i7-6700HQ CPU 2.6GHz RAM 16GB
- Arduino Yun 7300

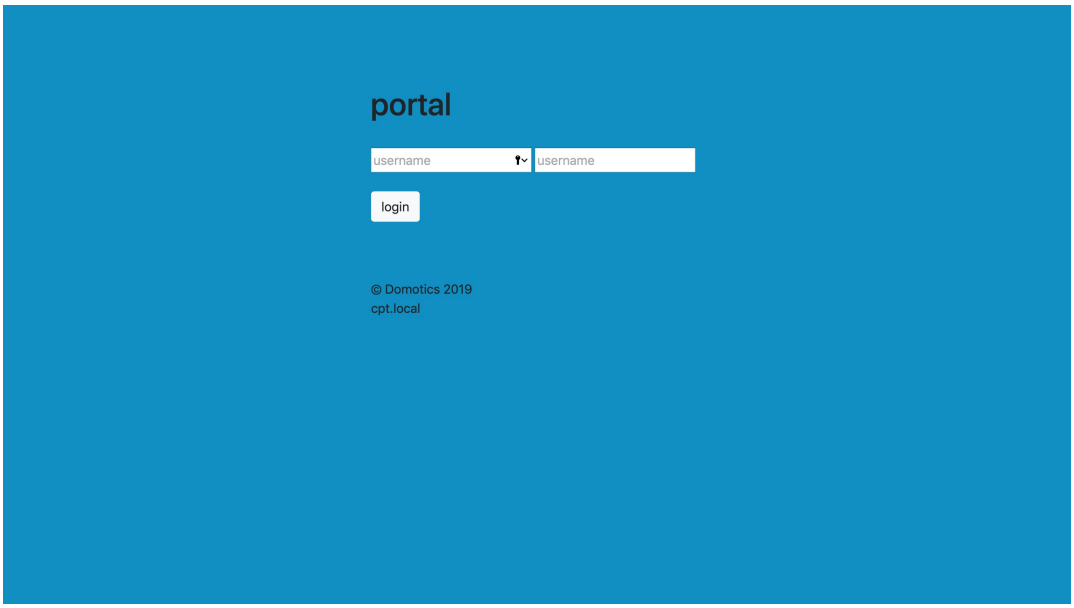
2 Progettazione

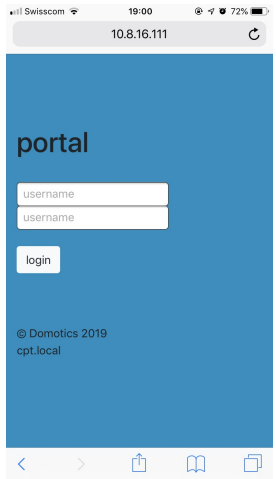
2.1 Architettura progetto



Questo è lo schema che rappresenta il nostro progetto. Il nostro progetto è strutturato nel seguente modo: Nella parte fisica ci saranno i vari componenti dell'aula, come le luci, il beamer, le tende e i vari sensori. Nella parte Control l'arduino viene controllato tramite l'Arduino Connection Controller Server e Client che gli permettono di comunicare con il sito web, da cui si può accedere tramite il login. Una volta che si accede il sito porta l'utente alla dashboard iniziale dove è possibile vedere e gestire tutte le aule presenti. Il sito web è caricato sul web server in Tomcat mentre domotics server si occupa di auto configurare gli arduino presenti sulla rete, di trovarli e di memorizzare porta, indirizzo e chiave che poi salverà sul nostro database. LDAP connector invece gestisce il login, infatti ogni volta che verrà effettuato lui andrà a confrontare le credenziali con quelle del AD-DS della scuola per controllare che l'utente che sta cercando di accedere al sito sia un account di un docente.

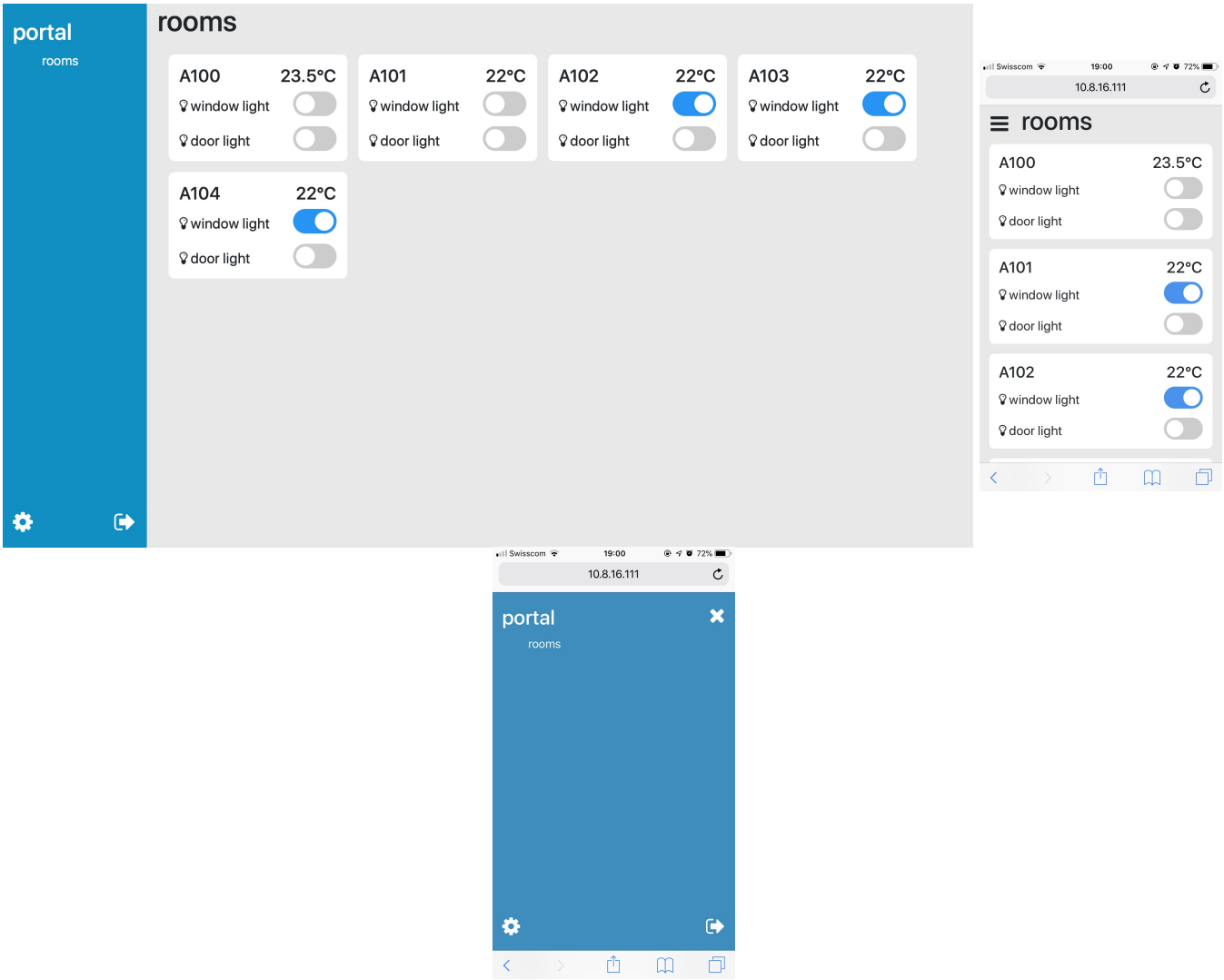
2.2 Design dell'interfaccia di login





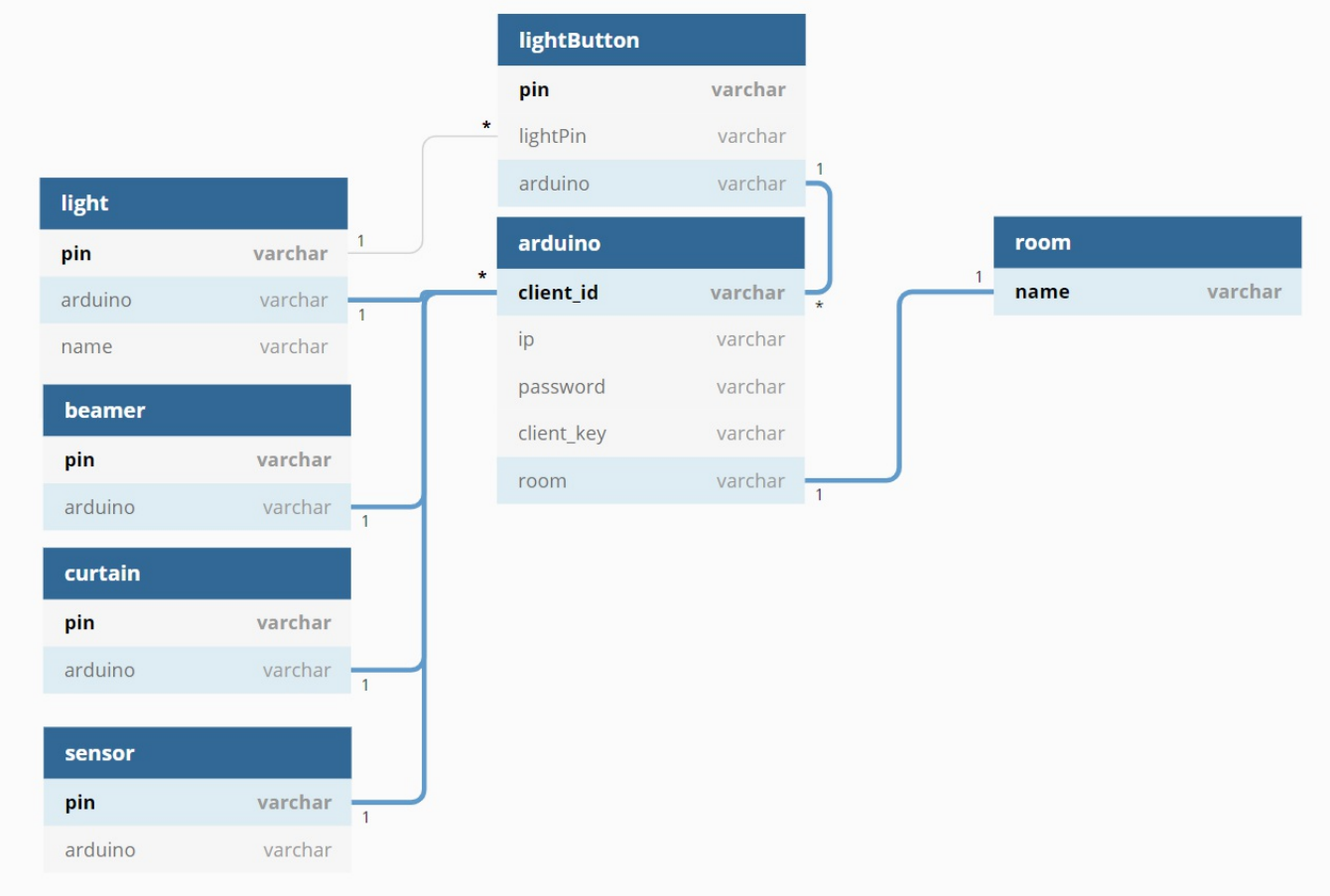
Questo è l'architettura iniziale del login del sito web, l'interfaccia è molto semplice infatti ci sarà un form di login che richiederà nome utente e password.

2.3 Design della dashboard del sito web



Questo è il design del sito web dopo che si ha fatto il login e dopo che le credenziali sono state convalidate e confermate. In questa pagina si posssono visualizzare le varie aule con la possibilità di vedere la temperatura e accendere/spengenere le luci.

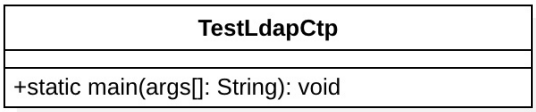
2.4 Architettura Database

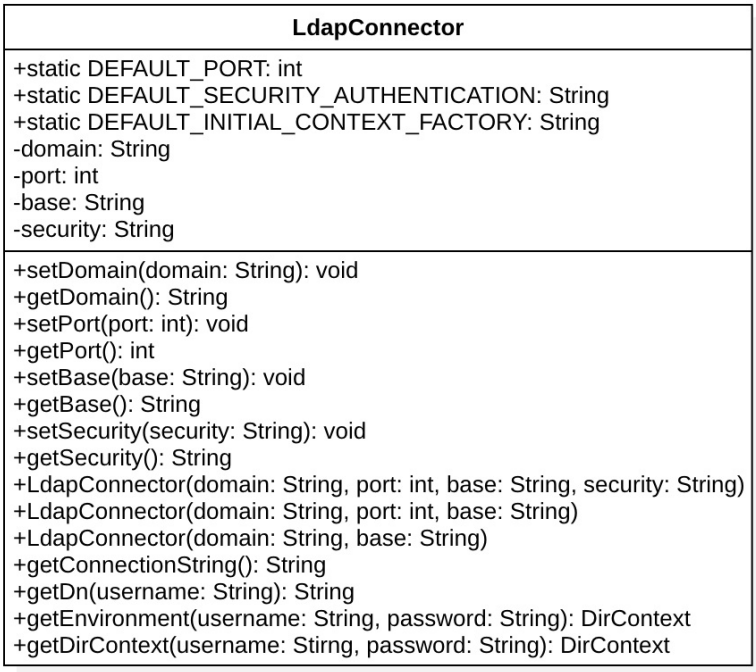


Questo è il design del database, il database è formato da sette tabelle. La tabella room rappresenta le aule di cui viene salvato il loro identificatore, un arduino associato di cui viene memorizzato l'id, l'indirizzo ip, la password e la chiave per il client. Per ogni arduino si gestiscono i bottoni delle luci, di cui vengono memorizzati il suo pin e la luce che controlla. Ogni arduino gestisce le luci, il beamer, le tende e i sensori che ci sono all'interno dell'aula di cui vengono memorizzati i pin. In questo modo grazie alla chiave che ogni arduino ha, il server riesce a riconoscerli e identificarli per poi ricavare lo stato di tutti i suoi moduli, quindi luci, tende, beamer e sensori andando a ricercare le varie informazione nel database.

2.5 Connessione al database della scuola

Per ricavare le credenziali dei docenti abbiamo deciso di utilizzare LDAP Connector perché era la soluzione più facile e sicura da implementare dato che passi le credenziali all sistema AD-DS della scuola, gli dici cosa vuoi sapere e fa tutto lui così nessuno accede direttamente al AD-DS per evitare problemi.



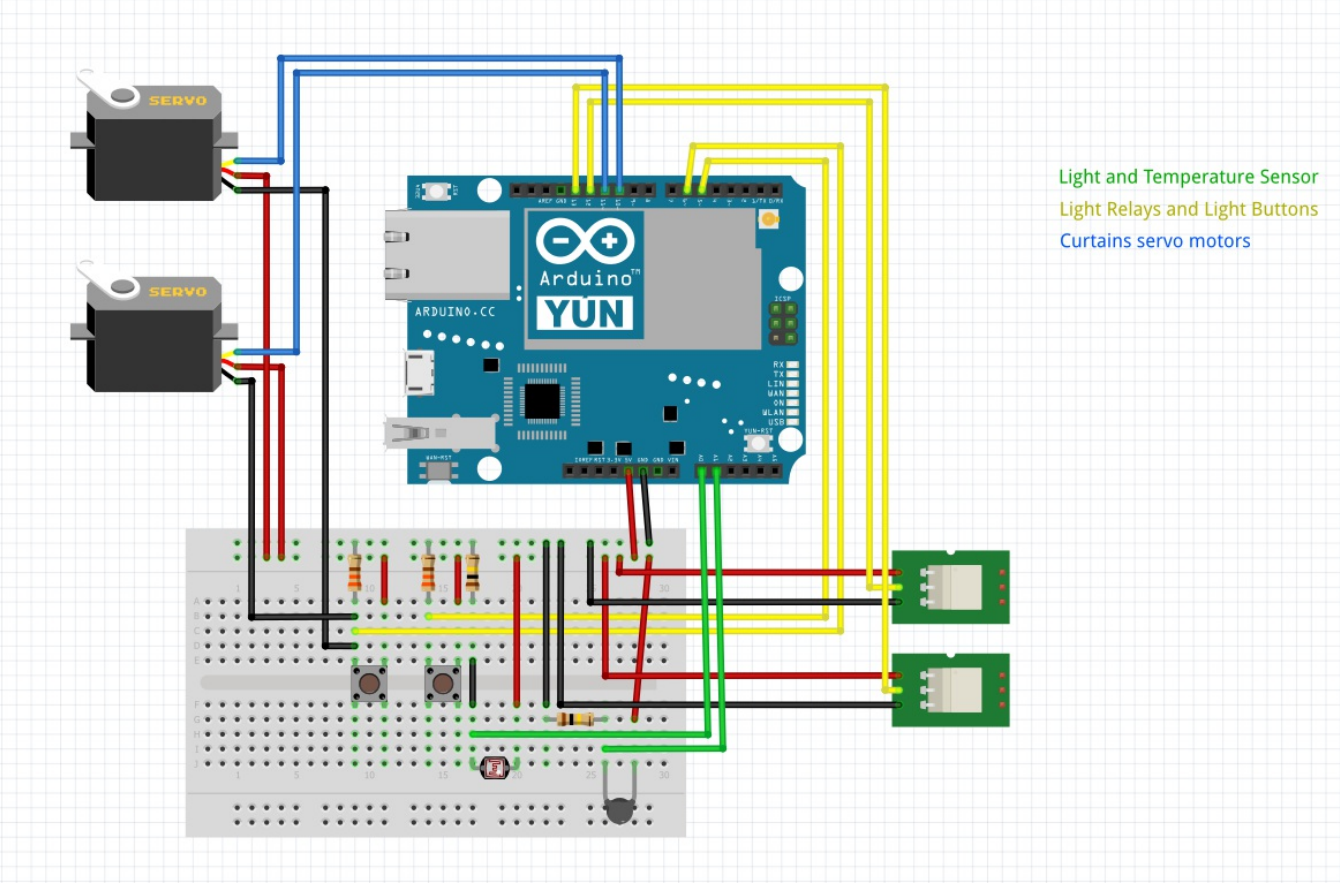


![LDAP](img/ldap/uml.png)

La classe LdapConnector viene utilizzata nel login, infatti quando l'utente si loggerà le credenziali che mette vengono prese e vengono confrontate con le credenziali del database della scuola dove vede se l'utente è un docente e quindi se ha i permessi per accederci o è un allievo e quindi non ha i permessi. LDAP permette di mantenere anche una certa sicurezza essendo che la comunicazione è criptata. Nella classe ci saranno i seguenti attributi statici: la porta di default del server, la chiave di autenticazione, e una variabile che rappresenta il contesto iniziale del LDAP. Poi nelle variabili domain viene salvato il dominio del server LDAP, nella variabile port la porta del server se è diversa da quella di dafault, la variabile base rappresenta il livello del server nelle unità organizzative dove deve andare a controllare le credenziali e security rappresenta il tipo di sicurezza che viene utilizzata per connettersi. Nella classe vengono implementati poi tutti i vari get e set per settare o ritornare i valori delle variabili, ci sono tre costruttori, uno a cui viene passato il dominio, la porta, l'unità organizzativa e il tipo di sicurezza, uno a cui non viene passato il tipo di sicurezza e nel terzo viene passato solo il dominio e l'unità organizzativa. Il metodo getEnvironment ritorna l'ambiente hashtable della connessione, getConnectionString ritorna la stringa di connessione, getDN ritorna una stringa con le credenziali e l'unita organizzativa da inviare nel metodo getEnvironment per creare la connessione, e getDirContext ritorna se l'utente ha i permessi o no.

2.6 Architettura Arduino Yun

All'inizio del progetto ci siamo trovati a dover scegliere quale arduino utilizzare per controllare i vari moduli delle aule, innanzitutto abbiamo deciso di utilizzare la connessione ethernet per comunicare tra interfaccia web e l'arduino, questo per un fattore di comodità infatti per utilizzare una connessione wireless avremmo dovuto usare il fishino che però non abbiamo mai utilizzato e quindi sarebbe potuto essere un problema in più, inoltre tramite connessione ethernet non avremmo avuto problemi di delay e avremmo avuto più stabilità.



Questo è il design dello schema dell'arduino, come si può notare questo schema rappresenta una simulazione di ciò che poi si dovrà implementare fisicamente, infatti i motori delle tende sono stati sostituiti con dei servomotor che rappresenteranno poi i motori che verranno utilizzati per muovere le tende. Le luci invece vengono controllate tramite dei relays. Per le simulazioni si può sostituire i relays con dei LED. Gli interruttori delle luci delle aule vengono simulati con dei bottoni in pull-down. Nello schema c'è anche il sensore di temperatura e il sensore di luce che possono essere usati per dare dati in più sull'aula.

2.7 Comunicazione Arduino Server

Per comunicare tra l'arduino e il server abbiamo deciso di creare un sistema chiamato Arduino Connection Controller (ACC) che verrà utilizzato per comunicare fra gli arduino e il server Domotics, quindi questo modulo sarà diviso in due parti, una sviluppata in Java (ACC-Server) mentre l'altra sviluppata in Arduino (ACC-Client).

In modo tale che quando l'arduino si collegherà ad una rete l'ACC dovrà preoccuparsi di collegarsi alla rete con il DHCP, questo per permettere il corretto funzionamento del sistema in ogni caso. Dopo aver collegato entrambi sulla stessa rete, l'ACC dovrà ricercare il nostro server domotics tramite (ACC-Client-Discover), il quale servirà per registrare l'arduino sul server e per poterlo configurare dal server. Dopo che l'arduino si è collegato al server o se questo è già collegato il server configura l'arduino in modo tale che poi possono iniziare a lavorare. Siccome si

utilizza l'arduino Yun che è suddiviso in due parti arduino e OpenWRT il codice sarà suddiviso in due parti come detto in precedenza:

- arduino: ACC-Client.ino
- python: ACC.Client.py

Protocollo

Il Protocollo ACC per domotics è molto semplice, essendo basato su un codice univoco (ACC-Client-ID) per ogni Arduino, ed una chiave di comunicazione (ACC-Client-KEY) di comunicazione scelta dal server. Il codice univoco sarà salvato in una costante sull'Arduino, e servirà per identificare l'Arduino quando si connette alla rete e mentre ricerca il server. Invece la KEY viene inviata dal server all'Arduino, la quale verrà utilizzata dal server e dall'Arduino come token per avere una comunicazione sicura, verrà inviata assieme a tutti i messaggi che i due si scambieranno.

ACC-Client-ID:

Codice identificativo di un ACC-Client, formato da 12 numeri esadecimali:

```
1234567890ABCD
```

ACC-Client-KEY:

Ogni KEY sarà composta di un codice esadecimale di 12 caratteri, casuali, quali vengono generati sul server semplicemente perchè ha più potenza di calcolo.

Authentication Discover:

È un codice di comunicazione fra il ACC-Client e ACC-Server, viene utilizzato per riconoscere che le informazioni sono autentiche. Questa viene generata dal ACC-Server ed inviata al client al momento della configuazione. Anch'essa è formata da 12 numeri esadecimali:

```
1234567890ABCD
```

Comunicazione ACC-Client -> ACC-Server:

La comunicazione fra i due elementi ACC deve avvenire tramite HTTP, utilizzando il metodo GET, la risposta dovrà essere un file JSON.

Per esempio quando viene cliccato un bottone l'ACC-Client, invia una richiesta al server simile a questa:

```
http://192.168.1.2:8080/acc?key=ABCDEFABCDEF&pin=13&set=1
```

Tutte le richieste destinate all'ACC dovranno essere fatte verso l'IP del server domotics sulla porta `8080`, alla pagina `acc` e inviando la KEY di comunicazione dell'Arduino (AC-Client-KEY) con il parametro `key` con il valore della KEY.

Per le richieste di invio di dati bisogna inserire il parametro `type` a `send`, questo per segnalare che è una richiesta HTTP di invio dati, poi settare il parametro `pin` al numero del pin (nel caso sia un pin analogico inserire un `a` prima del numero) ed infine settare il parametro `value` al valore del pin.

Il server ritornerà una risposta simile alla seguente.

```
{
    "response": "OK",
    "message": "<value>"
}
```

Il parametro `response` della risposta del server, segnala se la richiesta è stata ricevuta correttamente altrimenti ritorna `FAILED` con un messaggio d'errore.

Comunicazione ACC-Server -> ACC-Client:

La comunicazione fra i due elementi avviene anche qui tramite http, le richieste dovranno essere effettuate con il metodo GET e le risposte saranno dei file JSON.

Le richieste potranno essere di due tipi:

- `set`, che serve per inviare dei dati all'arduino, per esempio il valore che deve assumere un pin
- `get`, che serve per richiedere dei dati all'Arduino, per esempio richiedere il valore di un pin

Esempio di richiesta *set*:

```
http://192.168.1.34:18086/acc?key=ABCDEFABCDEF&pin=4&set=1
```

risposta:

```
{
    "response":"OK",
    "message":"<value>"
}
```

Esempio di richiesta *get*:

```
http://192.168.1.34:18086/acc?key=ABCDEFABCDEF&pin=A0
```

risposta:

```
{
    "response":"OK",
    "message":"<value>"
}
```

ACC-AutoConfiguation:

La richiesta deve essere:

```
http://<serverAddress>:<serverPort>/acc?autoconf&id=<ACC-Client-ID>
```

E la risposta sarà

```
{"id":"<ACC-Client-ID>", "key":"<ACC-Client-KEY>", "server_address":"<serverAddress>:
<serverPort>"}
```

Tutte le risposte saranno inviate in formato JSON, questo per facilitare il l'interpretazione da parte del client.

3 Implementazione

3.1 Sito web

WEB-APP

Il front-end di domotics, è stato pensato come una web app costruita su moduli, quindi per facilitare questo metodo di sviluppo è stato deciso di basarlo su AngularJS, un framework, che permette di aggiungere e togliere moduli indipendenti senza andare ad intaccare gli altri. Questo è stato pensato per poter usare questo sistema come base per un portale al quale si possono aggiungere altri moduli.

moduli presenti

Al momento sono presenti solamente due moduli, quelli di base per il l'interazione con il modulo domotics:

- Login: modulo per il login basato su LDAP.
- Rooms: Controllo delle luci di domotics.

AngularJS

La web-app basata su AngularJS, è formata dalla pagina index.html, la quale carica tutte le librerie e i framework utilizzati dal progetto, per esempio AngularJS, jQuery e bootstrap. In oltre carica l'applicazione Angular, configura le routes delle pagine ed inifine carica i controller ed i services.

Per ogni pagina bisogna creare un controller, il quale richieda i services di cui necessita ed inserisca i valori nella view.

Per inizializzare la web app:

```
var app = angular.module('ViewsAPP', ['ngRoute','ngSanitize']);
```

Questa stringa di codice inizializza l'applicazione angular, con il nome `ViewsAPP` e le configura le librerie `ngRoute`, che serve per caricare le giuste routes, richiede al server la giusta view, dato l'url. Mentre `ngSanitize`, serve per stampare del codice html scaricato tramite un service, per esempio all'interno di un file JSON.

Creare un controller

I controller servono, per trasferire i dati dal servizio alla view (e nel caso in cui è necessario eseguire delle operazioni su di essi). Un controller si crea come segue:

```
app.controller('Controller', ['$scope', '$sce', 'Service', function ($scope, $sce, service) {
        service.getFromService().then(function (data) {
                $scope.data = data;
        });
}]);
```

Creando un controller, bisogna inserire il suo nome, poi un array contenente gli elementi che necessita il controller, ed infine la funzione del controller, con i parametri richiesti precedentemente.

Dopo di che eseguire le operazioni che si necessitano nel controller, la variabile `$scope`, viene utilizzata per passare i valori fra i controller e le view.

Creare un service

I service servono per eseguire le richieste al server, queste possono essere per esempio richieste HTTP.

```
app.factory('Service', ['$http', function($http) {
        var service = [];
        var urlBase = "/data/rooms";

        service.getFromService = function () {
                return $http({
                        method: 'GET',
                        url: url
                }).then(function (response){
                        return response.data;
                },function (error){
                        return error;
                });
        };

        return service;
}]);
```

Un service richiede il nome, ed un array, con gli oggetti di angular di cui necessita, quindi per esempio `$http`, che sarebbe la libreria, per eseguire le richieste HTTP. In ogni service solitamente si mette una sola richiesta, che può essere eseguita in modalità diverse, quindi per ogni modalità si crea una funzione per eseguire la richiesta.

Tutte le richieste vanno inserite in un oggetto, il quale verrà poi ritornato.

Views

Per gestire le pagine vengono usate delle views che vengono caricate da `app.js` nel body della pagina `index.html`, queste vengono gestite dal modulo angular `ngRoute`.

```
app.config(function ($routeProvider) {
        // index
        $routeProvider.when('/', {
                templateUrl: 'views/index.html'
        });

        // login
        $routeProvider.when('/login', {
                templateUrl: 'views/login.html'
        });

        // rooms
        $routeProvider.when('/rooms', {
                templateUrl: 'views/rooms.html'
        });

        // settings
        $routeProvider.when('/settings', {
                templateUrl: 'views/settings.html'
        });

        // else
        $routeProvider.otherwise({
                redirectTo: '/'
        });
}).run(function ($rootScope, $route) {
        $rootScope.$route = $route;
});
```

Per configuare `ngRoute`, bisogna congiurarlo con una funziona da inserire nel metodo `config()` dell'app. Al quale viene passato l'oggetto `$routeProvider`, il quale ha un metodo `when()`, questo metodo permette di settare ad uno specifico url (dopo il simbolo `#`, per esempio `localhost#!/test`), una view da caricare. Ed utilizzare anche un metodo `otherwise()`, che viene utilizzato nel caso in cui viene inserito un url non specificato prima.

Infine avviare la web app con le routes configurate precedentemente.

Stile

Per lo stile delle pagine usiamo bootstrap (https://getbootstrap.com) e fontawesome (https://fontawesome.com). Abbiamo usato Fontawesome per aggiungere delle icone al sito, abbiamo usato Fontawesome al posto delle immagini perché le icone che contiene sono dei caratteri che quindi sono più leggeri e più facili da usare delle immagini.
Ecco un esempio di view implementata usando bootstrap e fontawesome.

```
<div class="navbar">
        <i class="fa fa-bars fa-2x"></i>
</div>
<div class="sidebar">
        <div class="col-md-12">

                <h2>portal<i class="pull-right fa fa-times"></i></h2>
                <ul class="a-white">
                        <li><a href="#!/rooms">rooms</a></li>
                </ul>
        </div>
        <div class="bottom col-md-12 a-white">
    <a href="#!/settings"><i class="fa fa-cog fa-2x"></i></a>
    <a href="/Logout"><i class="fa fa-sign-out pull-right fa-2x"></i></a>
  </div>
</div>
<script src="assets/js/scripts/sidebar.js"></script>
```

Fontawesome usa il tag "i" per inserire le icone. Per definire l'icona bisogna inserire nel attributo class: fa che definisce l'uso di fontawesome, fa-[nome icona] che definisce quale icona si vuole usare, e se si vuole la grandezza dell'icona

```
<i class="fa fa-cog fa-2x">
```

3.2 Database

Per realizzare il database abbiamo utilizzato MySQL. Prima di tutto bisogna creare il database tramite il seguente comando.

```
/* create the database */
CREATE DATABASE domotics;
```

In questo pezzo di codice creiamo la tabella che rappresenta l'aula che avrà un singolo attributo che rappresenta il nome dell'aula.

```
/* create the room table */
CREATE TABLE domotics.room (
    name VARCHAR(255) PRIMARY KEY
);
```

Dopo aver creato l'aula creiamo la tabella che rappresenta l'arduino e dovra avere al suo interno l'id del client, l'indirizzo ip, la password della root, la chiave del client e l'aula i cui è posizionato.

```
/* create the arduino table */
CREATE TABLE domotics.arduino (
        client_id VARCHAR (255) PRIMARY KEY,
        ip VARCHAR (255),
  root_password VARCHAR (255),
        client_key VARCHAR (255),
        room VARCHAR (255),

        FOREIGN KEY (room) REFERENCES domotics.room(name)
);
```

Adesso bisogna creare le tabelle che rappresentano i moduli che l'arduino andrà a controllare. I moduli dovranno avere come attributi il numero del pin, l'arduino da cui vengono controllati e il loro stato.

```
/* create the light table */
CREATE TABLE domotics.light (
    pin VARCHAR (255),
        arduino VARCHAR (255),
        status INT (1),
        PRIMARY KEY (pin, arduino),
        FOREIGN KEY (arduino) REFERENCES domotics.arduino(client_id)
);
```

3.3 Lightweight Directory Access Protocol (LDAP)

Per controllare se l'utente che sta provando a fare il login sia veramente un docente e quindi con i permessi per accedere si usa LDAP, che va sul database della scuola e controlla se le credenziali corrispondono a un docente o no.

Per fare questo serve una stringa di connessione che dice a LDAP dove andare a connettersi. Questa stringa deve contenere il dominio e la porta del server ADDS.

```
    private String getConnectionString() {
        return "ldap://" + getDomain() + ":" + getPort();
    }
```

Questo metodo viene utilizzato per creare e poi ritornare DN, è simile a un percorso assoluto solo che invece di scendere l'albero da sinistra scende da destra. Ecco un esempio di DN: CN=john.doe,OU=People,DC=example,DC=com CN è il nome utente, OU è l'unita organizzativa a cui deve puntare (che possono essere più di una) mentre il primo DC rappresenta le componenti del dominio. Per creare il DN prendiamo l'username e lo uniamo con il percorso dell'unità organizzativa che ricaviamo con getBase();

```
    private String getDn(String username) {
        return "CN=" + username + "," + getBase();
    }
```

Il metodo sottostante si occupa di creare una Hashtable che contiene tutti i parametri che poi verranno inviati. I parametri che andranno inviati sono: DEFAULT_INITIAL_CONTEXT_FACTORY, Connessione iniziale predefinita del contesto iniziale di fabbrica. getConnectionString(), stringa di connessione. getDn(username), percorso a cui deve puntare. password, con cui si è tentato di accedere e che deve essere controllata.

```
    private Hashtable<String, String> getEnvironment(String username, String password) {
        Hashtable<String, String> environment = new Hashtable<String, String>();

        environment.put(Context.INITIAL_CONTEXT_FACTORY, DEFAULT_INITIAL_CONTEXT_FACTORY);
        environment.put(Context.PROVIDER_URL, getConnectionString());
        environment.put(Context.SECURITY_AUTHENTICATION, getSecurity());
        environment.put(Context.SECURITY_PRINCIPAL, getDn(username));
        environment.put(Context.SECURITY_CREDENTIALS, password);

        return environment;
    }
```

Quest'ultimo metodo invece utilizza l'Hashtable per collegarsi e controllare se l'utente è presente all'interno dell'unità organizzativa a cui gli è stato detto di andare a controllare. Se l'autenticazione è valida continua altrimenti richiama un eccezione.

```
public DirContext getDirContext(String username, String password) throws NamingException {
        return new InitialDirContext(getEnvironment(username, password));
    }
```

3.4 Implementazione Java DataBase Connectivity (JDBC)

Java DataBase Connectivity viene utilizzato per connettersi tramite java al database e ricavarne le informazione che si vogliono sapere.
La prima cosa da fare è scaricare i driver corretti per MySql, quindi andando sul sito: https://dev.mysql.com/downloads/connector/j/ selezionare il proprio sistema operativo e scaricare i driver della stessa versione di MySQL sul proprio computer (Consigliata 8.0.15).

Dopodichè bisogna realizzare una connection string che permette di connettersi al database.
Nel seguente pezzo di codice realizziamo la stringa di connessione, la stringa è composta da una parte iniziale che rappresenta il protocollo di rete per arrivare a MySQL, poi bisogna passargli l'host, la porta e il nome del database e in più per evitare problemi con il driver è consigliabile anche aggiungere l'orario della propria zona geografica.

```
String connectionString = "jdbc:mysql://" + host + ":" + port + "/" + database + "?";
        connectionString += TIMEZONE_UTC;
```

Quest'altro metodo serve per dire a DriverManager quando dovrà instaurare la connessione quali driver deve utilizzare.

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

Per creare la connessione DriverManager ha bisogno della connectionstring, dell'username e della password. Infine viene ritornata la connessione tramite DriverManager.

```
DriverManager.getConnection(connectionString, username, password);
```

Metodo che crea un istruzione dal collegamento al database, che verrà poi utilizzato per inviare le query al database.

```
this.statement = this.connection.createStatement();
```

Quest'altro metodo invece chiude l'istruzione dal collegamento al database.

```
this.statement.close();
```

Questo metodo serve per fare delle query sul database, gli viene passata la stringa contenente la query, viene creato il collegamento tramite il metodo spiegato in precedenza ed infine tramite executeQuery viene inviata la richiesta al database che verrà poi ritornata al metodo sotto forma di stringa.

```
public ResultSet query(String query) throws SQLException {

    this.createStatement();

    return this.statement.executeQuery(query);
}
```

Nel caso dovessero esserci errori con i driver provare a seguire i seguenti procedimenti:

- Se vi dice che "Loading class com.mysql.jdbc.Driver. This is deprecated." è perché dalla nuova versione la

stringa dentro Class.forName() contiene cj invece nelle vecchie versioni non lo contiene.

- Se vi dice che la zona oraria non è valida basta scrivere dentro a MySQl Workbench indicando a quale fuso orario appartenete:

```
SET @@global.time_zone = '+01:00';
SET @@session.time_zone = '+01:00';
```

3.5 Arduino Connection Controller

È un protocollo che abbiamo ideato per comunicare con facilita con l'Arduino, mentre lo stavamo progettando ci siamo accorti che questo protocollo può essere esteso per qualunque micro controllore che possa essere connesso ad una rete LAN, siccome è basato sul protocollo HTTP, per poterlo utilizzare basta un server HTTP personalizzato ed un client HTTP per eseguire le richieste al server ACC.
Il server HTTP e il client, sono entrambi sia sul server ACC che sul client ACC (Microcontrollore).
Per semplicità il server verrà chiamato `ACC-Server` mentre il lato client verrà denominato `ACC-Client`.

Questo protocollo, come un modulo a se stante, dal progetto `domotics`, quindi potrà venir utilizzato anche da altri progetti.
L'idea del funzionamento di questo modulo, è poter inviare dei valori da settare sui pin, oppure richiedere lo stato dei pin. Questo protocollo deve funzionare in maniera "sicura" se vi è presente un ACC-Server, oppure in maniera autonoma se il suo server non c'è.

La modalità `sicura` utilizza una chiave per scambiare i valori fra l'ACC-Client e l'ACC-server, la chiave è una stringa esadecimale di 12 caratteri. Quando la modalià sicura è abilitata, per richiedere i valori al microcontrollore o settare dei valori sui pin, mentre nella modalità senza l'ACC-Server, chiunque conosce l'indirizzo IP del server ed il funzionamento del protocollo può inviare comandi o richiedere valori all'ACC-Client.

ACC-Server (Arduino Connection Controller - Server)

L'ACC-Server, è composto di un server HTTP ed un elemento per creare le richieste HTTP. Il server HTTP, ha bisogno di una pagina, la quale deve essere in grado di interpretare due richiste:

- autoconf: Questa richiesta richiede tramite il suo ID, la quale ritorna la chiave di comunicazione.
- set: Questa richiesta deve contenere, la chiave di comunicazione, il pin ed il valore, questa serve per aggiornare l'ACC-Server nel caso in cui un pin (per esempio bottone), cambia stato.
  L'ACC-Server, si basa sul un database, che viene utilizzato per risolvere le richieste, per esempio quale input deve accendere o spegnere quale luce.

ACC-Server - autoconf

Il comando autoconf serve per autoconfigurare l'arduino così da permettergli di comunicare con il server. La richiesta deve essere:

```
http://<serverAddress>:<serverPort>/acc?autoconf&id=<ACC-Client-ID>
```

E la risposta sarà

```
{"id":"<ACC-Client-ID>", "key":"<ACC-Client-KEY>", "server_address":"<serverAddress>:
<serverPort>"}
```

Tutte le risposte saranno inviate in formato JSON, questo per facilitare l'interpretazione da parte del client.

ACC-Server - set e set toggle

Quando cambia lo stato di un pin digitale di input sull'arduino, (per esempio la pressione di un bottone) questo deve notificarlo al server, per permettere al server di eseguire le giuste operazioni, per esempio modificare lo stato di altri pin.

In alcuni casi potrebbe essere comodo avere una funzione toggle, per esempio per i bottoni, eseguire la richiesta `set`, la quale semplicemente indica che il pin ha cambiato stato.

ACC-Server - il codice

Nel caso di domotics l'ACC-Server, è stato implementato in java, per poterlo integrare direttamente con il modulo del web e per riutilizzare in questi due ambienti gli stessi modelli dei dati.

Quindi è stata creata una pagina del web server, con le funzionalità dell'ACC-Server (`src/acc/AccServlet.java`), la quale sarà resa disponibile dal webserver all'indirizzo: `http://<serverAddress>:<serverPort>/acc`.

ACC-Client (Arduino Connection Controller - Client)

Anche L'ACC-Client, è composto di un server HTTP ed un client, il server rimane in ascolto sulla porta `8080` , per la ricezione delle richieste dell'ACC-Server, mentre il client esegue le richieste all'ACC-Server, quando cambia lo stato di un bottone (o di un pin digitale di input). Per identificare il client esso contiene un ID formato da 12 numeri esadecimali.

ACC-Client-KEY

È un codice di comunicazione fra il ACC-Client e ACC-Server, viene utilizzato per riconoscere se le informazioni sono autentiche. Questa viene generata dal ACC-Server ed inviata al client al momento della configuazione. Anch'essa è formata da 12 numeri esadecimali.

ACC-Client - alive

Questo comando serve per controllare che l'arduino sia attivo e funzioni correttamente, per il quale eseguire la seguente richiesta:

```
http://<ACC-ClientIP>:<ACC-ClientPort>/alive
```

La relativa risposta sarà:

```
{"status":"OK"}
```

ACC-Client - get e set

Sono due comandi che servono a ricevere o impostare i valori sui pin del arduino. Per impostare i valori si usa la seguente richiesta:

```
http://<ACC-ClientIP>:<ACC-ClientPort>/acc?key=<ACC-Client-KEY>&pin=<pinToSet>&set=<valueToSet>
```

Invece per riceverli si usa questa richiesta:

```
http://<ACC-ClientIP>:<ACC-ClientPort>/acc?key=<ACC-Client-KEY>&pin=<requiredPin>
```

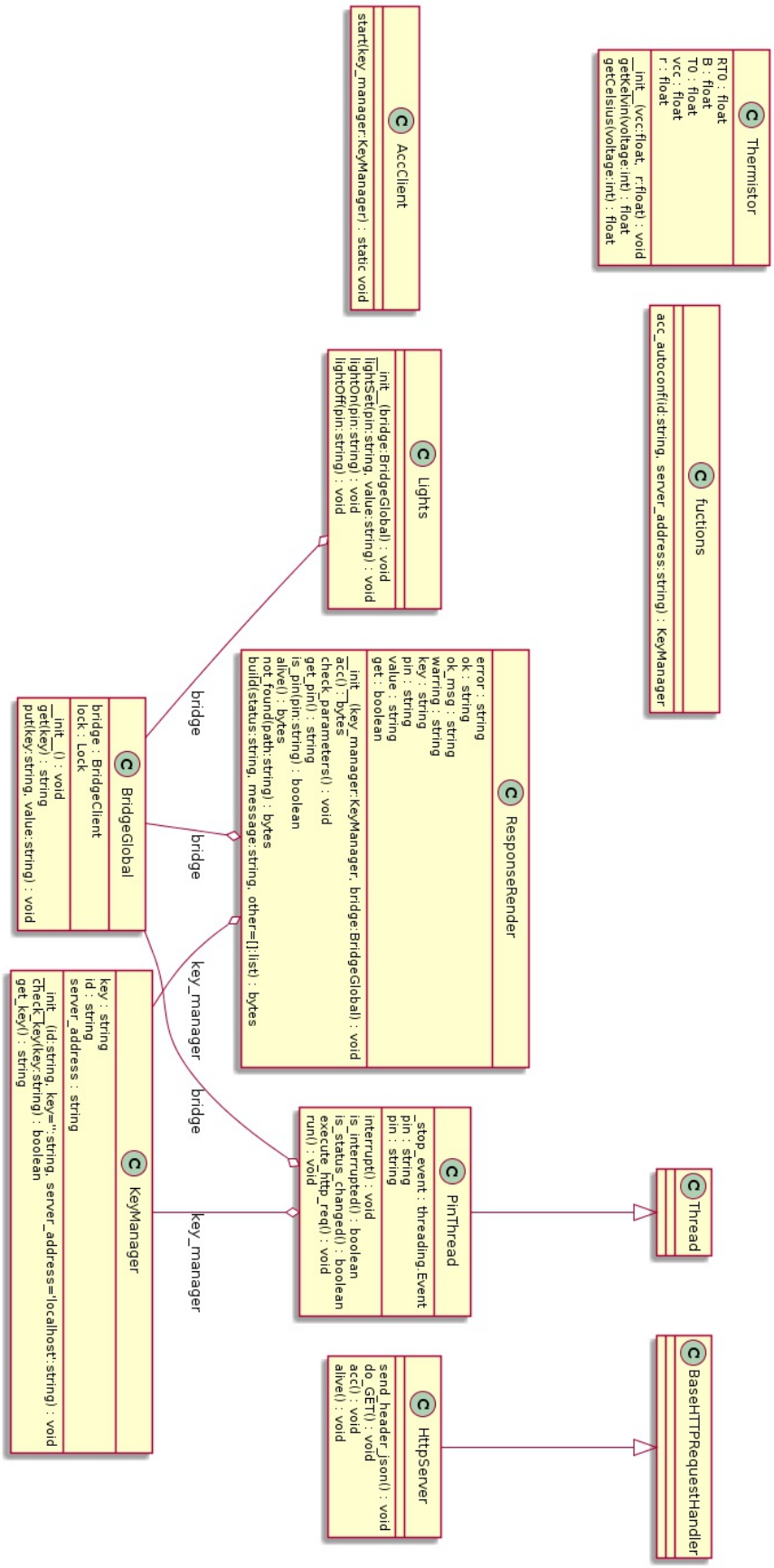
La risposta è in entrambi i casi in questo formato:

```
{"status":"OK","message":"<value>"}
```

ACC-Client - il codice

Nel caso dell'ACC-Client il codice è stato scritto in python, siccome l'Arduino YÚN (Lilino) permette di scrivere il codice in python ed eseguirlo a stretto contatto con il codice sul lato Arduino. Per una spiegazione più accurata del funzionamento dell'Arduino, usare la guida `doc/Documentazione/Guide/arduinoYun.md`.

Principalmente il codice è strutturato in due parti, il codice eseguito sull'Arduino, che semplicemente aggiorna lo stato dei pin di output (prendendo i valori dal Bridge condiviso con la parte Lilino) e aggiorna lo stato del bridge con i pin di input. Codice: `ino/ACC-Client.ino/ACC-Client.ino.ino`.
Mentre il codice relativo a Lilino, si occupa di eseguire il web server, esaudire le richieste dell'ACC-Server e di inviare al server, i cambiamenti dei pin di input, come quelli dei bottoni. Il codice: `py/acc-client.py/`. Il tutto è strutturato come mostrato nell'immagine sottostante, che rappresenta il diagramma delle classi.


Thermistor
RT0 : float
B : float
T0 : float
vcc : float
r : float
__init__(vcc:float, r:float) : void
getKelvin(voltage:int) : float
getCelsius(voltage:int) : float
AccClient
start(key_manager:KeyManager) : static void
fuctions
acc_autoconf(id:string, server_address:string) : KeyManager
Lights
__init__(bridge:BridgeGlobal) : void
lightSet(pin:string, value:string) : void
lightOn(pin:string) : void
lightOff(pin:string) : void
ResponseRender
error : string
ok : string
ok_msg : string
warning : string
key : string
pin : string
value : string
get : boolean
__init__(key_manager:KeyManager, bridge:BridgeGlobal) : void
acc() : bytes
check_parameters() : void
get_pin() : string
is_pin(pin:string) : boolean
alive() : bytes
not_found(path:string) : bytes
build(status:string, message:string, other=[]:list) : bytes
BridgeGlobal
bridge : BridgeClient
lock : Lock
__init__() : void
get(key) : string
put(key:string, value:string) : void
KeyManager
key : string
id : string
server_address : string
__init__(id:string, key='':string, server_address='localhost':string) : void
check_key(key:string) : boolean
get_key() : string
PinThread
_stop_event : threading.Event
pin : string
pin : string
interrupt() : void
is_interrupted() : boolean
is_status_changed() : boolean
execute_http_req() : void
run() : void
Thread
HttpServer
send_header_json() : void
do_GET() : void
acc() : void
alive() : void
BaseHTTPRequestHandler
bridge
bridge
key_manager
bridge
key_manager
key_manager

4 test

4.1 Protocollo di test

Test Case	TC-001
Nome:	Pagina login
Riferimento:	REQ-01
Descrizione:	Verificare che la pagina di login venga rappresentata correttamente.
Prerequisiti:	• Sito web deve essere attivo e funzionante.
Procedura:	• Aprire il sito web.
Risultati attesi	• Deve comparire una pagina di login che richiede username e password.

Test Case	TC-002
Nome:	Pagina dashboard
Riferimento:	REQ-02
Descrizione:	Verificare che una volta eseguito il login vengano rappresentate tutte le aule presenti con tutti i moduli che si possono gestire.
Prerequisiti:	• Sito web deve essere attivo e funzionante.
Procedura:	• Aprire il sito web. • Effettuare il login con i permessi giusti.
Risultati attesi	• Il sito web deve portare alla dashboard di tutte le aule collegate il quale per ognuna è possibile gestire tutti i vari moduli.

Test Case	TC-003
Nome:	Verifica accessi
Riferimento:	REQ-06
Descrizione:	Verificare che il controllo dei dati di accesso avvenga correttamente.
Prerequisiti:	• Sito web deve essere attivo e funzionante.
Procedura:	• Aprire il sito web. • Far effettuare il login ad un docente della scuola. • Far effettuare il login ad un allievo della scuola.
Risultati attesi	• Nel primo caso il login deve far accedere l'utente. • Nel secondo caso il login deve dare errore.

Test Case	TC-004
Nome:	Accendere led
Riferimento:	REQ-08
Descrizione:	Verificare la simulazione delle luci tramite dei led provando ad accenderli tramite il sito.
Prerequisiti:	• Sito web deve essere attivo e funzionante. • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa. • L'arduino deve essere collegato ai led per poterli controllare. • I led devono essere spenti.
Procedura:	• Aprire il sito web. • Effetuare il login con i permessi. • Selezionare l'aula desiderata. • Premere il checkbox della prima luce. • Premere il checkbox della seconda luce.
Risultati attesi	• Entrambi i led devono accendersi

Test Case	TC-005
Nome:	Spegnere led
Riferimento:	REQ-08
Descrizione:	Verificare che la simulazione delle luci tramite dei led provando a spegnerli tramite il sito.
Prerequisiti:	• Sito web deve essere attivo e funzionante. • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa. • L'arduino deve essere collegato ai led per poterli controllare. • I led devono essere accesi.
Procedura:	• Aprire il sito web. • Effetuare il login con i permessi. • Selezionare l'aula desiderata. • Premere il checkbox della prima luce. • Premere il checkbox della seconda luce.
Risultati attesi	• Entrambi i led devono spegnersi.

Test Case	TC-006
Nome:	Rotazione dei motori in senso orario
Riferimento:	REQ-09
Descrizione:	Verificare che la simulazione delle tende funzioni correttamente ruotando i motori da 0° a 360°.
Prerequisiti:	• Sito web deve essere attivo e funzionante. • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa. • L'arduino deve essere collegato ai motorini. • I motori devono a 0°.
Procedura:	• Aprire il sito web. • Effetuare il login con i permessi. • Selezionare l'aula desiderata. • Premere il checkbox delle tende.
Risultati attesi	• Entrambi i motorini devo ruotare di 360° in senso orario.

Test Case	TC-007
Nome:	Rotazione dei motori in senso antiorario
Riferimento:	REQ-09
Descrizione:	Verificare che la simulazione delle tende funzioni correttamente ruotando i motori da 360° a 0°.
Prerequisiti:	• Sito web deve essere attivo e funzionante. • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa. • L'arduino deve essere collegato ai motorini. • I motori devono a 360°.
Procedura:	• Aprire il sito web. • Effetuare il login con i permessi. • Selezionare l'aula desiderata. • Premere il checkbox delle tende.
Risultati attesi	• Entrambi i motorini devo ruotare di 360° in senso antiorario.

Test Case	TC-008
Nome:	Accendere luci
Riferimento:	REQ-10
Descrizione:	Verificare il funzionamento reale delle luci provando ad accenderle tramite il sito.
Prerequisiti:	• Sito web deve essere attivo e funzionante. • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa. • L'arduino deve essere collegato ai relay delle luci dell'aula. • Le luci devono essere spente.
Procedura:	• Aprire il sito web. • Effetuare il login con i permessi. • Selezionare l'aula desiderata. • Premere il checkbox delle tende.
Risultati attesi	• Entrambi le luci dell'aula devono accendersi.

Test Case	TC-009
Nome:	Spegnere luci
Riferimento:	REQ-10
Descrizione:	Verificare il funzionamento delle luci dell'aula provando a spegnerle tramite il sito.
Prerequisiti:	• Sito web deve essere attivo e funzionante. • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa. • L'arduino deve essere collegato ai relay delle luci dell'aula. • Le luci devono essere accese.
Procedura:	• Aprire il sito web. • Effetuare il login con i permessi. • Selezionare l'aula desiderata. • Premere il checkbox della prima luce. • Premere il checkbox della seconda luce.
Risultati attesi	• Entrambi le luci dell'aula devono spegnersi.

Test Case	TC-010
Nome:	Aprire Tende
Riferimento:	REQ-11
Descrizione:	Verificare il funzionamento delle tende dell'aula provando ad aprirle tramite il sito.
Prerequisiti:	• Sito web deve essere funzionante, quindi login, portale per la gestione delle aule • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa • L'arduino deve essere collegato ai motori che muovono le tende. • Le tende devono essere chiuse.
Procedura:	• Aprire il sito web • Effetuare il login • Selezionare l'aula desiderata • Premere il checkbox delle tende.
Risultati attesi	• Le tende devono aprirsi.

Test Case	TC-011
Nome:	Chiudere Tende
Riferimento:	REQ-11
Descrizione:	Verificare il funzionamento delle tende dell'aula provando a chiuderle tramite il sito.
Prerequisiti:	• Sito web deve essere funzionante, quindi login, portale per la gestione delle aule • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa • L'arduino deve essere collegato ai motori che muovono le tende. • Le tende devono essere aperte.
Procedura:	• Aprire il sito web • Effetuare il login • Selezionare l'aula desiderata • Premere il checkbox delle tende.
Risultati attesi	• Le tende devono chiudersi.

Test Case	TC-012
Nome:	Accendere beamer
Riferimento:	REQ-12
Descrizione:	Verificare il funzionamento del beamer dell'aula provando ad accenderlo tramite il sito.
Prerequisiti:	• Sito web deve essere funzionante, quindi login, portale per la gestione delle aule • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa • L'arduino deve essere collegato al beamer che poi lo spegnerà o accenderà • Il beamer deve essere spento.
Procedura:	• Aprire il sito web • Effetuare il login • Selezionare l'aula desiderata • Premere il checkbox del beamer.
Risultati attesi	• Il beamer deve accendersi.

Test Case	TC-013
Nome:	Spegnere beamer
Riferimento:	REQ-12
Descrizione:	Verificare il funzionamento del beamer dell'aula provando a spegnerlo tramite il sito.
Prerequisiti:	• Sito web deve essere funzionante, quindi login, portale per la gestione delle aule • Tutta la parte di comunicazione tra sito web, server e arduino deve funzionare anch'essa • L'arduino deve essere collegato al beamer che poi lo accenderà o spegnerà • Il beamer deve essere acceso.
Procedura:	• Aprire il sito web • Effetuare il login • Selezionare l'aula desiderata • Premere il checkbox del beamer.
Risultati attesi	• Il beamer deve spegnersi.

4.2 Risultati Test

Test Case	Nome	Descrizione	Risultato
TC-001	Pagina login	Verificare che la pagina di login venga rappresentata correttamente.	OK
TC-002	Pagina dashboard	Verificare che una volta eseguito il login vengano rappresentate tutte le aule presenti con tutti i moduli che si possono gestire.	OK
TC-003	Verifica accessi	Verificare che il controllo dei dati di accesso avvenga correttamente.	OK
TC-004	Accendere led	Verificare che la simulazione delle luci tramite dei led provando ad accenderli tramite il sito.	OK
TC-005	Spegnere led	Verificare che la simulazione delle luci tramite dei led provando a spegnerli tramite il sito.	OK
TC-006	Rotazione dei motori in senso orario	Verificare che la simulazione delle tende funzioni correttamente ruotando i motori da 0° a 360°.	FAILED
TC-007	Rotazione dei motori in senso antiorario	Verificare che la simulazione delle tende funzioni correttamente ruotando i motori da 360° a 0°.	FAILED
TC-008	Accendere luci	Verificare il funzionamento reale delle luci provando ad accenderle tramite il sito.	FAILED
TC-009	Spegnere luci	Verificare il funzionamento delle luci dell'aula provando a spegnerle tramite il sito.	FAILED
TC-010	Aprire tende	Verificare il funzionamento delle tende dell'aula provando ad aprirle tramite il sito.	FAILED
TC-011	Chiudere tende	Verificare il funzionamento delle tende dell'aula provando a chiuderle tramite il sito.	FAILED
TC-012	Accendere beamer	Verificare il funzionamento del beamer dell'aula provando ad accenderlo tramite il sito.	FAILED
TC-013	Spegnere beamer	Verificare il funzionamento del beamer dell'aula provando a spegnerlo tramite il sito.	FAILED

4.3 Mancanze/limitazioni conosciute

Queste sono le mancanze che conosciamo del nostro progetto:

- Curtain Managment, aggiungere l'apertura e la chiusura delle tende nell'aula (anche in modo automatico in base alla luce).
- Beamer Managment, aggiungere l'accensione e lo spegnimento del beamer nell'aula.
- ACC-Client Full-AutoConfiguation, aggiungere all'ACC-Client la funzionalità di ricerca di un ACC-Server nella rete.
- Aggiungere lo stato di ogni arduino in una pagina
- Poter inviare dei comandi di reboot agli arduino direttamente dall'interfaccia web.
- Il server non funziona.

5 Consuntivo

I cambiamenti più grandi avvenuti dalla pianificazione iniziale sono nei tempi dell'implementazione. Per iniziare abbiamo avuto 5 giorni in più per consegnare il progetto quindi al posto del 17.05.2019 abbiamo consegnato il progetto il 22.05.2019. Poi uno dei più grandi cambiamenti è nell'implementazione della parte Database/JDBC che hanno richiesto molto meno tempo del previsto. Un'altro cambiamento abbastanza grande si vede nella parte finale dell'implementazione dove si può notare che i test prendono meno tempo del previsto, questo cambiamento è causato principalmente dal fatto che una parte dei requisiti non è stata implementata quindi alcuni test non erano necessari. Per esempio abbiamo rimosso l'attività "installazione Arduino nell'aula" perché non abbiamo avuto il tempo di farla. Anche per integrare i moduli ci abbiamo messo meno del tempo visto che i moduli sono minori di quello che abbiamo previsto nella progettazione.

6 Conclusioni

6.1 Considerazioni finali

Il progetto è stato molto interessante anche se impegnativo, ci è stato assegnato un progetto molto utile e funzionale che negli anni futuri aumenterà sempre di più in tutte le case, uffici e industrie. Grazie a questo progetto è stato possibile entrare nel mondo della domotica che necessita conoscenze varie nell'ambito dell'informatica. Durante il progetto abbiamo dovuto studiare e progettare molte componenti da dover collegare assieme, dato che è suddiviso in molte parti differenti. Il quale ci hanno permesso di fare esperienza con un progetto di dimensioni maggiori a quelle di cui siamo abbituati. Per tutti è stata la prima volta a fare un progetto intero in un gruppo da tre, all'inizio l'organizzazione è stata un po' difficoltosa dato che nessuno di noi era abituato al lavoro di gruppo ma nell'ultimo periodo siamo migliorati e la velocità di lavoro è aumentata. La sfida più grande di questo progetto, è stata svilupparlo interamente modulare, per poter riciclare il codice. Purtroppo durante il progetto abbiamo avuto svariati imprevisti che non ci hanno permesso di portare a termine il lavoro. Quindi durante il corso del lavoro, abbiamo cambiato le priorità di certi requisiti. Per questo abbiamo documentato tutto ciò che abbiamo fatto passo passo in modo tale che la prossima persona o gruppo che ci lavorerà possa capire in fretta come funziona il nostro progetto e quindi possa completarlo e migliorarlo.

6.2 Sviluppi futuri

Nuove funzionalità

- Poter gestire le assenze sia da parte dei docenti sia da parte degli allievi tramite sito della domotica presente.
- Poter gestire l'apertura e la chiusura della rete tramite sito della domotica presente.

Moduli Arduino

- Curtain Managment, aggiungere l'apertura e la chiusura delle tende nell'aula.
- Beamer Managment, aggiungere l'accensione e lo spegnimento del beamer nell'aula.

ACC

- ACC-Client Full-AutoConfiguation, aggiungere all'ACC-Client la funzionalità di ricerca di un ACC-Server nella rete.

WEB

- aggiungere lo stato di ogni arduino in una pagina
- poter inviare dei comandi di reboot agli arduino direttamente dall'interfaccia web.
- modifica (aggiungere/togliere/modificare) delle luci, degli arduino, e dei bottoni (interazione con il db).

7 Bibliografia

7.1 Sitografia

- https://stackoverflow.com/
- https://www.w3schools.com/
- https://getbootstrap.com/docs/4.3/getting-started/introduction/
- http://www-db.deis.unibo.it/courses/BDPG/JDBC.pdf
- https://www.html.it/pag/15159/jdbc-introduzione/
- https://www.theserverside.com/definition/Java-Database-Connectivity-JDBC
- https://searchmobilecomputing.techtarget.com/definition/LDAP
- https://www.gracion.com/server/whatldap.html
- https://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm
- https://www.arduino.cc/
- https://www.instagram.com/awsmcolor/?hl=it

8 Allegati

- Quaderno dei compiti
- Diari di lavoro
- Guida ACC
- Guida ArduinoYun
- Guida Back-end
- Guida Front-end
- Guida database
- Guida Ldap
- Guida Tomcat CantOS7