

Gestione Entrate Uscite Stabile

Titolo del progetto: Gestione Entrate Uscite Stabile
Alunno/a: Matteo Arena
Classe: I4AA
Anno scolastico: 2020/2021
Docente responsabile: Guido Montalbetti

1	Introduzione	4
1.1	Informazioni sul progetto	4
1.2	Abstract	4
1.3	Scopo	4
2	Analisi	5
2.1	Analisi del dominio	5
2.2	Analisi e specifica dei requisiti	5
2.3	Use case	9
2.4	Pianificazione	10
2.5	Analisi dei mezzi.....	12
2.5.1	Software	12
2.5.2	Hardware.....	12
3	Progettazione	13
3.1	Design dell'architettura del sistema	13
3.1.1	Posizionamento dei sensori ad infrarossi	13
3.1.2	Struttura dell'applicativo WEB.....	16
3.2	Design dei dati e database.....	17
3.2.1	Database	17
3.2.2	Struttura LDAP	18
3.3	Design delle interfacce	19
4	Implementazione	22
4.1	Montaggio Hardware	22
4.2	Codice Arduino	24
4.3	Struttura base	27
4.4	Models	27
4.4.1	Database	27
4.4.2	Admin	28
4.4.3	Building	29
4.4.4	Classroom	29
4.4.5	DataChecker	30
4.4.6	GAGI	31
4.4.7	Log	32
4.4.8	User.....	32
4.5	Controllers	35
4.5.1	AdminLogin	35
4.5.2	Login	36
4.5.3	Main	37
4.5.4	AdminController	38
4.5.5	teacherControler	40
4.6	Views	40
4.6.1	Header	40
4.6.2	Footer.....	41
4.6.3	Admin	41
4.6.4	Management	43
4.6.5	Teachers	45
4.6.6	Helpers.....	45
4.6.7	Home.....	45
4.6.8	Login	45
4.6.9	User.....	46
4.7	Funzionamento.....	47
4.7.1	Login	47
4.7.2	Permessi degli utenti.....	48
4.7.3	Log (solo admin)	49
4.7.4	Gestione amministratori (solo admin)	51
4.7.5	Gestione aule (solo amministratori)	52
4.7.6	Gestione stabili (solo amministratori).....	53
4.7.7	Statistiche giornaliere (amministratori, direzione e segretariato).....	55
4.7.8	Statistiche mensili (amministratori, direzione e segretariato)	56

4.7.9	Ricerca (amministratori, direzione, segretariato e docenti)	56
4.7.10	Informazioni utente	57
5	Test.....	58
5.1	Protocollo di test.....	58
5.2	Risultati test.....	62
5.3	Mancanze/limitazioni conosciute.....	66
6	Consuntivo.....	67
7	Conclusioni	68
7.1	Sviluppi futuri.....	68
7.2	Considerazioni personali.....	68
8	Bibliografia	68
8.1	Sitografia	68
9	Glossario	69

1 Introduzione

1.1 Informazioni sul progetto

1.2 Abstract

Nowadays there is an automation of practically all actions that take place during the day, from turning on the lights to even cars that drive themselves. In spite of this, in thousands of years of schooling the counting of students has never changed, in fact until this project the teacher had to count all the students to check that they were present in class. To make this task easier, the following application was developed, Gestione Entrate Uscite Stabile, which allows you to count in a completely automatic way the people who enter a classroom. In this way, teachers will have more time to dedicate to the lesson, without wasting useless time counting the students present.

1.3 Scopo

Lo scopo didattico del progetto è riuscire a gestire nel modo più ottimale possibile un progetto IT con le risorse datoci dalla scuola. Questo servirà successivamente a prepararmi all'esame pratico di fine scuola.

Lo scopo operativo invece riguarda il creare un applicativo WEB dove si possano visualizzare tutte le entrate e le uscite da un'aula all'interno della scuola. Le entrate e le uscite dovranno essere archiviate in modo che non vengano visualizzate nelle statistiche e gli amministratori dovranno avere accesso completo a tutta la gestione dell'applicativo. Gli utenti che potranno accedere saranno unicamente gli amministratori (salvati in un database), i docenti, la segreteria e la direzione. Gli ultimi tre utenti dovranno essere recuperati dal server AD della scuola.

Il progetto dovrà essere facilmente espandibile, per questo sarà presente un manuale di installazione per permettere a chiunque di poter applicarlo in qualsiasi aula interna ad uno stabile.

2 Analisi

2.1 Analisi del dominio

L'applicativo verrà utilizzato principalmente dalla direzione, dalla segreteria e dai docenti della scuola. Principalmente si vuole semplificare il controllo delle assenze da parte degli allievi senza dover contare manualmente ogni allievo singolarmente. In questa maniera il docente in questione potrà visualizzare il numero di persone che sono entrate in un'aula nella maniera più veloce possibile potendo subito dedicarsi alla lezione (senza quindi perdere tempo nel conteggio degli allievi).

2.2 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Accesso
Priorità	1
Versione	1.0
Note	Solo l'amministratore, la direzione e la segreteria possono accedere al sito web
Sotto requisiti	
001	Si necessita la struttura LDAP della scuola

ID: REQ-02	
Nome	LDAP
Priorità	1
Versione	1.0
Note	Bisogna utilizzare la struttura LDAP della scuola per controllare l'accesso degli utenti
Sotto requisiti	
001	Conoscere il nome del server AD della scuola

ID: REQ-03	
Nome	Accesso amministratori
Priorità	1
Versione	1.0
Note	L'amministratore locale ha l'accesso completo all'applicativo WEB
Sotto requisiti	
001	Creare un utente amministratore e fare i controlli nelle sessioni

ID: REQ-04	
Nome	Accesso/Uscita
Priorità	1
Versione	1.0
Note	I sensori devono capire se una persona è entrata o uscita dalla porta
Sotto requisiti	
001	Collegare i sensori all'Arduino
002	Far interagire Arduino con PHP
003	Confrontarmi con un elettronico per capire come posizionare i sensori nella maniera più corretta

ID: REQ-05	
Nome	Margine errore basso
Priorità	1
Versione	1.0
Note	L'applicativo deve avere un margine di errore basso, quindi riuscire a capire quante persone passano anche in casi estremi
Sotto requisiti	
001	Collegare i sensori
002	Far interagire Arduino con PHP

ID: REQ-06	
Nome	Informazioni quando passa una persona
Priorità	1
Versione	1.0
Note	Ogni volta che passa una persona attraverso una porta devono venir salvati: Data, Ora, Aula e se si tratta di un'entrata o di un'uscita
Sotto requisiti	
001	Interagire con il database per inserire tutti i dati

ID: REQ-07	
Nome	Log
Priorità	1
Versione	1.0
Note	Tutti gli eventi vanno divisi nei log file, più precisamente in Info, Warning ed Error
Sotto requisiti	
001	Interagire con il database per inserire tutti i dati

ID: REQ-08	
Nome	Visualizzazione dei Log
Priorità	1
Versione	1.0
Note	L'amministratore deve poter visualizzare tutti i file di log, di cercarli e di filtrarli (anche stamparli)
Sotto requisiti	
001	Interagire con il database per inserire tutti i dati
002	Possibilità di filtrare i dati nella tabella che mostra i log
003	Possibilità di ordinare i dati nella tabella che mostra i log
004	Possibilità di mostrare solo un tipo di log (ad esempio mostrare unicamente gli errori, ...)

ID: REQ-09	
Nome	Statistica
Priorità	1
Versione	1.0
Note	Deve essere presente una statistica tra gli utenti che dovrebbero essere in un'aula e quelli effettivi, questa può essere anche stampata.
Sotto requisiti	
001	La statistica consiste nella differenza tra gli allievi che dovrebbero essere in aula e quelli che ci sono veramente (sfruttando il GAGI)
002	Bisogna comunicare con il GAGI

ID: REQ-10	
Nome	Archiviazione dati
Priorità	1
Versione	1.0
Note	I vecchi dati devono poter essere archiviati, quindi esclusi dalla statistica. In questo modo non verranno eliminati ma unicamente esclusi
Sotto requisiti	
001	Avere un database e comunicare con esso.

ID: REQ-11	
Nome	Tipologie utenti
Priorità	1
Versione	1.0
Note	I diversi tipi di utenti sono dettati dall'LDAP, ad esclusione dell'amministratore che è locale
Sotto requisiti	
001	Collegarsi all'LDAP della scuola
002	Distinguere le varie tipologie di utente

2.3 Use case

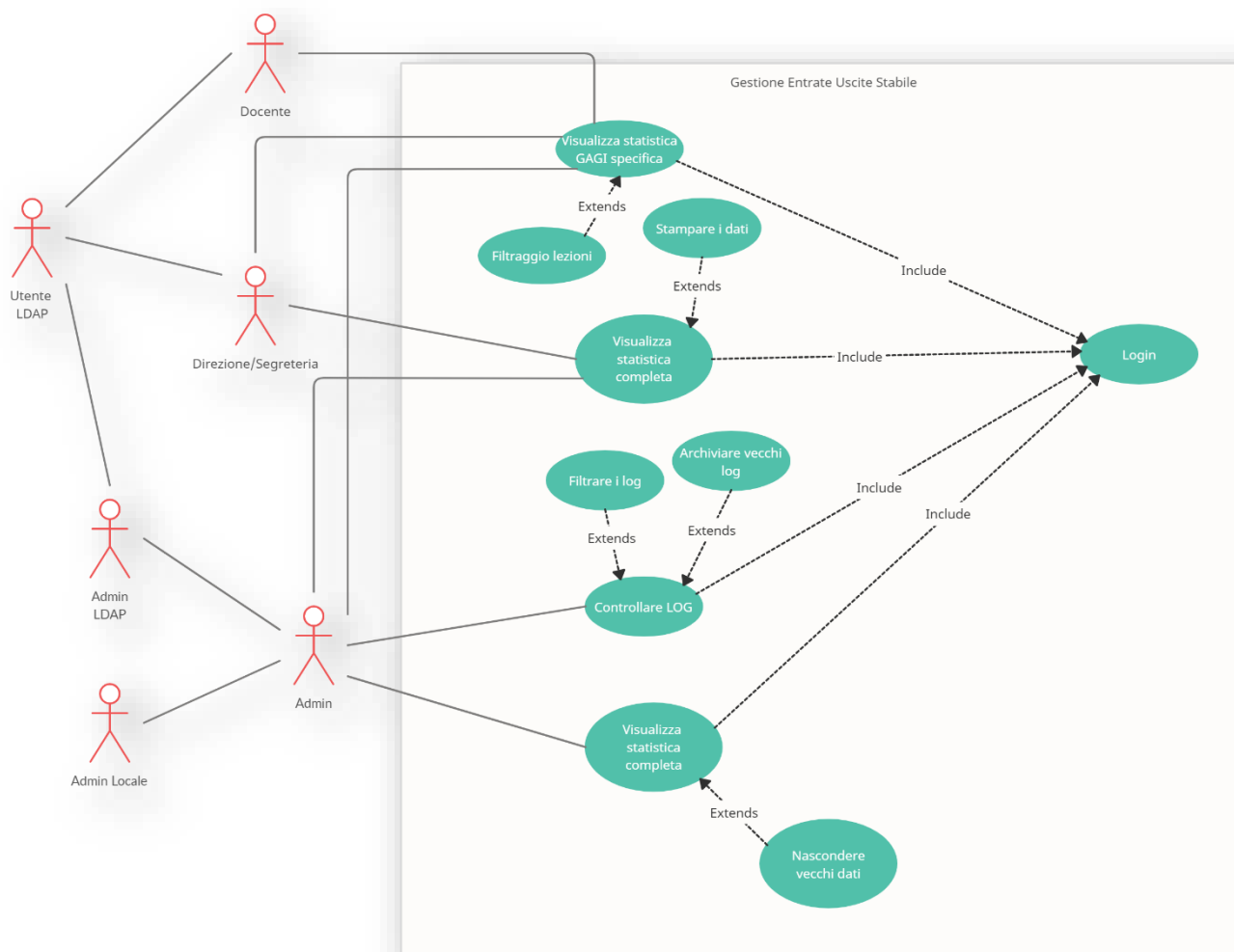


Figura 1 -- Use Case

All'interno dell'applicativo ho inserito 3 tipi diversi di utente, più specificatamente

- **Utente LDAP:** Gli utenti appartenenti a questo gruppo non sono presenti nel database (a parte che per l'Admin LDAP) ma sono presi direttamente dall'Active Directory della scuola.
 - **Docente:** L'unica interazione che avranno i docenti con l'applicativo sarà visualizzare una statistica che sarà confrontata con i dati di GAGI, essa riporterà il numero di allievi che sono presenti in aula e quelli che dovrebbero esserlo. Per visualizzare questo dato il docente dovrà inserire all'interno dell'applicativo il giorno della lezione, l'inizio, la durata e l'aula. Dati questi dati verrà riportata l'informazione con anche un andamento in base ai vecchi dati.
 - **Direzione/Segreteria:** La direzione e la segreteria avranno dei permessi in più confronto ai docenti, oltre a quello che già vedono i docenti in più potranno visualizzare le statistiche (entrate e uscite) riguardanti tutte le aule, in questo caso verrà implementato unicamente in un'aula ma sarà facilmente amplificabile. Ovviamente potendo vedere tutte le entrate e uscite avranno anche a disposizione dei grafici avanzati per vedere anche i dati in un lasso di tempo specifico. Per fare tutto ciò dovranno obbligatoriamente eseguire l'accesso tramite un account LDAP nel gruppo direzione o segreteria.
 - **Admin LDAP:** Questo utente avrà gli stessi identici permessi di un admin locale, l'unica differenza che alla sua creazione i suoi dati verranno estrapolati dall'LDAP ed inseriti all'interno del database degli amministratori così che quando cercherà di fare l'accesso l'applicativo saprà che si tratta di un amministratore.

- **Admin:** Oltre ovviamente a visualizzare tutte le informazioni degli altri utenti potrà in più controllare tutti i log (in una tabella apposita), li potrà filtrare o archiviare. Oltre a tutto questo l'utente può anche scegliere di archiviare dei vecchi dati per non mostrare più le informazioni sulle entrate e le uscite più vecchie. Questa operazione non eliminerà i dati ma li sposterà in una sezione apposita dove si potranno sempre visualizzare e riestrarre.

2.4 Pianificazione

La durata del progetto, come si può vedere dall'immagine nella prossima pagina, è leggermente più breve di quella prefissata. Questa scelta è stata fatta per aver un margine di errore (di all'incirca una settimana) che mi permetta di poterle sfruttare in caso che qualche attività dovesse prendere più tempo del dovuto. Per questo progetto sono riuscito a ricavare molte attività, avendo un Gantt con un complessivo di righe pari a 41. Le macrocategorie nelle quali possono includere tutte le attività sono le seguenti:

1. **Analisi:** All'interno di questa categoria ho compresso tutte la Attività riguardanti l'analisi del progetto in sé e la sua pianificazione. Teoricamente dovrebbe durare all'incirca 20.8 lezioni che hanno la durata complessiva di 15 ore e 30 minuti, ovvero il 10% del progetto.
2. **Preparazione:** Contenuto all'interno di questa categoria ci sono tutte le operazioni che vanno fatte precedentemente di iniziare a sviluppare l'applicativo, questo non è previsto nella pianificazione datoci però essendo un progetto che integra hardware e software la ritengo necessaria. Un esempio di quello che vi è all'interno è la richiesta della struttura LDAP della scuola o anche l'acquisto dei materiali hardware per la realizzazione del progetto.
3. **Implementazione:** Diviso in altre tre sottocategorie contiene tutta la parte pratica del progetto, dall'installazione dell'hardware sino alla scrittura del codice per l'applicativo WEB.
 - a. **Hardware:** Installazione e utilizzo base di tutte le componenti Hardware del progetto, come l'Arduino.
 - b. **Applicativo WEB:** L'applicativo visualizzato dagli amministratori, dalla direzione e dalla segreteria. La differenza sarà che l'amministratore avrà un accesso più completo rispetto agli altri
 - i. **LOG:** Parte molto importante dell'applicativo WEB, dovrà essere presente una tabella per l'amministratore che permetta di visualizzare tutti gli eventi in una determinata fascia oraria o tramite altri filtri. Oltre a questo si dovranno anche archiviare i vecchi dati.
4. **GAGI:** Ci sarà un'interazione tra il mio applicativo e GAGI che permette di visualizzare la differenza tra persone entrate in un'aula e quelli che dovrebbero essere all'interno di essa.

▲ Gestione Entrate Uscite Stabile	23.84 g	ven 08.01.21	mer 21.04.21	
▲ Analisi	1.71 g	ven 08.01.21	mar 19.01.21	
Intervista al mandante	180 m	ven 08.01.21	ven 08.01.21	
Requisiti	180 m	ven 08.01.21	ven 08.01.21	3
Attività	45 m	mar 12.01.21	mar 12.01.21	4
Pianificazione (Gantt)	45 m	mar 12.01.21	mar 12.01.21	5
Analisi dei mezzi necessari per lo sviluppo	60 m	mar 12.01.21	mar 12.01.21	6
Use Case	180 m	mar 12.01.21	gio 14.01.21	7
Design del database	90 m	gio 14.01.21	ven 15.01.21	8
Design delle interfacce	180 m	ven 15.01.21	mar 19.01.21	9
Definire comunicazione tra hardware e software	40 m	mar 19.01.21	mar 19.01.21	10
▲ Preparazione	0.38 g	mar 19.01.21	mer 20.01.21	2
Acquisto materiali	0 g	mar 19.01.21	mar 19.01.21	
Richiedere struttura LDAP	60 m	mar 19.01.21	mar 19.01.21	13
Richiedere accesso GAGI	60 m	mar 19.01.21	mer 20.01.21	14
Richiedere accessi web host	60 m	mer 20.01.21	mer 20.01.21	15
▲ Implementazione	14.44 g	mer 20.01.21	mar 23.03.21	12
Struttura database	180 m	mer 20.01.21	gio 21.01.21	
▲ Hardware	3.96 g	gio 21.01.21	mar 09.02.21	18
Installazione hardware	500 m	gio 21.01.21	mer 27.01.21	
Riconoscimento persone entrata	600 m	mer 27.01.21	mar 02.02.21	20
Riconoscimento persone uscita	600 m	mar 02.02.21	ven 05.02.21	21
Inserire dati nel database	200 m	ven 05.02.21	mar 09.02.21	22
▲ Applicativo web	8.23 g	mar 09.02.21	ven 12.03.21	19
Pagina di login	300 m	mar 09.02.21	gio 11.02.21	
Controllo degli accessi	150 m	gio 11.02.21	ven 12.02.21	25
Sicurezza del login (caratteri speciali,...)	200 m	ven 12.02.21	ven 12.02.21	26
Pagina home	300 m	ven 12.02.21	mer 17.02.21	27
Pagina statistiche	900 m	mer 17.02.21	mer 24.02.21	28
▲ LOG	3.96 g	mer 24.02.21	ven 12.03.21	29
Lettura dei log (tabella)	400 m	mer 24.02.21	ven 26.02.21	
Filtraggio dei dati	500 m	ven 26.02.21	gio 04.03.21	31
Ordinamento dei dati	500 m	gio 04.03.21	mar 09.03.21	32
Archiviazione dei dati	500 m	mar 09.03.21	ven 12.03.21	33
Inserire all'accesso anche gli utenti del LDAP	200 m	ven 12.03.21	ven 12.03.21	34
▲ GAGI	1.88 g	ven 12.03.21	mar 23.03.21	24
Mostrare differenza tra allievi presenti e entrati	900 m	ven 12.03.21	mar 23.03.21	
Test	1000 m	mar 23.03.21	mer 31.03.21	36
Documentazione	2000 m	mer 31.03.21	ven 16.04.21	38
Fine progetto	0 g	ven 16.04.21	ven 16.04.21	39
Consegna	0 g	mer 21.04.21	mer 21.04.21	

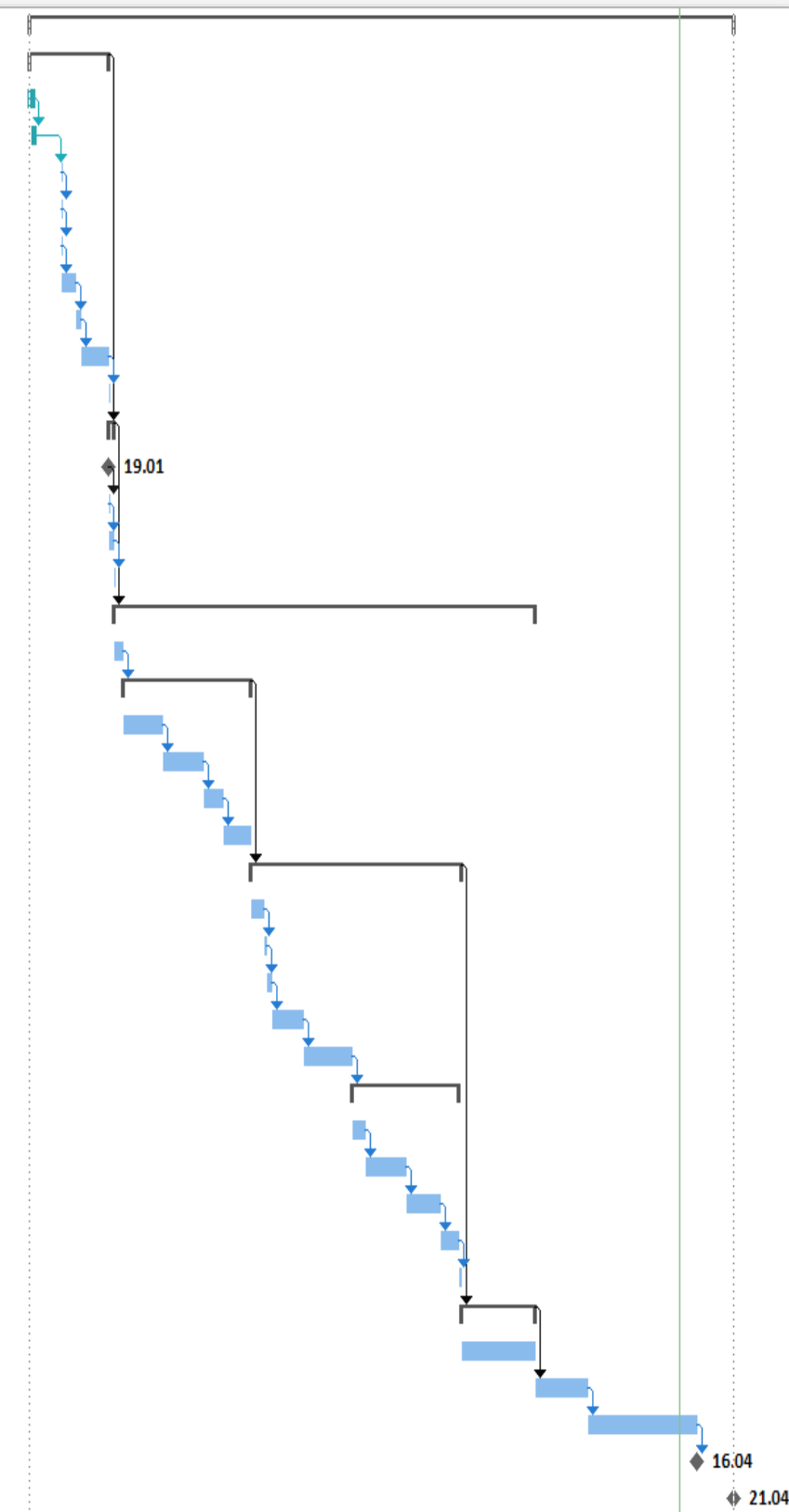


Figura 2 -- Gantt iniziale

Titolo del progetto: Gestione Entrate Uscite Stabile
Alunno/a: Matteo Arena
Classe: I4AA
Anno scolastico: 2020/2021
Docente responsabile: Guido Montalbetti

2.5 Analisi dei mezzi

2.5.1 Software

- **Visual Studio Code 1.52.1**
- **PHP 7.2**
- **HTML 5**
- **CSS 3**
- **Javascript 1.8.5**
- **MySQL**
- **Arduino IDE 1.8.13**
- **ADEplorer 1.50**
- **Composer 1.10.20**
- **ChartJS 2.9.4**
- **Ethernet 2.0.0 (Arduino)**
- **ArduinoThread 2.1.0 (Arduino)**
- **FileZilla 3.52.2**

Le versioni di PHP, MySQL ed Apache sono definite da quelle del host.

2.5.2 Hardware

- **3 sensori infrarossi + 1:** Questi sensori emettono un segnale (fino a 4 metri) per capire se il loro raggio è libero oppure ostruito da qualche corpo, nel progetto i corpi che interromperanno i raggi saranno le persone. Si è voluto tenere un sensore aggiuntivo in caso che uno di quelli che si testeranno dovesse danneggiarsi o addirittura rompersi.
- **3 riflettori infrarossi + 1:** Essi non sono altro che degli specchi per i sensori ad infrarossi, il loro compito è prendere il segnale dei sensori e rimandarli indietro per fargli capire che il raggio non è ostacolato da alcun oggetto (in caso che invece ci sia in mezzo un corpo il riflettore non potrà né ricevere né rimandare il segnale quindi il sensore capirà che c'è un corpo).
- **1 PC:** Le componenti del PC che verrà utilizzato per lo sviluppo dell'applicativo sono:
 - I7 – 9700
 - 32 GB RAM
 - SSD 512 GB
 - NVIDIA GeForce RTX 2060
- **1 Arduino:** Sarà il mezzo che si interfacerà con i sensori ed inserirà nel database i dati di accesso, quali la data, se si tratta di un'uscita o di un'entrata, ... Ovviamente per poter inserire dei dati nel database l'Arduino necessiterà di un modulo Wi-Fi/Ethernet.
- **1 Cavo Ethernet:** Esso servirà per collegare l'Arduino alla rete, probabilmente servirà un cavo abbastanza lungo (dai 5 ai 10 metri).
- **1 Alimentatore esterno** Utilizzato per alimentare il circuito, siccome la tensione dell'Arduino non era sufficiente.

3 Progettazione

3.1 Design dell'architettura del sistema

3.1.1 Posizionamento dei sensori ad infrarossi

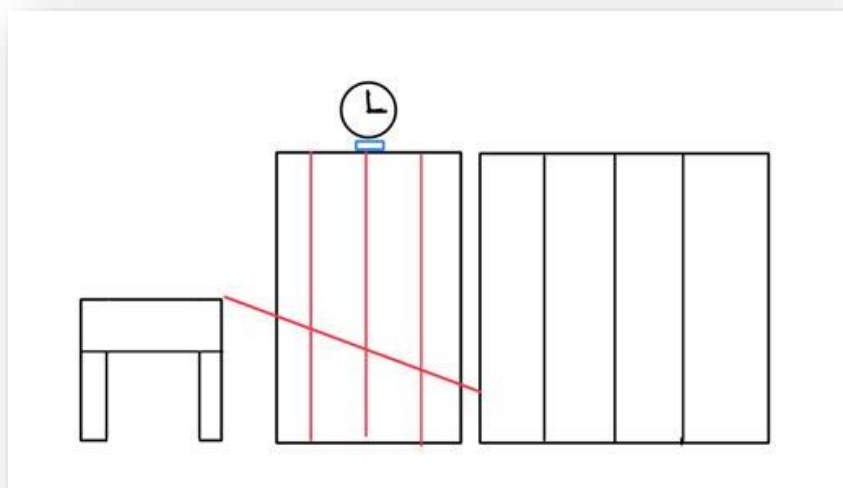


Figura 3 -- Possibilità 1

La prima soluzione adottabile, mostrata nell'immagine sovrastante posizionerebbe l'Arduino (in blu) al di sopra della porta e sotto di lui 3 sensori ad infrarossi, ovviamente adottando questa soluzione le placchette che fanno tornare il segnale andrebbero piazzate per terra. In aiuto a questi sensori è presente un quarto messo in obliquo che parte dall'angolo più vicino alla porta del tavolo ed arriva all'angolo più lontano dalla porta dell'armadio. Il problema di questa soluzione è ovviamente il fatto che per essere montata in qualsiasi aula ci sarebbe bisogno di montare dei distanziali per arrivare alla stessa lunghezza dell'armadio.

Il funzionamento sarebbe nel seguente modo: in caso che entri una persona essa potrà occupare i sensori (partendo da sinistra) 1 e 2 oppure 2 e 3 (se non addirittura solo 1 o 3). Questi due sensori manderanno l'input all'Arduino che hanno captato un movimento e infine quello messo in obliquo confermerà il dato ricevuto dai primi sensori. Infine in caso che dovessero entrare due persone contemporaneamente esse occuperebbero tutti e tre i sensori (oppure solo quelli esterni) ed è praticamente impossibile che entrambi passino nel modo perfetto per far sì che il sensore posizionato obliquamente non riesca a captarli insieme.

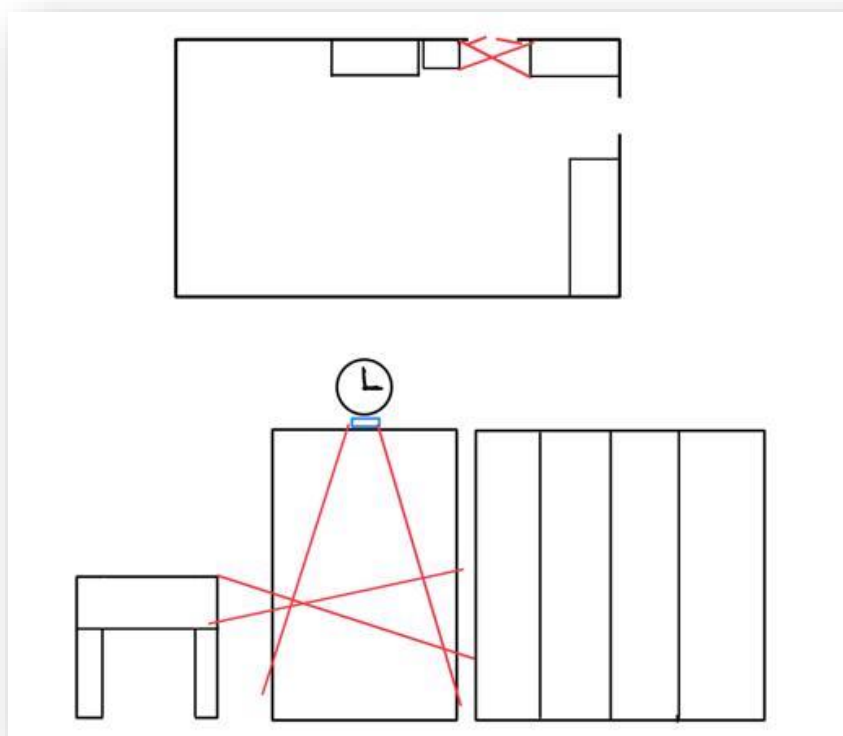


Figura 4 -- Possibilità 2

La seconda soluzione invece prevede di posizionare sempre 4 sensori ma in un ordine leggermente diverso. Innanzitutto si dovrebbero montare due sensori sopra la porta, i quali raggi arriverebbero sino ad in basso all'angolo della porta (facendo così la placchetta che rimanda indietro il segnale non andrebbe posizionata per terra e il sistema potrebbe durare di più nel tempo). Gli altri due sensori invece andrebbero montati il primo nell'angolo lontano del tavolo sino all'angolo vicino la porta dell'armadio, mentre il secondo al contrario (quindi il sensore partirebbe dall'angolo lontano dell'armadio e arriverebbe all'angolo vicino alla porta del tavolo). In questa soluzione le persone potrebbero essere prese più precisamente in caso che passino insieme ma probabilmente meno precisamente se dovessero passare singolarmente.

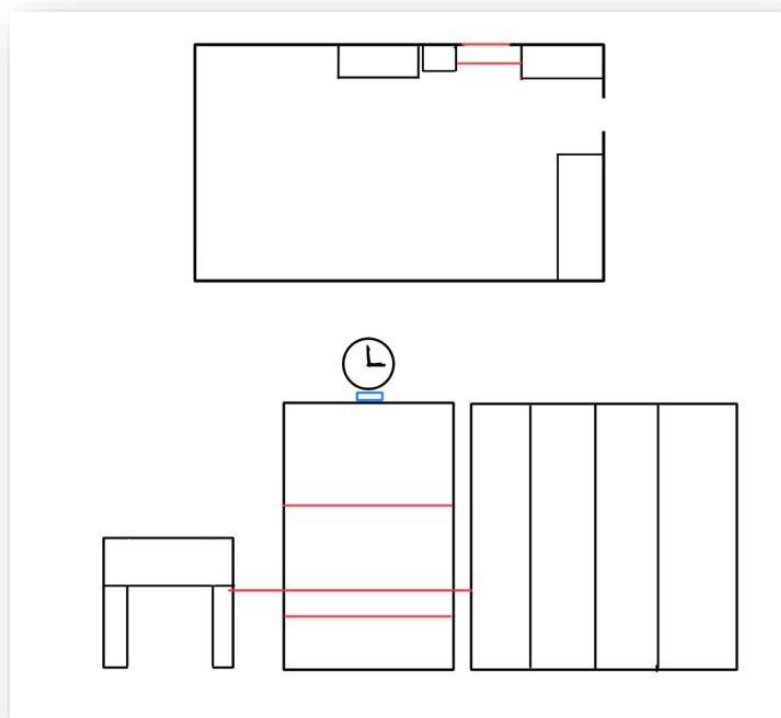


Figura 5 -- Possibilità 3

Come terza possibilità (insieme al mandante) si è pensato di posizionare due sensori a filo della porta, il primo all'altezza del polpaccio ed il secondo all'altezza del busto. Ovviamente queste posizioni non saranno esattamente nella stessa posizione per chiunque, perché una persona più alta si troverà i sensori più in basso rispetto ad una più bassa; per questo si posizioneranno i sensori ad un'altezza media. Infine l'ultimo sensore andrebbe posizionato sul tavolino sino alla parete, questo sensore avrà una distanza orizzontale rispetto a quelli a filo della porta di circa 50 centimetri. Quest'ultimo servirà per capire se una persona è entrata o uscita dalla porta. Tra tutte le soluzioni questa è sicuramente la più parsimoniosa dal lato dei sensori.

3.1.2 Struttura dell'applicativo WEB

La struttura dell'applicativo WEB si baserà su MVC, un framework base ma che basterà per il progetto in questione.

La struttura delle cartelle è la seguente e all'interno sono presenti i seguenti elementi:

config	27.01.2021 11:46	Cartella di file
controller	27.01.2021 11:46	Cartella di file
css	28.01.2021 13:56	Cartella di file
img	28.01.2021 14:19	Cartella di file
js	28.01.2021 13:26	Cartella di file
libs	27.01.2021 11:44	Cartella di file
models	27.01.2021 11:44	Cartella di file
views	28.01.2021 13:25	Cartella di file

Figura 6 -- Struttura MVC

- **Config:** Dentro questa cartella è presente un file di configurazione contenente il codice per mostrare tutti gli errori sulla pagina web e delle costanti che verranno utilizzate durante lo sviluppo. Queste costanti sono semplicemente la posizione delle cartelle *Img* e *Views*; oltre a questo c'è anche una costante che contiene l'URL per raggiungere l'applicativo WEB (in locale).
- **Controller:** Questa cartella, come si può facilmente intuire, conterrà tutti i controller. Essi sono i file che si occupano di prendere i dati dalle classi e darli alle views. Questi file sono pensati per avere una comunicazione specifica tra i dati effettivi e quello che vede l'utente, avendo così un codice estremamente più ordinato.
- **Css:** Cartella contenente tutti i file css (file per la veste grafica dell'applicativo web).
- **Img:** Contiene tutte le immagini che si visualizzeranno nell'applicativo.
- **Js:** Cartella che contiene tutti gli script javascript.
- **Libs:** Finora vuoto, qui ci saranno tutte le cartelle delle librerie che verranno utilizzate.
- **Models:** All'interno ci saranno tutte le classi che si occuperanno di gestire i dati, esse si interfaceranno con il database ed estrapoleranno i dati dal suo interno.
- **Views:** È quello che vede l'utente, ad esempio la pagina di login o quella principale.

3.2 Design dei dati e database

3.2.1 Database

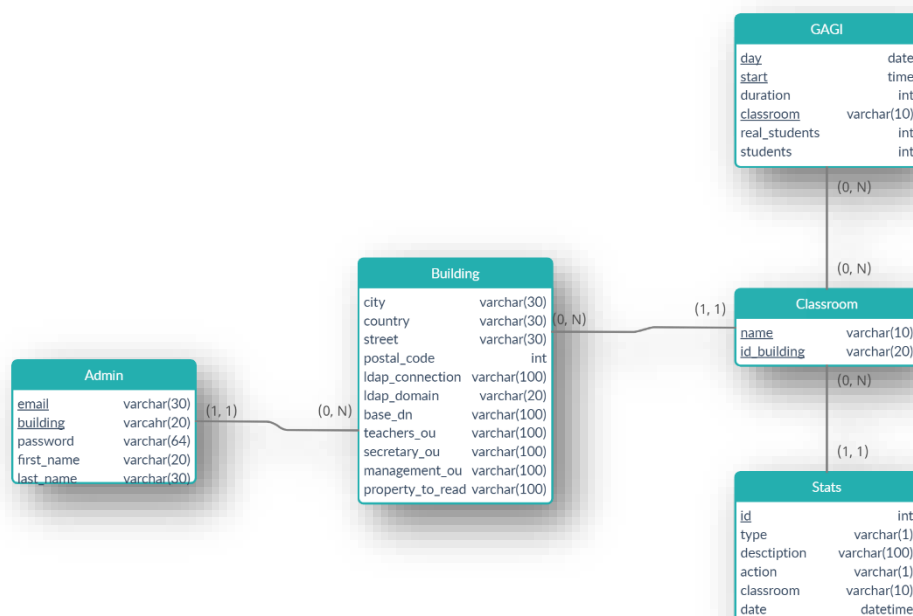


Figura 7 -- Database

La struttura del database si basa su 5 tabelle principali, queste sono tutte quelle necessarie all'utilizzo del software come richiesto sul QDC. Il database contiene le seguenti tabelle:

- **Admin:** All'interno di questa tabella sarà presente unicamente un amministratore locale e tutti quelli che quest'ultimo deciderà di fare amministratori direttamente dall'LDAP. Questo implicherà il fatto che il primo utente all'interno del database sarà inserito manualmente, mentre tutti gli altri saranno creati dall'amministratore originario. I dati che si vogliono sapere sono la mail, la password, il nome ed il cognome. Un amministratore può unicamente gestire uno stabile, quindi un amministratore non può gestire più stabili ma solo uno.
- **Building:** Rappresenta lo stabile nel quale si è collocato il sistema di controllo delle entrate e delle uscite, di esso si vuole sapere la posizione (tramite nome della città, stato, via e codice postale) e si attribuisce. All'interno di uno stabile c'è la possibilità di avere più aule e pure più amministratori. Importantissimo anche ldap_connection e ldap_domain che rispettivamente sono il server Active Directory e il nome del dominio dello stabile. In fine come ultimi parametri vengono chiesti il base_dn dello stabile, le OU per raggiungere le posizioni dei docenti, della direzione e del segretariato e la proprietà da leggere per effettuare il login (DN, ...).
- **Classroom:** Tabella che rappresenta l'aula nel quale si colloca il sistema di controllo delle entrate e delle uscite, le uniche informazioni che si vogliono sapere è il suo nome (che è anche il suo identificativo) e l'id dello stabile nella quale è collocata. Essa può fare parte unicamente di uno stabile (building), può avere più statistiche e può essere segnata più volte nel GAGI (per diverse lezioni).
- **Stats:** Contenente tutte le statistiche riguardanti un'aula specifica, i dati che si vogliono sapere sono di che tipo di informazione è, con la seguente formattazione:
 - **I:** Informazione
 - **W:** Warning
 - **E:** Error

Successivamente si vuole sapere anche una breve descrizione di quello che ha captato il sensore/sistema, l'aula nella quale è collocato il sensore e l'azione, seguendo la seguente formattazione: "I" in caso che una persona sia entrata, "O" in caso che sia uscita e "-" in caso che non si tratti di un'entrata o di un'uscita. Come ultima informazione è presente date, dato contenente l'orario ed il giorno in cui è accaduto qualcosa da registrare.

- **GAGI:** Contiene tutte le informazioni ricavate dal GAGI riguardante le lezioni, più precisamente il giorno in cui si svolge la lezione, l'inizio di essa, la durata, il nome dell'aula ed infine il numero di allievi della classe e quelli che si sono presentati.

3.2.2 Struttura LDAP

La richiesta del QDC richiede di permettere l'accesso ad uno stretto gruppo di utenti, più precisamente gli amministratori, la segreteria, i docenti e la direzione. Per gli ultimi tre è inoltre richiesto di prendere tutti i dati dal server LDAP della sede scolastica. Per fare ciò si è utilizzato un applicativo (sviluppato direttamente da Microsoft) per l'esplorazione del server, chiamato ADEplorer. Grazie ad esso sono riusciti facilmente a trovare tutti gli utenti dei quali necessitavo, quelli che si possono trovare evidenziati in rosso nell'immagine

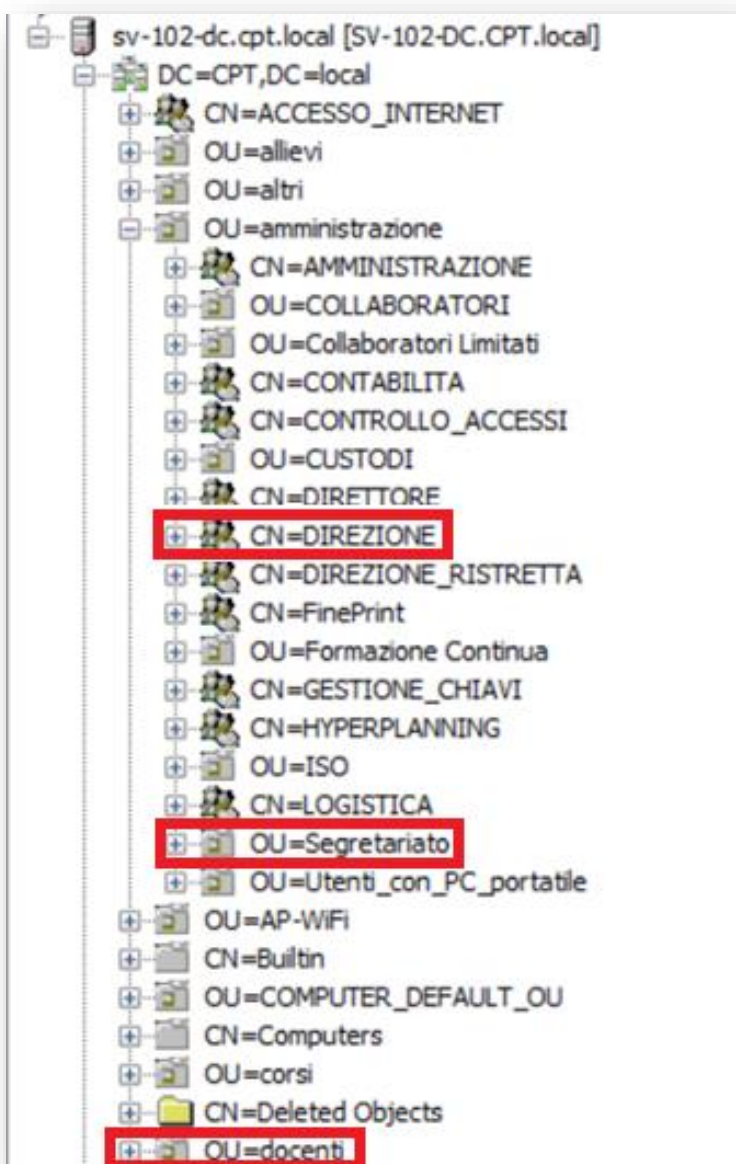


Figura 8 -- Struttura LDAP

sottostante. Come si può notare facilmente per il segretariato e i docenti è presente un OU (che contiene tutti gli utenti che ne fanno parte), mentre per la direzione è presente un gruppo (CN).

3.3 Design delle interfacce



Figura 9 -- Pagina principale

Come primissima interfaccia riporto quella che si visualizza non appena si effettua il login, quella in questione visualizzata al di sopra è quella dell'amministratore; l'unica differenza che avranno gli altri utenti saranno:

- **Docenti:** Visualizzeranno unicamente la voce *Ricerca* (che quindi si aprirà in automatico) e *Come funziona*, in modo che abbiano una guida (ovviamente contenente unicamente le operazioni che possono fare i docenti).
- **Segreteria/Direzione:** Oltre alle voci visualizzate dai docenti si aggiunge la voce *statistiche*, ovviamente per la voce *Come funziona* si aggiunge la guida per la funzione *Statistiche*.
- **Admin:** Tutte le voci visualizzate nell'immagine (*Log* e *Gestione Admin* fanno parte del menu dropdown chiamato *Amministrazione*).

Ovviamente la scritta *nome_utente* verrà modificata con il nome dell'utente all'interno dell'LDAP e cliccandoci sopra si potrà effettuare il logout. Al posto del campo *View da visualizzare* sarà presente la pagina che l'utente vorrà vedere (partendo da una di default come prima pagina).

Gestione Entrate Uscite Stabile						
Entrate Uscite	nome_utente					
Amministrazione	<input checked="" type="checkbox"/> Info	<input checked="" type="checkbox"/> Warning	<input checked="" type="checkbox"/> Error	da	21 Gen 2021	a 21 Gen 2021
	<input checked="" type="checkbox"/> Archiviato	<input checked="" type="checkbox"/> Visibile				
Log	ID	Tipo	Data	Descrizione	Visibilità	
Gestione Admin	1	Info	21.01.2020	Entrata nello stabile	Archiviato	
Statistiche	2	Warning	21.01.2020	...	Visibile	
Ricerca	3	Error	21.01.2020	...	Visibile	
Come funziona						

Figura 10 -- Tabella Log

Pagina visualizzabile unicamente dagli admin, la pagina che mostra tutti i file di log. Al suo interno verranno riportate tutte le informazioni sugli eventi che capitano in tutte le aule (ovviamente le varie differenziazioni vengono fatte nella voce descrizione). Sarà possibile filtrare tutti questi dati mediante le checkbox visualizzate al di sopra della tabella e cliccando sui tre puntini a destra di ogni riga sarà possibile archiviare o rendere visibile un file di log (in base se esso è archiviato o visibile) o addirittura eliminarlo.



Figura 11 -- Ricerca

Pagina visualizzabile a qualunque utente abbia l'accesso all'applicativo, *Ricerca* permette di cercare i dati per ogni aula in una determinata fascia oraria, saranno riportati gli allievi presenti in quel corso e quelli che dovrebbero essere presenti, oltre a riportare un grafico raffigurante l'andamento. Il grafico simboleggerà con

un colore diverso in base a quante persone il sensore ha riconosciuto (verde se sono esattamente il numero di allievi, arancione se sono un pelo meno e rosso se la classe risulta meno della metà).



Figura 12 -- Statistiche

Infine è presente la pagina visualizzabile unicamente dagli admin e dalla direzione/segreteria, riporterà a sinistra il numero di allievi presenti il giorno stesso mentre a sinistra un grafico dell'andamento dell'ultima settimana, mese oppure anno.

4 Implementazione

4.1 Montaggio Hardware

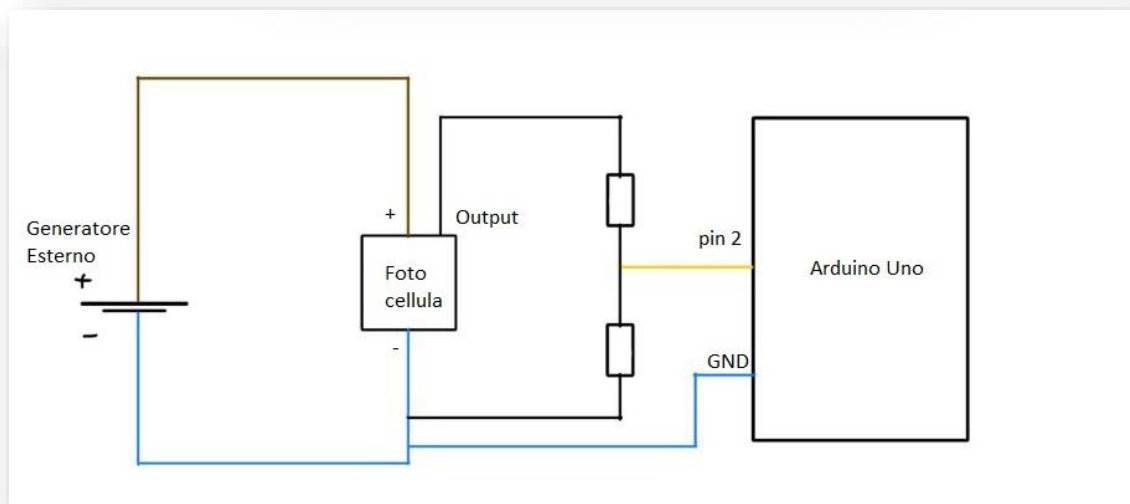


Figura 13 -- Circuito

Nell'immagine sovrastante è raffigurata la struttura del circuito finale, in esso viene rappresentato unicamente una fotocellula ma per la seconda la struttura non varia. Prima di tutto è presente un generatore esterno (raffigurato a sinistra dell'immagine) che alimenta direttamente la fotocellula tramite il pin marrone e la corrente esce dal pin blu, che ritorna al generatore. Facendo questo collegamento si ha già un sensore che funziona correttamente, il passaggio successivo è stato riuscire a leggere gli output attraverso l'Arduino.

La lettura dell'output viene fatta tramite il pin nero della fotocellula, esso però trasmette un dato con una tensione di 12 Volt ma l'Arduino è capace di leggere dati unicamente a 5 Volt (se non si rispetta questo valore la porta si potrebbe addirittura rompere). Per evitare questo ho calcolato quale resistenza inserire per far sì che la tensione si abbassasse dai 12 Volt ai 5 Volt, e il risultato è stato la prima (in alto) di 7,5kΩ e la seconda (in basso) di 5kΩ. Mettendo queste due resistenze si ha una tensione finale di 0 Volt e in mezzo alle due il valore corretto di 5 Volt. Esattamente in mezzo tra le due resistenze leggo il dato, che è diventato sicuro per l'Arduino. Oltre a questo è stato necessario anche mettere in comune il GND tra il generatore e l'Arduino, questo per evitare cali di tensione.



Figura 15 -- Fotocellula 1



Figura 14 -- Fotocellula 2

Quelle mostrate nelle immagini sopra sono le fotocellule che sono state montate all'interno dell'aula di test e sviluppo del progetto (A427 della Scuola Arti e Mestieri di Trevano). Le fotocellule danno un output visivo per mostrare il funzionamento, più precisamente se attraversate da qualcosa si accende un led arancione, mentre se non sono ostacolate (quindi il segnale ritorna) il led risulta spento.

Essi hanno di fronte a loro dei riflettori passivi (quindi non alimentati) che si occupano di riflettere il segnale emesso dalle fotocellule. Come si può notare l'implementazione finale è differente da quelle studiate durante la fase di analisi, ma si basa sull'ultima di esse. Infatti la struttura è la seguente.

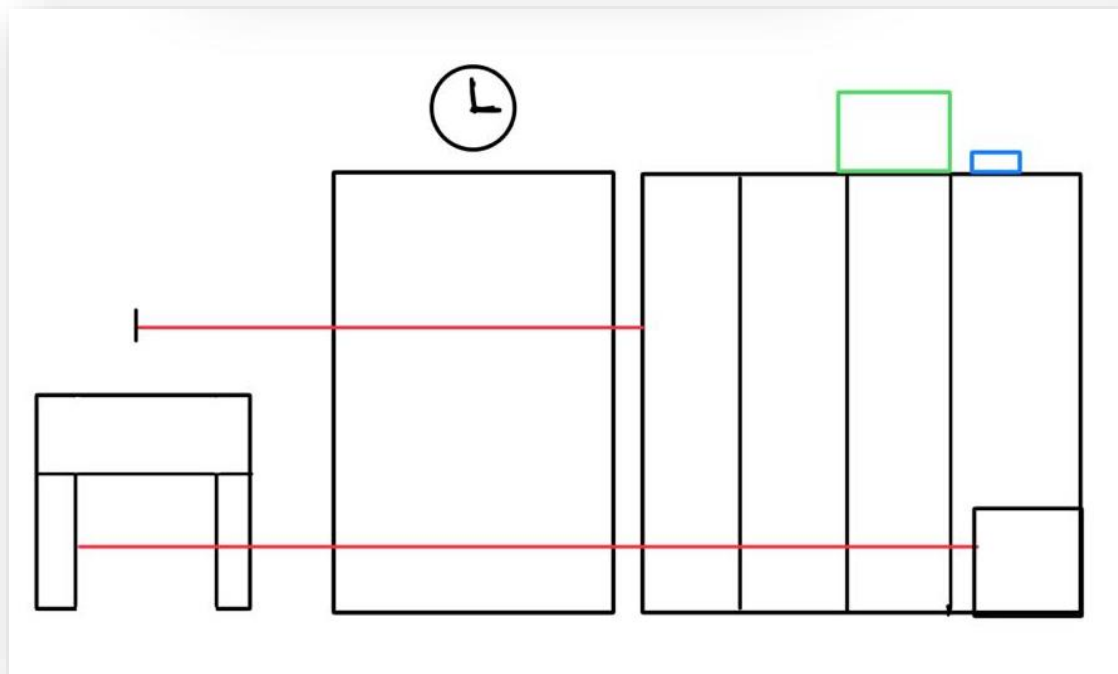


Figura 16 -- Piazzamento sensori

Il sensore in alto si occupa di leggere i busti mentre quello in basso (montato su un blocchetto apposta) si occupa di leggere le gambe. Inoltre l'Arduino è stato spostato sopra al mobile, insieme ad un generatore che all'inizio non era previsto.

4.2 Codice Arduino

Il codice dell'Arduino è stato sviluppato pensando alla semplicità, questo per far sì che non venga occupata troppa memoria RAM all'interno di esso.

```
#include <SPI.h>
#include <Ethernet.h>
```

Figura 17 -- Librerie Arduino

Come prima cosa includo le librerie delle quali successivamente avrò bisogno, le prime due (*SPI.h* e *Ethernet.h*) fanno parte della libreria Ethernet. Grazie a questi due "include" è possibile comunicare con il web server grazie all'Arduino (dotato di uno shield Ethernet).


```
//Indico mac address ed ip dell'Arduino
Ethernet.begin(mac, ip);

Serial.begin(9600);
pinMode(phOne, INPUT);
pinMode(phTwo, INPUT);

giveDataToDatabase("i", "Arduino%20riavviato%20e%20connesso", "-", "0");
```

Figura 19 -- Metodo setup

Nell'immagine sopra stante è raffigurato il metodo setup dell'Arduino, questo metodo viene richiamato unicamente ad Arduino avviato una sola volta. Come prima cosa imposto il Mac Address e l'IP dell'Arduino; come detto precedentemente il Mac Address è posizionato al di sotto dell'Ethernet Shield dell'Arduino e l'IP mi è stato assegnato da un sistemista. Fatto ciò imposto i pin sui quali sono collegate le due fotocellule che utilizzo (nel caso specifico nei pin 2 e 3). Come ultimissima cosa richiamo il metodo *giveDataToDatabase* che si occupa (dati i parametri) di inviare una richiesta tramite metodo POST al web server, che a sua volta si occuperà di inserire quei dati all'interno del database.

```
//Connessione al server
if(client.connect(server, 80)){
    //Preparazione dato da mandare
    String data = "type="+type;
    data += "&description="+description;
    data += "&action="+action;
    data += "&classroom=" + classroom;
    data += "&id_building=" + id_building;
    data += "&visible=" + visible;
    if(Ethernet.begin(mac) == 0){
        Serial.println("Failed to configure Ethernet using DHCP");
        while(1);
    }
}
```

Figura 18 -- Metodo *giveDataToDatabase* 1

Quello mostrato sopra è la prima parte del metodo *giveDataToDatabase*, esso viene richiamato ogni volta che si vuole inserire qualcosa all'interno del database. Nella prima parte tento la connessione al database (utilizzando l'oggetto client) e, se dovesse andare bene, preparo la stringa contenente tutti i dati che voglio salvare nel database separando i valori da un "&".

```
Serial.println("Connected to server");
//Manda la richiesta POST
client.println("POST /i17aremat/EntrateUscite/data.php HTTP/1.1");
//Indica l'host
client.println("Host: samtinfo.ch");
client.println("Content-Type: application/x-www-form-urlencoded");
//Lunghezza del dato da inoltrare
client.print("Content-Length: ");
client.println(data.length());
client.println();
//Inoltro del dato.
client.println(data);
client.println();
```

Figura 20 -- Metodo *giveDataToDatabase 2*

Preparata la stringa stampo sul terminale dell'Arduino che esso è riuscito a collegarsi al server (questo per debug in caso che si voglia collegare un computer) ed indico la pagina da richiamare, che è quella che gestisce le richieste POST. Infine invio la lunghezza dei dati che voglio inoltrare al web server e invio la variabile *data*, contenente per l'appunto tutti i dati che si vogliono inserire nel database.

Il codice che si occupa di riconoscere una persona in entrata/uscita è pensato per lavorare nel medesimo momento, questo significa che sono presenti delle variabili per controllare se una persona è entrata e delle variabili per controllare se una persona è uscita. Questo evita dei malfunzionamenti del codice, ad esempio che viene calcolata una persona in entrata quando invece ne sono uscite due, senza occupare troppo spazio nella memoria dell'Arduino.

Il funzionamento del codice Arduino è estremamente semplice, come prima cosa c'è da sapere che Arduino prevede l'utilizzo di un metodo chiamato *loop*. Esso viene richiamato continuamente, si comporta quindi come un vero e proprio loop infinito. All'interno di esso è presente la logica che controlla se una persona è entrata oppure è uscita.



Figura 21 -- Logica persona entrata

Sopra si vede il funzionamento pratico (semplificato in uno schema) di come funziona con esattezza il riconoscimento di una persona in entrata. Come prima cosa la fotocellula 1 (*F1*) deve essere disattiva, dopo di che si controlla che essa venga attivata. Attivata essa si passa a controllare che venga attivata anche la fotocellula 2 (*F2*), questo porta all'entrata di una persona. Ovviamente la logica per l'uscita di una persona funziona esattamente alla stessa maniera ma al contrario.

4.3 Struttura base

La base dell'applicativo WEB è il framework MVC, questo prevede di avere tre macro categorie di file che conterranno tutto il codice, per averlo più ordinato. Queste macro categorie (che si ritrovano anche nel nome delle cartelle) sono:

- **Models:** Classi che si occupano di gestire i dati, ad esempio prendere i dati dal database e prepararli per avere un dato accessibile ed utile. Quello che fanno essenzialmente è avere il comportamento di oggetti reali nel mondo reale con il quale si può interagire tramite i metodi.
- **Views:** Le views (o viste) sono la parte dell'applicativo che vede l'utente finale, questo significa che all'interno di questa categoria è presente soprattutto codice HTML e CSS che compongono l'interfaccia WEB dell'applicativo.
- **Controllers:** Infine i controllers si collocano a metà tra le classi e le viste, essi preparano i dati da far visualizzare all'utente, ad esempio un controller si può occupare di preparare la lista del contenuto della tabella con i log.

Oltre a queste macro categorie ne sono presenti anche altre, che però non sono previste all'interno della struttura MVC, ad esempio la cartella con le immagini, quella con gli script etc.

4.4 Models

Come spiegato prima le classi in questo applicativo servono, come anche per ogni altro applicativo, a semplificare moltissime operazioni che altrimenti sarebbero ridondanti (siccome andrebbero fatte più volte).

4.4.1 Database

Database è senza ombra di dubbio la classe che viene più utilizzata all'interno dell'applicativo. Il suo codice è estremamente semplice ma allo stesso momento riesce a semplificare moltissimi procedimenti. Lo scopo di essa, come si può intuire dal nome, è interfacciarsi con l'intero database. Questo non significa che tramite essa si fa le query ad ogni tabella (per quello si passa attraverso altre classi) ma che si occupa di gestire le query da fare al database e di ritornare il risultato a chi ne necessita, la maggior parte delle volte le altre classi.

```
public function __construct()
{
    self::getConnection();
}
```

Figura 22 -- Metodo costruttore database

Nel metodo costruttore la unica cosa che si va a fare è richiamare un metodo, statico, all'interno della classe stessa; esso si occupa di creare la connessione al database (se non è già presente) e di salvarla all'interno del parametro denominato *connection*.

```
public static function getConnection()
{
    try {
        //Se non esiste ancora fa la connessione.
        if (!isset(self::$connection)) {
            //Utilizza PDO per fare la connessione.
            self::$connection = new PDO(DSN, DB_USERNAME, DB_PASSWORD);
            self::$connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }
        //In caso di qualche errore lo mostra.
    } catch (PDOException $e) {
        echo "Errore nella connessione";
    }
    //Ritorna la connessione.
    return self::$connection;
}
```

Figura 23 -- getConnection database

Il metodo che viene chiamato all'interno del costruttore, mostrato nell'immagine sovrastante, si occupa di creare la connessione al database in caso che essa non sia già impostata. Come prima cosa appunto controlla che la connessione sia presente, se dovesse essere così salta tutto il procedimento e ritorna direttamente la connessione che era già stata calcolata. In caso invece che la connessione non sia ancora presente, il metodo istanzia un nuovo oggetto di tipo PDO (oggetto per interfacciarsi con il database) e gli imposta le credenziali di accesso; definite all'interno del manuale degli accessi. Le uniche cose da sapere sono i differenti formati per le varie stringhe, che devono essere:

- **DSN:** "mysql:host=<host>; dbname=<nome_database>; port=<db_port>" nel caso del mio progetto ho usato mysql quindi la stringa è la medesima ma per altri linguaggi potrebbe cambiare leggermente. Oltre a questo la porta del database è opzionale se è quella di default, quindi la 3306 per mysql.
- **DB_USERNAME:** Username che ha l'accesso al database.
- **DB_PASSWORD:** Password dell'utente con il quale si vuole accedere al database.

4.4.2 Admin

La prossima classe, denominata Admin, rappresenta un utente amministratore. Essa come proprietà ha *email*, *building* e *database* (ovviamente sono tutte private). Nel metodo costruttore viene richiesto di passare *email*, che è quella con la quale l'utente amministratore esegue l'accesso, e lo stabilimento (denominato *building*) presso il quale vuole accedere. Ovviamente per eseguire l'accesso ad uno stabile deve essere registrato a quello stabile come amministratore, altrimenti l'accesso non verrà effettuato. I metodi che sono contenuti nella classe sono:

- **getAdmins:** Permette di ricevere una lista di tutti gli amministratori presenti all'interno del database che sono nel medesimo stabile dell'amministratore che esegue l'accesso.
- **remove:** Rimuove l'utente amministratore, non uno passato tramite argomento ma direttamente quello specificato nel metodo costruttore.
- **getBuildings:** Ottiene una lista di tutti gli stabilimenti nei quali l'amministratore in questione ha permessi elevati, quindi gli stabili nel quale è amministratore.

4.4.3 Building

Building è la rappresentazione degli stabili, questa viene usata per controllare se un certo stabile esiste o meno e per leggere soprattutto i dati dal database (degli stabilimenti). I parametri della classe sono *database* per ottenere i dati dal database, *domainName* che sarebbe il nome della struttura e *exists* che non viene richiesto nel costruttore e diventa vero unicamente se lo stabile esiste. Questa ultima variabile diventando vera “sblocca” tutti i metodi che eseguono le query al database, altrimenti sarebbero query inutili che non ritornerebbero nulla e farebbero perdere tempo all’applicativo. Come per la classe *Admin* anche qui è presente un metodo per eliminare lo stabilimento, mostrato nell’immagine sottostante.

```
public function remove() {
    $query = "DELETE FROM building WHERE ldap_domain = '".$this->domainName.'";
    $this->database->executeQuery($query);
}
```

Figura 24 -- remove Building

Grazie all’immagine che mostra il funzionamento dell’eliminazione dello stabile si può vedere come funziona la classe *Database*. Prima di tutto viene istanziata la classe *database* all’interno del costruttore, dopodiché quando si ha bisogno di ricavare dei dati si richiama il metodo *executeQuery* e si specifica come argomento la query che si vuole eseguire. Come ritorno si hanno tutti i dati che si sono richiesti, in questo caso che si tratta di un *delete* non si ha alcun tipo di ritorno.

4.4.4 Classroom

Come per le altre classi essa rappresenta una classe scolastica rappresentata tramite codice, che si interfaccia con l’omonima tabella all’interno del database. Anch’essa come parametri (oltre a *database*) ha le chiavi della tabella, ovvero *building* (nome dello stabile) e *name* (nome dell’aula). La differenza con le altre classi è che quando si istanzia questo oggetto non deve per forza esistere la medesima aula nel database, ma si può creare. Questo viene fatto specificando un’aula che non esiste nel costruttore e poi richiamando il metodo *create*, che si occupa di creare la classe all’interno del database. Anche qui è presente il metodo *exists* e *delete*, oltre a tutti i metodi *getter* per ottenere i dati dal database.

4.4.5 DataChecker

DataChecker è una classe particolare pensata unicamente per il controllo delle stringhe. Gli unici metodi che ha sono statici, questo quindi implica che la classe non ha bisogno di un metodo costruttore siccome risulterebbe inutile.

```
public static function checkInput($data){
    //Rimuove gli spazi esterni
    $data = trim($data);

    //Rimuove gli slash
    $data = stripslashes($data);

    //Rimuove i caratteri speciali
    $data = htmlspecialchars($data);

    //ritorna il dato filtrato
    return $data;
}
```

Figura 25 -- controllo degli input

Tramite il metodo *checkInput*, mostrato nell'immagine, controllo che una stringa passata come argomento non abbia caratteri che possano mettere a rischio la struttura dell'applicativo WEB. Come primo controllo vengono direttamente eliminati tutti gli spazi a sinistra e a destra della stringa, dopodiché vengono rimossi gli slash (essi potrebbero avere dopo di sé dei caratteri speciali) ed infine controllo i caratteri speciali. Infine ritorno la stringa che è diventata sicura.

```
public static function checkPassword($data){
    return hash("sha256", self::checkInput($data));
}
```

Figura 26 -- Controllo password

Infine come secondo ed ultimo metodo è presente *checkPassword*, esso serve a controllare le password. Come prima cosa viene richiamato il metodo *checkInput*, dopo di che viene eseguito un hash tramite l'algoritmo *sha256* della stringa. Grazie a questo algoritmo rendo la stringa estremamente meno decriptabile, sarebbe richiesto troppo tempo.

4.4.6 GAGI

GAGI è una classe che si interfaccia con l'omonima tabella all'interno del database. Nel costruttore viene richiesto unicamente il nome dello stabilimento. Inizialmente era previsto che questa classe prendesse i dati direttamente da GAGI, ma non essendo questo possibile per motivi di privacy è stato necessario fare una tabella all'interno del database dell'applicativo.

```
public function getTotalUserEntered($date){
    $query = "SELECT SUM(real_students) FROM gagi WHERE id_building = '". $this->building.'" AND day = '". $date.'" ";
    return $this->database->executeQuery($query)[0][0];
}
```

Figura 27 -- *getTotalUserEntered* GAGI

Sopra è mostrato il metodo per ottenere tutti gli allievi che sono stati segnati in GAGI, quindi gli allievi effettivi che sono in aula. Come argomento viene richiesta la data della quale si vuole sapere quante persone sono entrate in aula, questo metodo torna quindi tutta la giornata e non un orario specifico. Questo serve per il conteggio delle persone totali all'interno della scuola in un giorno.

```
public function getStudentsInClassroom($date, $start, $delta, $classroom){
    $start = date("H:i:s", strtotime($start));
    $query = "
        SELECT real_students
        FROM gagi
        WHERE
            id_building = '". $this->building.'"
            AND day = '". $date.'"
            AND start = '". $start.'"
            AND duration = '". $delta.'"
            AND classroom = '". $classroom.'" ";
    $data = $this->database->executeQuery($query);
    if(sizeof($data) > 0){
        return $data[0][0];
    }else{
        return $data;
    }
}
```

Figura 28 -- *getStudentsInClassroom* GAGI

Altro metodo interessante è *getStudentsInClassroom*, mostrato nell'immagine, esso fa il medesimo lavoro del metodo *getTotalUserEntered* ma in un orario specifico. Più precisamente viene richiesto di passargli tramite argomento i seguenti dati:

- **Date:** Data nella quale si è svolta la lezione, con giorno mese ed anno.
- **Start:** Inizio della lezione, con ora minuti e secondi.
- **Delta:** Durata, in minuti, della lezione.
- **Classroom:** Aula nella quale si è svolta la lezione.

Come altri metodi la classe ha *getEnteredUsers* e *getCurrentMonth* che rispettivamente ritornano il numero di persone entrate in aula in un giorno ed in un mese. Nel secondo metodo è presente un ciclo che si occupa di ottenere giorno per giorno tutti i dati, in caso che in un giorno non siano presenti allievi in aula ritorna 0 come dato (altrimenti i dati del giorno seguente sarebbero intesi come quelli del giorno precedente).

4.4.7 Log

Altra classe che si interfaccia con il database, principalmente ritorna i dati all'interna dell'omonima tabella. Gli unici parametri che possiede questa classe sono il nome dello stabilimento e l'oggetto database, che però non viene passato tramite argomento nel costruttore. Una volta istanziato assegna al parametro *building* il nome dello stabilimento ed istanzia l'oggetto *database*.

Primissimo metodo al suo interno *getAllLogs* permette di ottenere tutti i log di uno stabilimento, il medesimo stabilito nel metodo costruttore. Successivamente si trovano due metodi complementari, ovvero *archive* e *disarchive*. Come si può intuire questi due metodi vanno ad archiviare o di togliere dall'archivio. Questi metodi si occupano di modificare una colonna della tabella denominata *visible*, portandola da 0 ad 1 in caso di rimozione dall'archivio e viceversa per l'archiviazione. Per entrambi questi metodi hanno bisogno di ottenere tramite argomento l'Id del log interessato, questo numero non sarà legato allo stabile ma sarà univoco di ogni log all'interno della tabella.

Successivamente si trova il metodo *getDateInterval*, utilizzato per ottenere i log contenuti in un certo intervallo di tempo. Il funzionamento di questo metodo prevede che la prima data sia inclusa all'interno dell'intervallo mentre la seconda esclusa.

Infine come per GAGI sono presenti i metodi per ottenere il numero di allievi entrati in una certa giornata o addirittura nell'ultimo mese.

4.4.8 User

La classe *User* è quella che si deve occupare del login degli utenti normali (direzione, segreteria e docenti), quindi per funzionare deve collegarsi al server AD della scuola. Per fare ciò mi sono basato su una libreria trovata su GitHub denominata adLDAP2. Per installarla ci si reca nel *Prompt dei comandi* e si digita il seguente comando:

```
composer require adldap2/adldap2
```

Fatto ciò (nella directory del progetto) ho potuto includere la libreria tramite i seguenti *use*.

```
use Adldap\Adldap;
use Symfony\Component\VarDumper\Cloner\Data;
```

Figura 29 -- use Adldap2

Inclusi i vari file ho proceduto con la struttura base della classe, ovvero i suoi parametri ed il costruttore. I parametri includono il nome dell'utente, la password di esso, lo stabilimento al quale appartiene, il nome, il cognome, il tipo di utente e come sempre l'oggetto database. Nel costruttore richiedo unicamente l'username dell'utente, la sua password e lo stabilimento; gli stessi dati che si ritroveranno all'interno della pagina di login dell'applicativo.


```
public function login(){
    //Crea l'oggetto ADLDAP
    $ad = new Adldap();

    //Ottenimento dei dati necessari per il login
    $baseDn = $this->building->getBaseDN();
    $domain = $this->building->getLdapDomain();

    //Impostazioni di connessione
    $config = [
        'hosts' => [$this->building->getLdapConnectionString()],
        'base_dn' => $baseDn,
        'timeout' => 2,
        'username' => $domain . "\\\" . $this->username,
        'password' => $this->password,
    ];

    //Aggiunge la configurazione alla classe.
    $ad->addProvider($config);

    //Tenta la connessione
    try {
        $provider = $ad->connect();

        //Controlla se è presente un utente chiamato come quello che sta cercando di fare il login
        $results = $provider->search()->where($this->building->getPropertyToRead(), '=', $this->username)->get();

        //Ottiene i dati
        $array = json_decode($results, true);

        //Legge il dn
        $info = $array[0]['distinguishedname'];
        $distinguishedname = explode(",", $info[0]);
        $ou = str_replace("OU=", "", $distinguishedname[count($distinguishedname) -3]);
    }
```

Figura 30 -- login parte 1

Il metodo di login è sicuramente il metodo più complesso all'interno della classe, come prima cosa si istanzia un oggetto di tipo ADLDAP (ricavato dalla libreria inclusa). Dopo di che tramite l'oggetto *building* ricavo la *basedn* ed il *domain* che rispettivamente sono la stringa di connessione per LDAP e il dominio dell'AD dello stabile. Avendo questi due dati si procede con la configurazione della connessione, visibile nella variabile *config*. All'interno di questa variabile viene salvato:

- **Host:** Host al quale collegarsi per effettuare la connessione LDAP:
- **Base_dn:** Stringa di connessione, è salvata all'interno del database.
- **Timeout:** Tempo limite in secondi per la connessione, in questo caso il tempo è di 2 secondi.
- **Username:** Username dell'utente che vuole effettuare l'accesso.
- **Password:** Password dell'utente che vuole effettuare l'accesso.

Avendo la configurazione pronta la si aggiunge tramite il metodo *addProvider* ed in seguito si esegue la connessione tramite il metodo *connect*. Avendo la connessione (in caso che non avvenga viene attivato il *catch*) si continua con la ricerca dell'utente che ha effettuato l'accesso. Per fare ciò si richiede al database la proprietà da leggere, solitamente *samaccountname* e si ottengono tutti gli utenti che si chiamano come quello che ha effettuato l'accesso. Fatto ciò si decodifica il dato tornato sotto forma di json e si legge il *dn* (*distinguishedname*) ed infine si ottiene la OU nella quale è contenuto l'utente.

```
//In base alle OU in cui è presente effettua tipi diversi di login
if($ou === $this->building->getManagementOU()){
    $this->type = 'management';
}
else if($ou === $this->building->getTeacherOU()){
    $this->type = 'teacher';
}
else if($ou === $this->building->getSecretaryOU()){
    $this->type = 'management';
}
else{

    //Ritorna falso se non è nelle OU
    return false;
}
return true;
} catch (\Adldap\Auth\BindException $e) {
    return false;
}
```

Figura 31 -- login parte 2

Avendo trovato la OU al quale appartiene l'utente si controlla se sia una delle quali ci si può fidare per lasciar effettuare l'accesso all'utente. In caso che faccia parte di uno di quegli utenti che ha il permesso viene salvato che tipo di utente è all'interno dell'apposita proprietà e ritorna *true*, in caso invece che non esista o che non faccia parte di una delle OU interessate ritorna *false*.

```
public static function addUser($login, $building){
    $query = "INSERT INTO admin (email, building, password) VALUES ('".$login."', '".$building."', '-')";
    $database = new Database();
    $database->executeQuery($query);
    header("location:".URL."main/loadAdminsManagement");
}
```

Figura 32 -- addUser user

Altro metodo interessante quello che si occupa dell'aggiunta degli utenti all'interno del database. Esso ho deciso di farlo statico siccome non necessita di alcun parametro della classe. Come argomento richiede il nome utente e lo stabile nel quale è contenuto, esso infatti serve a creare un utente amministratore. Come si può notare non viene richiesta una password perché essa verrà richiesta successivamente all'utente quando effettuerà il primo accesso. Di conseguenza nascono in automatico i metodi *passwordIsSet* per controllare se un utente ha la password impostata e *addPassword* per configurare la password dell'utente. Ovviamente per il secondo metodo è necessario passare una stringa codificata tramite hash.

4.5 Controllers

Come detto precedentemente i controllers si occupano di prendere i dati dai models e servirli alle views, questo permette una struttura più ordinata dell'applicativo.

4.5.1 AdminLogin

AdminLogin è il controller che si occupa, come dice il nome, di gestire l'autenticazione dell'utente amministratore.

```
public function index(){

    //Indica che l'utente è un amministratore
    $admin = true;

    //Titolo della pagina
    $title = "Accedi con account amministratore";

    //Require delle view necessarie
    require 'application/views/header.php';
    require 'application/views/Login/sign-in.php';
    require 'application/views/footer.php';

}
```

Figura 33 -- index AdminLogin

Come prima cosa nel metodo index (quello che viene richiamato la prima volta) imposto che l'utente che andrà a effettuare l'accesso è un amministratore e imposto il titolo della pagina, *Accedi con account amministratore*. Infine richiamo tutte le views necessarie alla corretta visualizzazione della pagina di login.

```
//Ottenimento degli input inseriti dall'utente
$email = DataChecker::checkInput($_POST['email']);
$building = DataChecker::checkInput($_POST['domainName']);
$password = DataChecker::checkPassword($_POST['password']);

//Impostazione delle variabili di sessione
$_SESSION['username'] = $email;
$_SESSION['building'] = $building;

//Istanziamento dell'utente
$user = new User($email, $password, $building);
```

Figura 34 -- AdminLogin login parte 1

Effettuato il login da parte dell'amministratore vengono ottenuti tutti gli input digitati da esso, vengono controllati (e nel caso della password anche codificata), imposto nelle variabili di sessione tutte le informazioni dell'utente ed infine istanzio l'oggetto *User*.

```
//In caso che sia impostata
if($user->passwordIsSet()){

    //In caso che esista l'utente amministratore
    if($user->loginAdmin() > 0){

        //Imposta il tipo di utente che ha effettuato il login
        $_SESSION['type'] = 'admin';

        //Porta l'utente alla pagina principale
        header("location:".URL."main");

    }

    //In caso che l'utente non esiste
    }else{

        //Imposta l'errore da mostrare all'utente
        $_SESSION['error'] = "Accesso non riuscito, riprova";

        //Ricarica la pagina di accesso amministratore
        header("location:".URL."adminlogin");

    }
}

//Se la password dell'utente non è impostata
}else{

    //Chiede all'utente di registrare la password
    require 'application/views/header.php';
    require 'application/views/Login/registerPassword.php';
    require 'application/views/footer.php';

}
```

Figura 35 -- AdminLogin login parte 2

Avendo l'oggetto istanziato viene controllato se la password dell'utente è impostata (altrimenti dovrebbe riportare l'amministratore all'impostazione di quest'ultima) e successivamente che esista nel database quest'ultimo. In caso che questo controllo vada a buon fine viene impostato che l'utente è un amministratore e viene portato l'utente alla pagina principale dell'applicativo. In caso contrario invece viene ricaricata la medesima pagina (quella di login degli amministratori) e viene visualizzato l'errore *Accesso non riuscito, riprova*. Ho deciso di inserire un errore specifico per motivi di sicurezza, infatti in caso che venga azzeccato lo stabilimento da un malintenzionato non viene specificato nell'errore che ha indovinato quest'ultimo. Infine nel controller è presente un metodo che viene richiamato dal controllo che la password sia impostata che permette all'utente di impostare appunto la propria password.

4.5.2 Login

Come si può intuire il controller *Login* è abbastanza simile al controller *AdminLogin*. Come si può facilmente intuire la variabile *admin* viene impostata a false e il titolo della pagina viene tramutato in *Accedi con l'account della sede*.

```
//Imposta che non si tratta di un utente amministratore
$admin = false;

//Nel caso che la variabile $_SESSION['error'] non sia impostata la imposta
//con il valore null
if (!isset($_SESSION['error'])) {
    $_SESSION['error'] = null;
}

//Titolo della pagina
$title = "Accedi con l'account della sede";
```

Figura 36 -- login Login

Ovviamente la differenza oltre a questa con *AdminLogin* è che al posto che controllare se l'utente esista all'interno del database viene controllato se esso esiste all'interno dell'AD dello stabile, sempre tramite la classe *User*.

4.5.3 Main

Una volta eseguito l'accesso tutti gli utenti vengono gestiti dal medesimo controller, ovvero *Main*. All'interno di questo controller sono presenti la maggior parte dei metodi. Siccome in questa classe vengono richiamate più volte diverse views ho deciso di fare due metodi che si occupano unicamente di fare i require necessari.

```
private function preload()
{
    require 'application/views/header.php';
    require 'application/views/Helpers/sidebar.php';
    require 'application/views/Helpers/user_info.php';
}
```

Figura 37 -- *preload*

Il primo, chiamato *preload*, si occupa di caricare l'header, la barra laterale e la barra in alto con le informazioni dell'utente. Grazie a questo metodo riesco a risparmiare tre righe di codice molto semplicemente. D'altra parte invece si trova il metodo *postload*, che è stato creato per coerenza più che per risparmio. In questo modo in ogni metodo del controller si richiama il metodo *preload*, poi si carica la pagina interessata ed

```
private function postload()
{
    require 'application/views/footer.php';
}
```

Figura 38 -- *postload*

infine si chiama il metodo *postload*.

Ogni metodo all'interno del controller si occupa di caricare le pagine principali dell'applicativo, quindi si parla delle pagine: log, Gestione amministratori, Gestione aule, Gestione stabili, Statistiche giornaliere, Statistiche mensili e ricerca.

In ogni metodo viene controllato che l'utente possa accedere a quella funzionalità, ad esempio un docente non può utilizzare la funzionalità di gestione degli amministratori. Questo viene fatto precauzionalmente togliendo l'opzione quando l'utente con permessi più bassi effettua il login, ma in caso che conosca il link per raggiungere la pagina viene effettuato anche un controllo sul tipo di utente che ha effettuato l'accesso.

```
if ($this->checkPermission("admin")) {
```

Figura 39 -- Controllo permessi

Ad esempio quella mostrata sopra è il controllo che l'utente sia amministratore nella pagina di gestione degli amministratori.

```
private function checkPermission($type)
{
    if ($_SESSION['type'] === $type) {
        return true;
    } else {
        return false;
    }
}
```

Figura 40 -- *checkPermission* main

Il metodo *checkPermission* si occupa semplicemente di confrontare il tipo richiesto dal controllo con quello salvato nella sessione.

4.5.4 AdminController

Questo controller è quello che si occupa di fare tutte le operazioni di un amministratore. Prima tra tutte l'aggiunta di un amministratore

```
public function addAdmin(){
    $username = DataChecker::checkInput($_POST['username']);
    $building = DataChecker::checkInput($_POST['building']);
    User::addUser($username, $building);
}
```

Figura 41 -- addAdmin AdminController

Per fare ciò il metodo ottiene l'input inserito all'interno dei campi di testo con id *username* e *building*, li controlla tramite la classe *DataChecker* ed infine chiama la classe sopra citata (*User*) per aggiungere quel determinato utente al database. Come si può intuire quindi la password viene già criptata tramite hash prima di inserirla nel database, per evitare di avere dati in chiaro al suo interno.

Successivamente si trova l'aggiunta delle classi. Questo viene fatto sempre tramite i soliti input e una volta eseguita l'azione viene ricaricata la pagina di gestione delle classi tramite il metodo *header*.

```
public function addClassroom()
{
    //Ottenimento del nome dell'aula
    $name = $_POST['name'];

    //Istanziamento dell'aula e creazione
    $classroom = new Classroom($name, $_SESSION['building']);
    $classroom->create();

    //Reindirizzamento alla visualizzazione delle aule
    header('location:' . URL . 'main/classroomsManagement');
}
```

Figura 42 -- addClassroom AdminController

Inoltre si trova anche l'aggiunta degli stabili, questo metodo è estremamente simili a quelli mostrati precedentemente; l'unica cosa che cambia sono gli input che vengono richiesti, che corrispondono alle colonne della tabella *building*.

Per ogni metodo che aggiunge qualcosa c'è anche il rispettivo che lo elimina, ad esempio nell'immagine mostrata sotto si può vedere l'eliminazione di una classe.

```
//Ottenimento del nome dell'aula da eliminare e dello stabile nel quale si colloca
$name = $_POST['name'];
$building = $_SESSION['building'];

//Istanziamento della classe e rimozione
$classroom = new Classroom($name, $building);
$classroom->remove();

//Reindirizzamento alla pagina di visualizzazione delle aule
header("location:" . URL . "main/classroomsManagement");
```

Figura 43 -- eliminazione aule

Come si può vedere viene richiesto il nome dell'aula e il rispettivo stabile per eliminare l'aula. L'eliminazione degli amministratori include anche un passaggio aggiuntivo. Esso è il controllo che l'amministratore eliminato sia il medesimo che quello loggato. In questo caso l'applicativo consente l'eliminazione ma esegue il logout automatico da esso.

```
//In caso che un amministratore si rimuove da solo torna alla pagina di login (effettua il logout)
if ($login == $_SESSION['username']) {
    header("location:" . URL . "login");
}else{
    header("location:" . URL . "main/adminsManagement");
}
```

Figura 44 -- Logout automatico

Per fare ciò si legge l'username dell'utente loggato e in caso che sia il medesimo di quello che si vuole eliminare l'utente viene riportato al login.

L'eliminazione degli amministratori non è l'unico metodo di eliminazione leggermente differente da quello delle aule ma anche l'eliminazione degli stabili implementa un controllo aggiuntivo. Il controllo citato è la richiesta della password di amministratore, questo permette di non eliminare accidentalmente uno stabile.

```
//Ottenimento della password da controllare, serve quest'ultima per eliminare uno stabile
$password = DataChecker::checkPassword($_POST['password']);

//Istanziamento dell'utente che vuole eliminare lo stabilimento
$user = new User($_SESSION['username'], $password, $_SESSION['building']);

//In caso che il login sia valido lo stabile viene eliminato
if(intval($user->loginAdmin()) == 1){
    $building = new Building($domain);
    $building->remove();
}
```

Figura 45 -- Eliminazione stabile

In caso che la password inserita sia sbagliata l'utente viene riportato automaticamente alla gestione degli stabili.

4.5.5 teacherController

Il controller *teacherController* è stato creato per includere in esso tutte le funzionalità che deve avere un docente, ovvero la ricerca. Come prima cosa viene richiesto l'inizio della lezione, la durata di essa e il giorno in cui si è svolta. In caso che tutti i dati siano veritieri vengono impostate le informazioni all'interno delle variabili di sessione e successivamente il controller *main* si occupa di ottenere dei risultati in base ai filtri impostati nella ricerca.

4.6 Views

Le views sono i file contenente per lo più codice html che contengono quello che vedono gli utenti. Generalmente si ha una view per ogni pagina che l'utente visualizza, escludendo però tutti i menu.

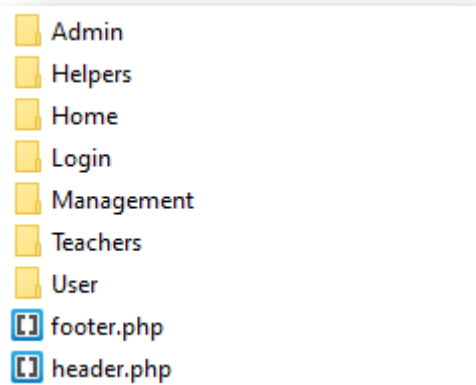


Figura 46 -- Struttura

La struttura delle cartelle, mostrata nell'immagine sovrastante, è divisa in base a chi utilizza quella determinata cartella. Ad esempio all'interno della cartella *Admin* sono presenti tutte le pagine che deve vedere solo l'amministratore, questo non significa che con le cartelle definisco anche i permessi ma che organizzo meglio la struttura generale. Infine le view che vengono chiamate più volte (come il menu laterale) ho deciso di fare una cartella denominata *Helpers* in modo di non dover buttare questi file nella root e compromettere tutta l'organizzazione.

4.6.1 Header

Header è una view che viene richiamata sempre, da ogni pagina all'interno dell'applicativo. Essa rappresenta la parte in comune di tutte le pagine, ovvero l'head. Per evitare di ricopiare il codice ogni volta in ogni view ho creato questa che viene richiamata ogni volta, in questo modo si evita moltissimo codice recursivo.

```
<meta charset="utf-8" />
<meta name="author" content="Matteo Arena"/>
<meta name="search-date" content="2021-02-26"/>
<meta name="version" content="1.1.1"/>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Figura 47 -- meta

I meta sono le informazioni sulla pagina, in questi campi ho inserito il mio nome (che sarebbe quello dell'autore), la data nella quale è stato messo online l'applicativo e la versione di esso. L'ultima riga invece serve unicamente a rendere responsive l'applicativo.

Le righe successive invece servono unicamente ad includere tutti i css che serviranno alle pagine che richiamano l'header, così che lo stile sia sempre il medesimo su tutto l'applicativo.

4.6.2 Footer

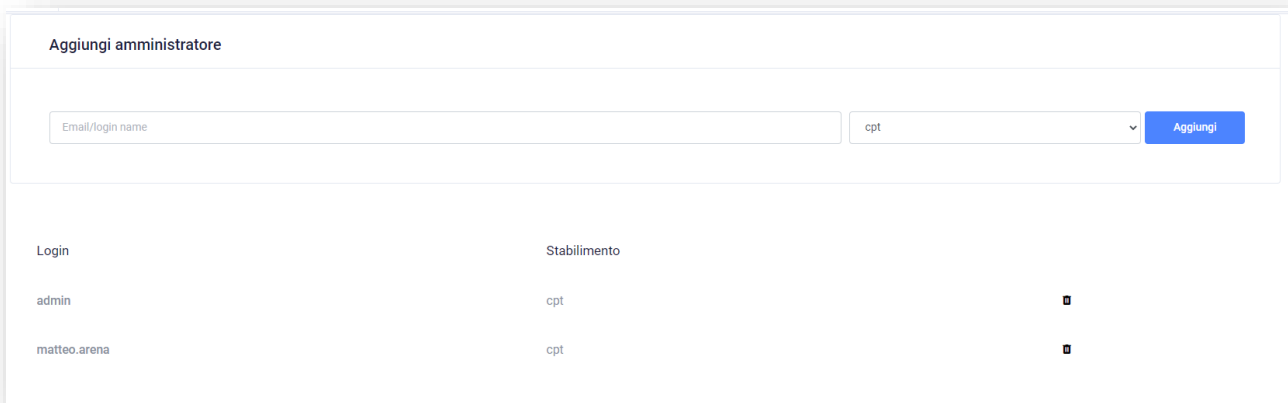
Al contrario dell'header il footer viene richiamato come ultima cosa, all'interno di esso sono presenti tutti i file javascript per la parte dinamica dell'applicativo (ad esempio i grafici). Infine nelle ultime due righe si occupa di chiudere i tag che sono stati aperti all'interno delle pagine che lo richiamano (*body* ed *html*).

4.6.3 Admin

Come detto nella spiegazione della struttura nella cartella Admin sono presenti tutti i file che si occupano di mostrare all'utente le pagine dell'amministratore come la gestione delle aule, degli stabili, ... Questo però non comporta che grazie alle cartelle viene fatto il controllo di chi accede al file, quello è gestito all'interno del codice php in base ai vari utenti e quale pagina cercano di raggiungere.

4.6.3.1 Admins_management

Come può far intuire il nome questa view rappresenta la gestione degli amministratori. Nell'immagine sottostante è riportata la resa grafica di tutto il codice all'interno del file.



Login	Stabilimento
admin	cpt
matteo.arena	cpt

Figura 48 -- Gestione amministratori

Le informazioni sugli amministratori sono salvate in una variabile all'interno del controller che si occupa di richiamare questa view e tramite un ciclo all'interno dello stesso stampo tutto. Tutte queste informazioni però sono interne ad un form che permette, in caso che si preme il cestino, di eliminare l'amministratore. Questo non viene fatto senza nessuna conferma ma una volta premuto il bottone si attiva un alert che avvisa l'utente e gli richiede la conferma di quello che sta facendo.

Particolare anche il menu in alto (chiamato *Aggiungi amministratore*), questo richiede il nome dell'utente da aggiungere e lo stabilimento nel quale è compreso. Gli stabili però non sono digitabili dall'utente ma sono bensì presi direttamente dal database, per evitare problemi con nomenclature sbagliate degli stabili.

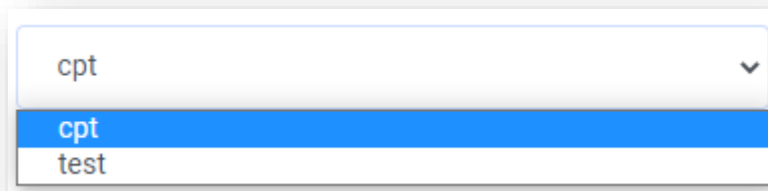


Figura 49 -- Selezione stabile

4.6.3.2 Classroom_management

Come per gli amministratori è presente anche una view per gestire tutte le aule. L'unica differenza tra le pagine è che in questa è presente, come informazioni, il nome dell'aula e lo stabile nel quale è contenuta. Oltre a questo nella creazione di un'aula non è possibile specificare lo stabile nel quale essa è contenuta. Bisogna per forza essere collegati con un account amministratore dello stabile per permettere di aggiungere la classe in un determinato stabile.

4.6.3.3 Buildings_management

Aggiungi stabilimento

Città

Nazione

Via

CAP

Dominio

Stringa di connessione LDAP

DN di base

Proprietà AD da leggere per il login degli utenti

OU contenente i professori

OU contenente il segretariato

OU contenente la direzione

Aggiungi



Dominio	Luogo	Azioni
cpt	Canobbio, Switzerland	 

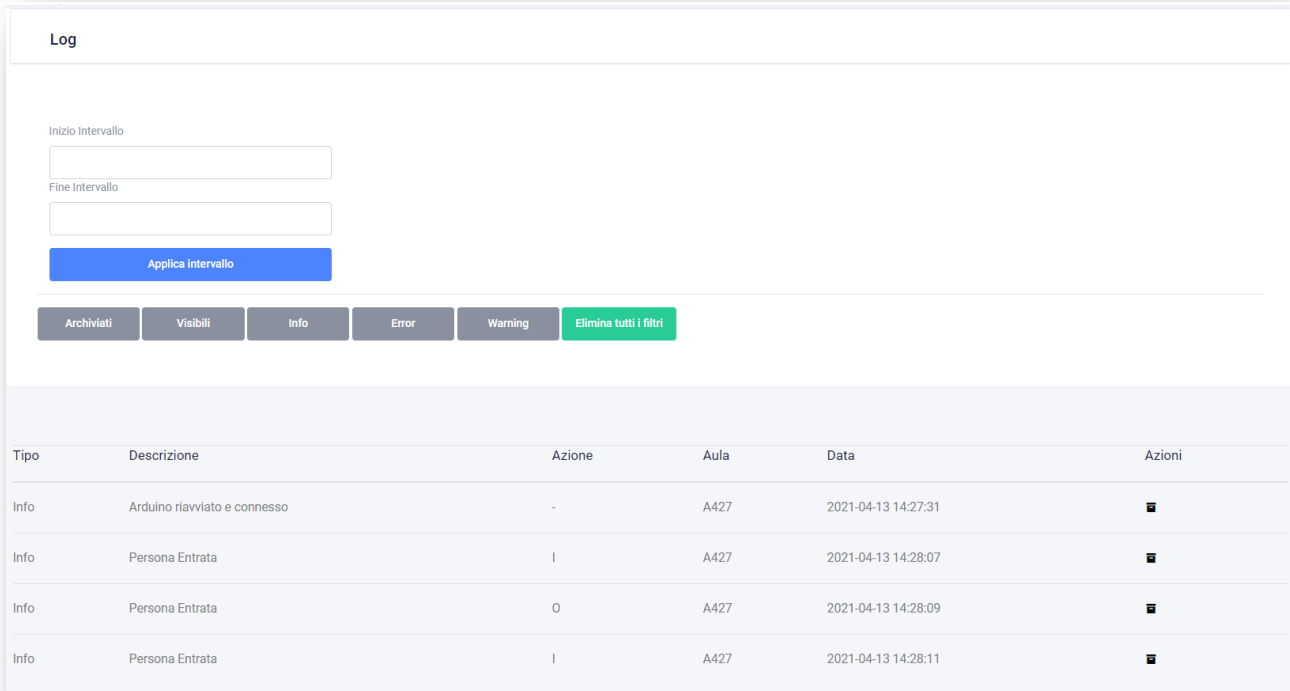
Figura 50 -- Gestione stabilimenti

Per non avere una pagina troppo piena ho deciso di inserire come informazioni visibili direttamente in questa pagina unicamente il nome dello stabile e la città nella quale si trova, con tra parentesi la nazione. Per visualizzare tutte le informazioni bisogna obbligatoriamente premere la *i* di informazioni a fianco al cestino. Cliccando su di essa vengono riportate tutte le informazioni sullo stabile specifico. Come si può ben notare sono anche presenti più voci nell'inserimento di un nuovo stabile.

Un'altra differenza sostanziale con le altre schermata è che in questa in caso di eliminazione di uno stabilimento richiedere oltre che la conferma la password dell'utente amministratore siccome rimuove tutte le informazioni dal database.

4.6.3.4 Logs

Come ultima pagina dedicata all'amministratore si trovano i log. In questa pagina sono presenti tutte le informazioni riguardanti l'Arduino, ad esempio se si è collegato alla banca dati o se sono entrate/uscite delle persone.







Tipo	Descrizione	Azione	Aula	Data	Azioni
Info	Arduino riavviato e connesso	-	A427	2021-04-13 14:27:31	
Info	Persona Entrata	I	A427	2021-04-13 14:28:07	
Info	Persona Entrata	O	A427	2021-04-13 14:28:09	
Info	Persona Entrata	I	A427	2021-04-13 14:28:11	

Figura 51 -- Logs

Come si può notare nell'immagine sovrastante nella schermata dei log non è possibile, giustamente, aggiungere un nuovo log ma unicamente navigare tra quelli generati dall'Arduino. La view permette anche, grazie alla collaborazione con il controller, di applicare dei filtri e l'archiviazione dei log.

4.6.4 Management

Come per gli amministratori questa è la cartella dedicata ad un tipo di utente specifico, più precisamente alla direzione e alla segreteria. All'interno sono presenti 3 file che si occupano tutti di mostrare i grafici delle presenze nelle aule.

4.6.4.1 todayStats

Questo è il file che si occupa di mostrare il grafico con tutte le entrate del giorno odierno. Per fare ciò ottiene tutte le entrate che sono state effettuate in un intervallo di 10 minuti dal suono della campanella (10 prima e 10 dopo). In questo modo si evita che se una persona entra nel bel mezzo della lezione venga contata come persona entrata e quindi come allievo presente.

```
new Chart(document.getElementById("canvas"), {
  "type": "bar",
  "data": {
    "labels": ["Allievi Totali", "Allievi segnati in GAGI", "Allievi contati"],
    "datasets": [{
      "label": "Dati odierni",
      "data": [
        <?php echo $gagi->getTotalUserEntered('curdate()'); ?>,
        <?php echo $gagi->getEnteredUsers('curdate()'); ?>,
        <?php echo $stats->getEnteredUsers('curdate()'); ?>
      ],
      "fill": false,
      "backgroundColor": ["rgba(255, 159, 64, 0.2)", "rgba(75, 192, 192, 0.2)", "rgba(153, 102, 255, 0.2)", "rgba(201, 203, 207, 0.2)"],
      "borderColor": ["rgb(255, 159, 64)", "rgb(75, 192, 192)", "rgb(153, 102, 255)", "rgb(201, 203, 207)"],
      "borderWidth": 1
    }]
  },
  "options": {
    "scales": {
      "yAxes": [{
        "ticks": {
          "beginAtZero": true
        }
      }]
    }
  }
});
```

Figura 53 -- Script statistiche

Nell'immagine sopra è mostrato come viene riportato il grafico giornaliero (mostrato nell'immagine sottostante), come si può notare è completamente gestito tramite javascript siccome utilizza la libreria CharJS. Come prima cosa salvo in *labels* il nome delle colonne del grafico a barre e in *datasets* inserisco i vari valori, ottenuti tramite php. Infine nelle ultime righe imposto tutte le configurazioni estetiche del grafico come il colore del background delle barre, il colore dei bordi e la larghezza dei bordi. Il risultato ottenuto con questo script è mostrato nell'immagine sottostante.

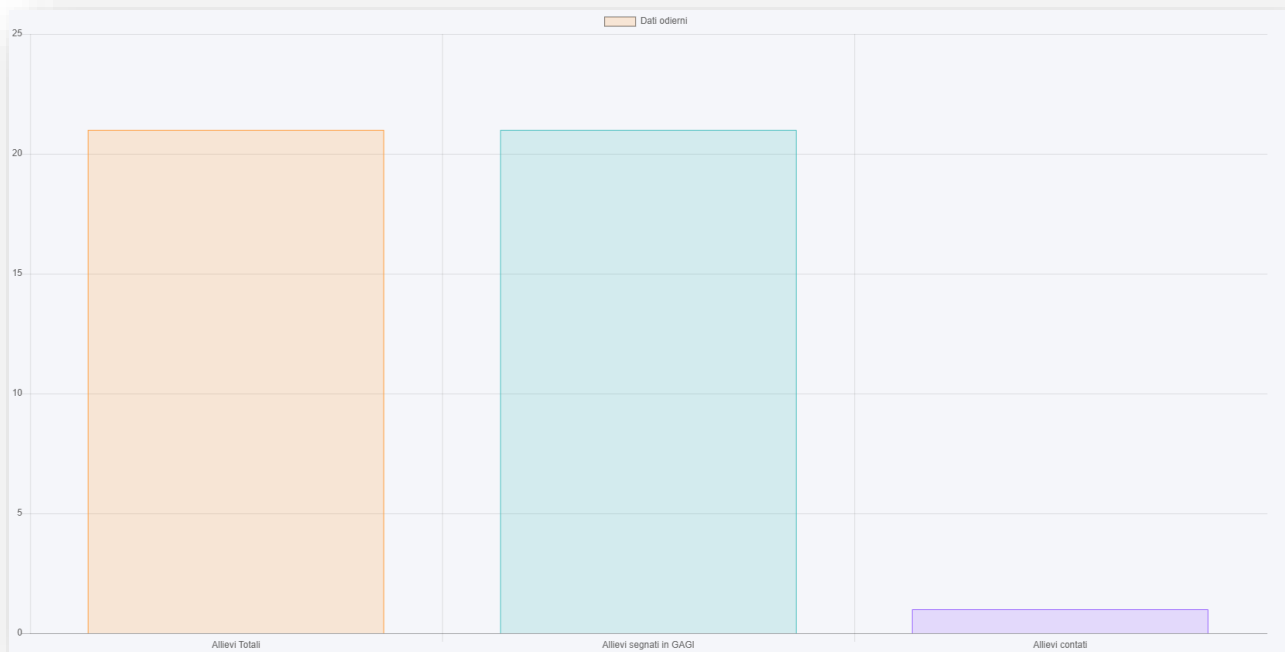


Figura 52 -- Resa del grafico

4.6.4.2 totalStats

Come dice il nome questo file mostra tutte le statistiche mensili, la differenza con il file precedente è che qui in caso di uscita viene decrementato il numero di persone in aula (in un certo lasso di tempo dall'inizio della lezione). Il funzionamento del grafico è molto simile a quello delle statistiche giornaliere solo che cambia il tipo del grafico. Le informazioni infatti si danno nel medesimo modo (ovviamente trattandosi di statistiche mensile i dati saranno di più) ma cambia il risultato finale.

4.6.4.3 Stats

Stats riporta esattamente lo stesso codice del grafico giornaliero solo che si occupa di portare la statistica di un'aula nelle ultime 3 settimane, includendo quella corrente. Questa view viene richiamata all'interno della pagina dei docenti quando richiedono la statistica di un'aula quindi non c'entra propriamente con il gruppo management ma l'ho messo qui dentro siccome si tratta sempre di statistica.

4.6.5 Teachers

All'interno di questa cartella sono presenti tutte le pagine che servono ai docenti. Ovviamente tutte queste pagine sono condivise anche con gli utenti che hanno permessi più elevati di loro, ovvero amministratori e direzione.

Come prima cosa viene richiamata la view *filters* nella quale si devono impostare i filtri per la lezione della quale si vogliono avere informazioni. Successivamente (se i dati sono corretti) vengono riportate tutte le informazioni su quell'aula. Queste informazioni sono racchiuse nell'altro file, ovvero *data*. Questo si occupa di richiedere il numero di allievi in aula, la percentuale di presenze e un grafico delle ultime 3 settimane di quella lezione in quell'aula.

4.6.6 Helpers

Gli helpers sono quelle views che si occupano di visualizzare graficamente qualcosa che viene richiamato più volte come il menu laterale. Infatti all'interno di questa cartella si trovano unicamente il menu laterale (che si adatta in base al tipo di utente che fa il login) e la pagina di informazioni sull'utente. Per mostrare più o meno voci all'interno del menu laterale faccio semplicemente un controllo del tipo di utente, se si tratta di un amministratore mostro determinate cose, se è della direzione di meno e se è un docente ancora meno.

Dall'altra parte invece si trova il menu in alto contenente il nome dell'utente con un menu che permette il logout e la possibilità di andare alle informazioni di esso.

4.6.7 Home

Nella cartella con le varie home l'unica cosa che cambia è il modo in cui viene denominato l'utente.

4.6.8 Login

All'interno di questa cartella, come si può facilmente intuire, sono contenute tutte le pagine che si occupano del login dell'utente. Essendo quella con AD e quella degli amministratori la medesima entrambe sono racchiuse all'interno del file *login.php* che viene modificato (nel titolo) in base al tipo di login.

Oltre al file che si occupa del login è presente anche un altro, chiamato *registerPassword*, che permette di registrare la propria password. Questa pagina viene richiamata, come detto in precedenza, quando viene creato un nuovo utente amministratore ed esso cerca di fare il login.

4.6.9 User

Come ultima cartella con le views si trova User, contenente unicamente un file che mostra le informazioni dell'utente, quali il suo nome e tutti gli stabili nel quale esso lavora.

4.7 Funzionamento

4.7.1 Login

Il funzionamento generale dell'applicativo è molto semplice ed intuitivo, per iniziare bisogna recarsi sull'applicativo web aprendo il proprio browser internet e digitando sulla barra di ricerca il seguente indirizzo:

- <https://saminfo.ch/i17aremat/EntrateUscite>

Digitato l'indirizzo ci si ritroverà davanti ad un sito web molto simile a quello mostrato nell'immagine.

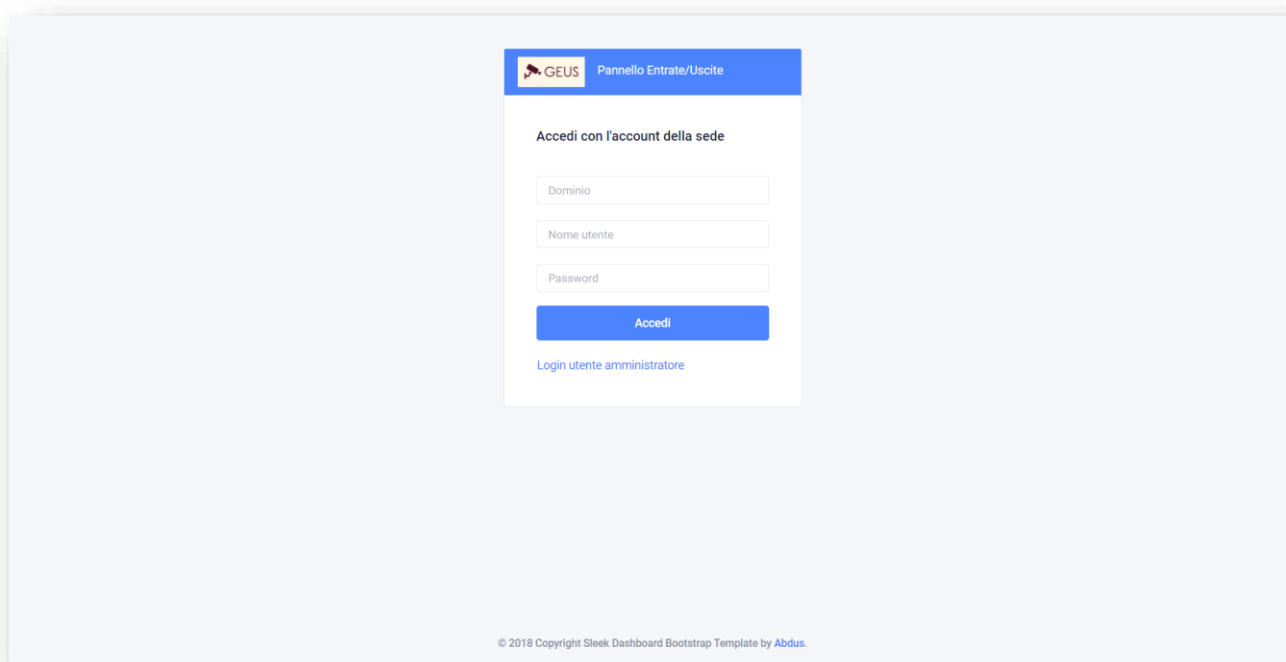


Figura 54 -- Schermata di login

Qua bisogna sapere che tipo di utente si ha a disposizione, nel caso dell'esempio siamo utenti amministratori (per poter mostrare tutte le possibilità dell'applicativo), quindi bisognerà premere su *Login utente amministratore*. In caso che non si sia amministratori si può tranquillamente inserire le proprie credenziali e ci si ritroverà davanti alla propria pagina.

La pagina di login amministratore non differisce dalla pagina di login normale ad eccezione che per il titolo che diventa *accedi con account amministratore* e la scritta per passare al login amministratore diventa la scritta per passare al login normale.

Eseguito il login ci si ritroverà in una schermata bianca con a lato tutte le voci alle quali possiamo accedere e in alto le informazioni sul nostro utente.

4.7.2 Permessi degli utenti

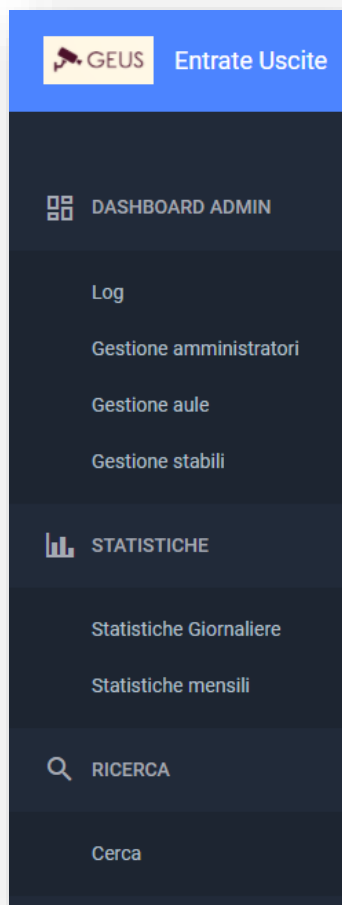


Figura 55 -- sidebar amministratore

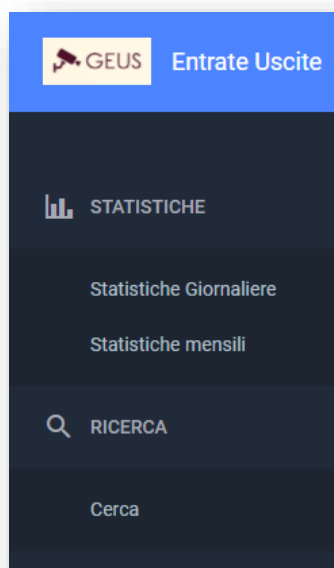


Figura 57 -- sidebar direzione



Figura 56 -- sidebar docenti

Quelle mostrate nelle immagini sovrastanti sono le varie barre laterali che si ritrovano davanti i vari utenti quando eseguono l'accesso. Come si può facilmente notare il menu è sempre il medesimo e in base al tipo di utente che si aggiungono o si rimuovono delle impostazioni. Ad esempio l'amministratore, mostrato nell'immagine di estrema destra, ha la possibilità di accedere a tutte le impostazioni. La direzione, in mezzo, ha la possibilità di visualizzare le statistiche e fare la ricerca in base alle aule ed infine i docenti possono unicamente fare delle ricerche in base alla classe.

4.7.3 Log (solo admin)

La gestione dei log, dedicata unicamente agli amministratori, è gestita tramite l'omonima impostazione. Per aprire questa pagina bisogna effettuare il login nell'applicativo e cliccare sulla voce *Log* sotto al menu *Dashboard Admin*. La pagina sarà molto simile a quella mostrata nell'immagine sottostante.

Tipo	Descrizione	Azione	Aula	Data	Azioni
Info	Arduino riavviato e connesso	-	A427	2021-04-13 14:27:31	
Info	Persona Entrata	I	A427	2021-04-13 14:28:07	
Info	Persona Entrata	O	A427	2021-04-13 14:28:09	
Info	Persona Entrata	I	A427	2021-04-13 14:28:11	

Figura 58 -- Pagina log

La pagina, come si può vedere, si separa in due parti. La prima parte contiene tutti i filtri applicabili (anche cumulabili) mentre la seconda racchiude la visualizzazione dei log e la loro gestione.

4.7.3.1 Filtri

I filtri permettono all'utente di poter filtrare i risultati che vuole vedere senza doversi districare tra molti dati inutili. Questo viene fatto nell'applicativo tramite due tipi di filtri.

4.7.3.1.1 Filtro temporale

Ci sono due tipi di filtri, i primi sono quelli di tempo. Per applicarli bisogna digitare nei due campi di testo l'intervallo di date delle quali si vuole visualizzare i dati. Si è voluto scegliere di includere la prima data ed escludere la seconda, questo significa che se nell'esempio si vogliono vedere tutti i dati dal *12-04-2021* al *13-04-2021* non verrà mostrato alcun risultato. Per far sì che venga incluso anche il 13 sarà obbligatorio fare sino al *14-03-2021*. Impostato l'intervallo al di sopra dei risultati verrà riportata una scritta per ricordare che il filtro è attivo.

Si Stanno Visualizzando I Dati Da 13-04-2021 A 14-04-2021

Figura 59 -- Info intervallo date

Se si vuole eliminare il filtro, come per tutti gli altri filtri, bisogna premere il bottone verde con scritto *Elimina tutti i filtri*. Ovviamente il filtro con l'intervallo di date è tranquillamente usabile combinandolo con gli altri filtri.

4.7.3.1.2 Filtro di tipo

Questo filtro permette di visualizzare i log in base al loro tipo. Ad esempio si può scegliere di vedere unicamente le informazioni o gli errori o anche entrambi insieme. Oltre a questo si possono anche visualizzare i log archiviati e quelli meno. Essendo i filtri combinabili (tutti i tipi) per rimuoverli tutti ho implementato un bottone che permette di eliminare tutti i filtri applicati precedentemente. Questo bottone è verde ed è a fianco ai filtri di tipo, si chiama *Elimina tutti i filtri*.

4.7.3.2 Archiviazione

L'archiviazione consiste nel nascondimento di un'informazione nei grafici, questo è molto utile con le entrate e le uscite che non si vogliono calcolare dei grafici. Facendo un esempio nell'immagine sottostante sono presenti 2 entrate il giorno odierno (14.04.2021) ed 1 uscita; tutti questi dati sono visibili.

Tipo	Descrizione	Azione	Aula	Data	Azioni
Info	Arduino riavviato e connesso	-	A427	2021-04-13 14:27:31	
Info	Persona Entrata	I	A427	2021-04-14 14:28:07	
Info	Persona Entrata	O	A427	2021-04-14 14:28:09	
Info	Persona Entrata	I	A427	2021-04-14 14:28:11	

Figura 60 -- info persone entrate/uscite

Se ci si reca sulle statistiche giornaliere il risultato sono due entrate ed un'uscita, quindi all'interno dell'aula è presente unicamente una persona.

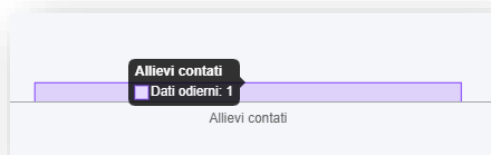


Figura 61 -- Allievi contati senza archiviare

Invece se si archivia il dato dell'uscita, quello delle 14:28:09, tramite l'apposito bottone e si torna sulla stessa schermata il risultato sarà il seguente.

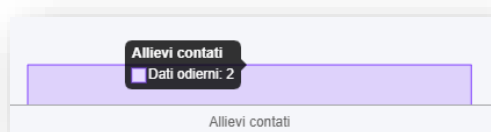


Figura 62 -- Allievi contati archiviando

Per applicare un'archiviazione bisogna andare nella pagina con i log e premere sull'icona a forma di cestino per la carta, questa si sostituirà in automatico con un occhio, questo significherà che se si preme di nuovo il dato tornerà visibile.

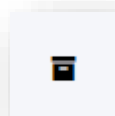


Figura 63 -- Bottone per archiviare

4.7.4 Gestione amministratori (solo admin)

La gestione degli amministratori, possibile unicamente da questi ultimi, è racchiusa in una pagina sotto la *Dashboard Admin* denominata *Gestione amministratori*. Aprendola ci si ritrova dinanzi alla seguente

Figura 64 -- Schermata gestione amministratori

schermata.

Anche per questa pagina ci sono due parti principali. La prima è l'aggiunta di nuovi amministratori mentre la seconda la gestione di quelli già esistenti.

4.7.4.1 Creazione amministratori

Per creare un amministratore bisogna digitare il nome utente all'interno del campo chiamato *Email/login name* e scegliere a destra il nome di uno stabile già esistente. In caso che si sceglie un altro stabile non verrà visualizzato l'amministratore siccome nella visualizzazione vengono riportati unicamente gli amministratori dello stabile interessato. Come esempio decido di fare un utente denominato *john.doe* ed inserirlo nello stabile corrente, ovvero *cpt*.

Figura 65 -- Creazione utenti

Cliccando su *aggiungi* l'utente verrà automaticamente creato, aggiungendolo pure alla lista, ma come si potrà notare non verrà richiesto di inserire alcuna password. Questo viene fatto per non fornire la password ad altri amministratori, per inserire la password l'utente dovrà eseguire il suo primo accesso. Quindi ci si reca sul login amministratore, si inserisce il nome dell'utente appena creato, il dominio corretto e non si inserisce alcuna password.

Figura 66 -- Registrazione password

In automatico verrà richiesto di inserire la nuova password per l'utente.

Inserita la nuova password, ed anche la sua conferma, si verrà riportati al login. Ora si potrà accedere con il nuovo utente senza problemi inserendo la sua password. Per essere sicuri che l'account con il quale si ha fatto il collegamento sia quello corretto bisogna controllare se il nome utente in alto corrisponde con l'utente appena creato.

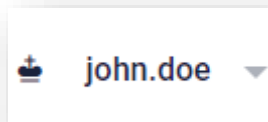


Figura 67 -- Controllo nome utente

4.7.4.2 Eliminazione amministratori

Gli amministratori vengono rimossi tramite l'apposito bottone a forma di cestino a fianco al loro nome. Cliccando sul bottone l'utente non verrà eliminato immediatamente ma verrà richiesta la conferma con un avviso che avvertirà l'utente dell'azione che sta eseguendo e le conseguenze che avrà. In caso che si voglia eliminare il proprio profilo verrà eseguita la disconnessione automatica dall'applicativo.

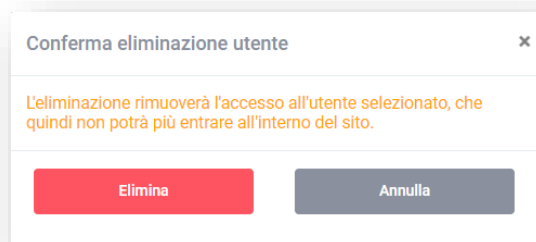


Figura 68 – Avviso eliminazione utenti

4.7.5 Gestione aule (solo amministratori)

Le aule vengono gestite in un modo estremamente simile alla gestione degli amministratori. Questa schermata è sempre sotto la *Dashboard Admin* cliccando il link *Gestione aule*. Qui però a differenza della gestione degli amministratori non viene richiesto quando si crea una classe di inserire lo stabile perché le aule si possono creare unicamente per il proprio stabile (infatti il nome di esso verrà preso in automatico).

4.7.5.1 Creazione aula

La creazione di un'aula viene fatta in maniera estremamente semplice nella parte superiore della schermata inserendo molto semplicemente il nome dell'aula da creare.

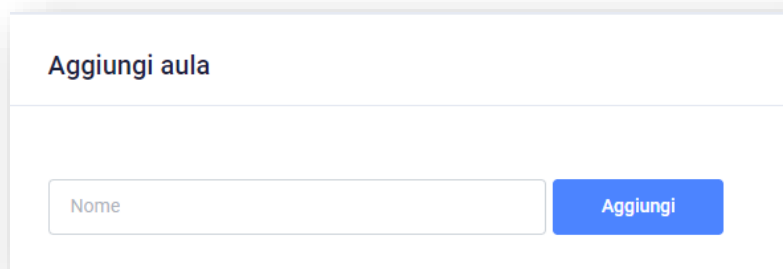


Figura 69 -- Creazione aula

4.7.5.2 Eliminazione aula

Come per gli amministratori l'eliminazione dell'aula viene fatta tramite il bottone a forma di cestino che si trova sulla medesima riga dell'aula interessata. Esattamente come per gli amministratori viene mostrato un avviso con le conseguenze dell'azione che l'utente sta facendo. Per confermare l'eliminazione bisogna quindi premere sul tasto rosso con scritto *Elimina*.



Figura 70 -- Avviso eliminazione aula

4.7.6 Gestione stabili (solo amministratori)

La gestione degli stabili, molto simile a tutte le altre gestioni (amministratori e classi), Permette di eliminare o aggiungere un nuovo stabile. La differenza sostanziale è che in questo caso è difficile mostrare tutte le informazioni sulla pagina siccome sono veramente tante, quindi si è deciso di mostrare unicamente il nome dello stabile e il luogo nel quale si trova. Nel caso che si voglia visualizzare tutte le informazioni dello stabile bisogna cliccare sulla *i* di informazioni a fianco al cestino, in questo modo verrà mostrata la seguente schermata (con un avviso).

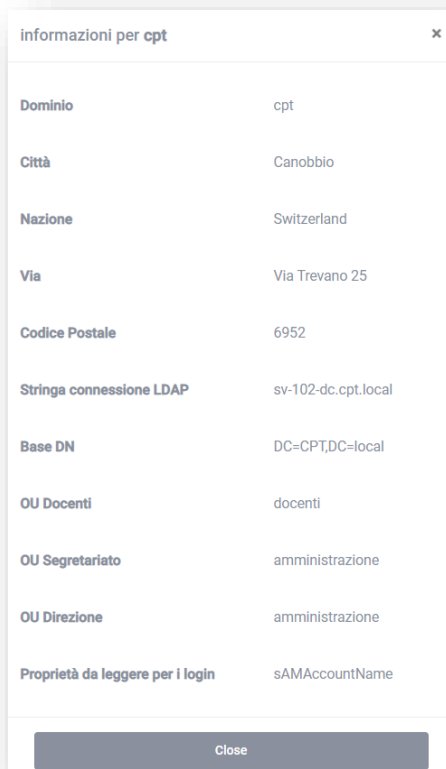


Figura 71 -- Informazioni complete stabile

4.7.6.1 Eliminazione stabili

Siccome gli stabili sono ancora più delicati degli amministratori e delle aule ho pensato di inserire un'ulteriore sicurezza prima dell'eliminazione di essi. Per eliminarli infatti bisogna, come per amministratori e classi, cliccare sul cestino ma al posto di dover dare unicamente la conferma bisognerà anche inserire la password dell'amministratore.

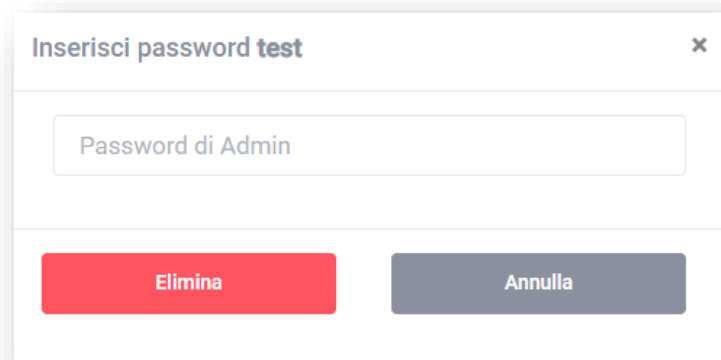


Figura 72 -- conferma password eliminazione stabili

In caso che l'utente inserisca la password corretta e preme elimina lo stabile verrà eliminato altrimenti l'eliminazione non andrà a buon fine.

4.7.6.2 Creazione degli stabili

Gli stabili sono molto facili da creare, infatti bisogna unicamente completare i vari input ed esso verrà creato.



Figura 73 -- input creazione stabili

I seguenti dati corrispondono a:

- **Città:** Città nella quale è collocato lo stabile, nel caso della scuola *Canobbio*.
- **Nazione:** Nazione nella quale si trova lo stabile, nel caso della scuola *Svizzera*.
- **Via:** Via dove si trova lo stabile, nel caso della scuola *Via Trevano 25*.
- **CAP:** Codice postale dello stabile, nel caso della scuola *6952*.
- **Dominio:** Dominio con il quale ci si collega all'AD, nel caso della scuola *cpt*.
- **Stringa di connessione LDAP:** Stringa con la quale si effettua la connessione tramite LDAP all'AD dello stabile, nel caso della scuola *sv-102-dc.cpt.local*.
- **DN di base:** Dn di base dalla quale ottenere poi le varie OU, nel caso della scuola *DC=CPT,DC=local*
- **Proprietà AD da leggere per il login degli utenti:** La proprietà da leggere per consentire gli accessi, solitamente questa informazione si chiama *samaccountname*.
- **OU contenente i professori:** OU nella quale sono presenti tutti i docenti.
- **OU contenente il segretariato:** OU nella quale sono presenti tutti i componenti del segretariato.
- **OU contenente la direzione:** OU nella quale sono presenti tutti i membri della direzione

Inseriti i vari dati lo stabile verrà creato e subito aggiunto alla lista sottostante.

4.7.7 Statistiche giornaliere (amministratori, direzione e segretariato)

All'interno di questa schermata sono presenti i dati di tutti gli allievi che sono entrati in classe nel giorno odierno.

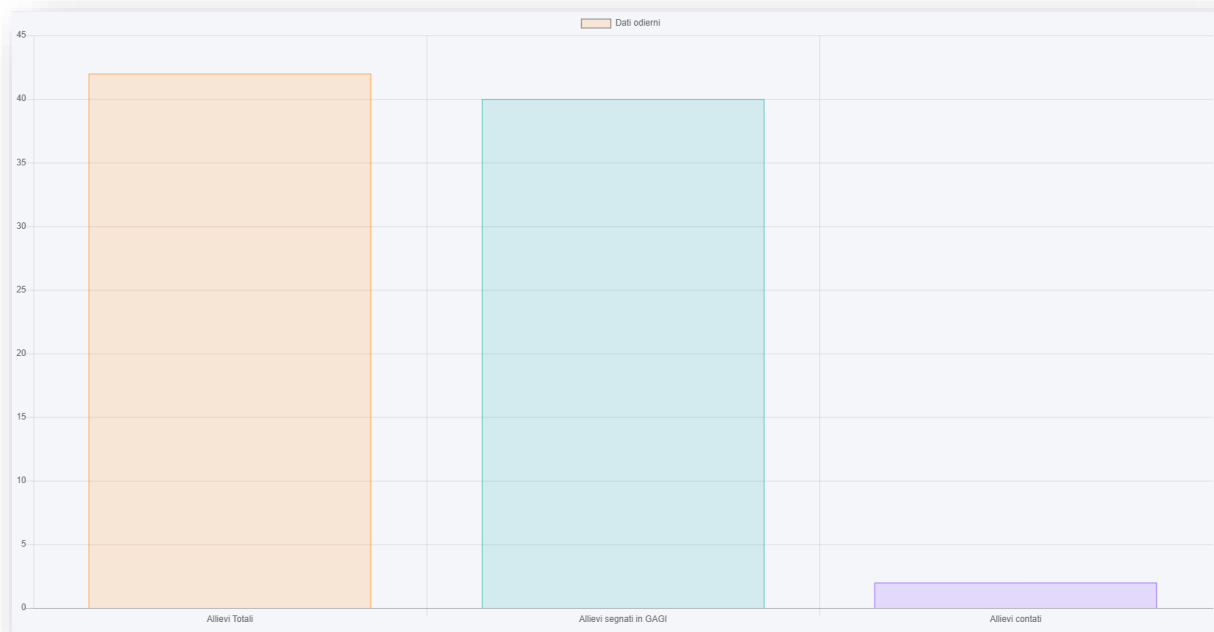


Figura 74 -- Pagina statistiche odierne

Nel primo dato (a sinistra) sono riportati gli allievi totali che dovrebbero essere presenti durante la giornata nelle aule con predisposto il sistema di conteggio. Nell'esempio si visualizza un'aula che durante la giornata ospita due classi da 21 allievi l'una dunque il numero totale di allievi che dovrebbe essere presente è 42 (questo dato viene rappresentato in arancione). Nel secondo dato sono riportati il numero di allievi segnati dai professori all'interno dell'applicativo GAGI, dedicato alla gestione delle assenze e dei ritardi. Questo dato è segnato in azzurro e come si può notare risulta che i docenti abbiano segnato 40 allievi presenti in totale. Infine in viola si trova il numero di allievi che sono stati rilevati dal sensore, nel caso riportato si tratta di dati di test quindi il dato non è quello corretto (sarebbero unicamente 2 allievi).

In questa schermata non si può fare nulla a parte visualizzare graficamente tutte le informazioni captate dai sensori.

4.7.8 Statistiche mensili (amministratori, direzione e segretariato)

Come per le statistiche giornaliere anche le statistiche mensili permette unicamente di essere visualizzata, senza poter effettuare modifiche ai dati.



Figura 75 -- Pagine statistiche mensili

Come si può notare i dati in rosso sono le presenze che vengono segnate in GAGI dai docenti mentre quelli in blu i dati letti dai sensori. In caso che si voglia visualizzare unicamente uno dei due dati basta premere sul colore che si vuole escludere, verrà fuori una barra e verrà visualizzato unicamente l'altro colore.

4.7.9 Ricerca (amministratori, direzione, segretariato e docenti)

La ricerca viene fatta in base alle lezioni, che durano per un massimo di 90 minuti. Per farla bisogna recarsi sulla pagina apposita (*Cerca*) ed inserire nei filtri le specifiche della lezione.

Filtri

Aula Durata (minuti)

Figura 76 -- Filtri ricerca

Nell'esempio viene richiesta la statistica di un'aula che durante la lezione ha avuto unicamente due presenze al posto di 21, il risultato è mostrato nell'immagine sottostante.

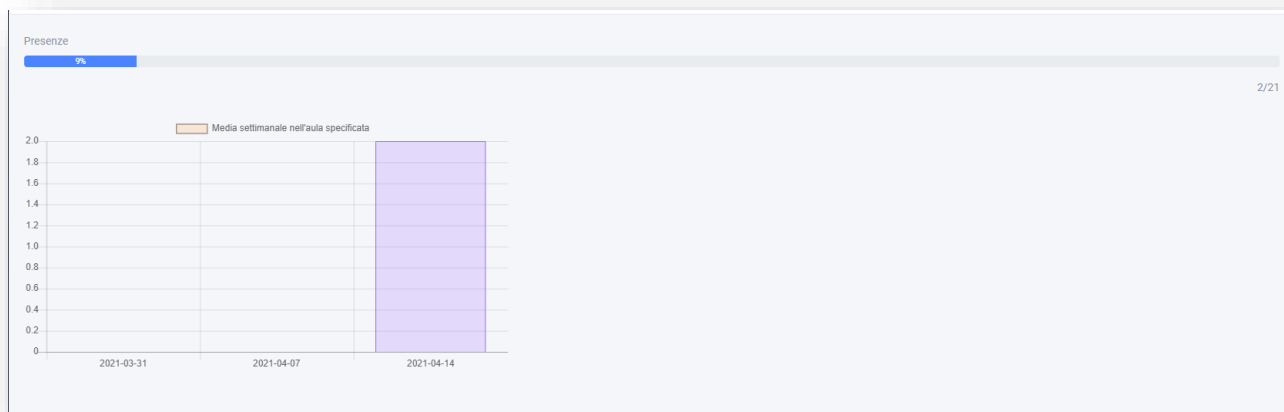


Figura 77 -- Risultati ricerca

I risultati sono riportati in più forme, innanzitutto si ha il vero e proprio dato a destra con scritto il numero di allievi presenti e quelli che dovrebbero esserlo (2/21). Nella barra blu si ha una percentuale del numero di persone presenti, in questo caso il 9% della classe. Infine si ha un grafico delle ultime 3 settimane con le presenze per ogni settimana.

4.7.10 Informazioni utente

Come ultima schermata si hanno le informazioni sull'utente, per arrivarci bisogna cliccare sul proprio nome utente e successivamente su *My profile*.

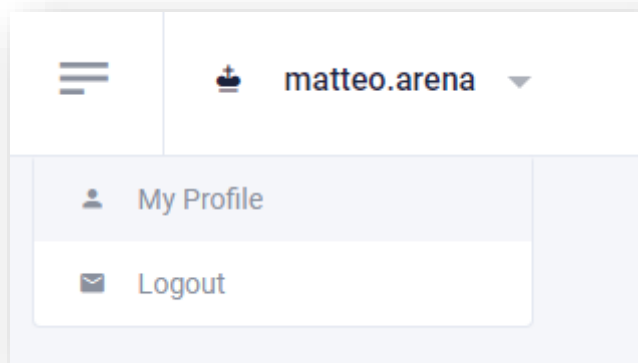


Figura 78 -- Menu a tendina utente

Nella pagina che si aprirà verrà riportata la propria scheda utente, con il proprio nome utente e il tipo di utente che si è. Al di sotto di questa informazione sarà presente il nome degli stabilimenti nel quale l'utente ha il permesso di accedere.

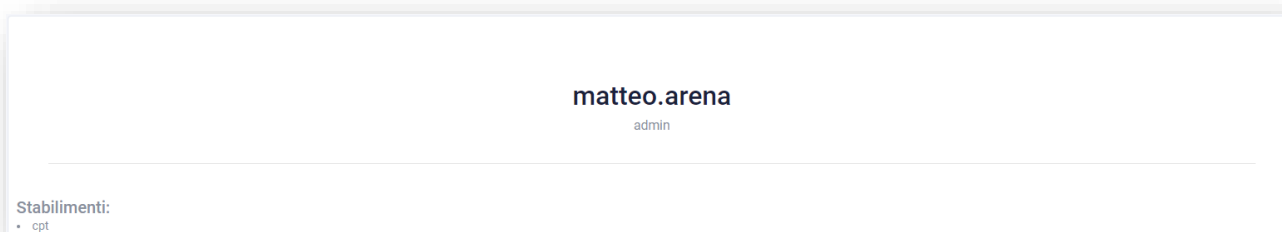


Figura 79 -- Info utente

5 Test

5.1 Protocollo di test

Test Case:	TC-001	Nome:	Accesso amministratore
Riferimento:	REQ-001		
Descrizione:	Si proverà ad effettuare l'accesso tramite l'utilizzo di un profilo amministratore.		
Prerequisiti:	Pagina di login completata.		
Procedura:	<ol style="list-style-type: none"> 1. Cercare sul proprio browser https://saminfo.ch/i17aremat/EntrateUscite/ 2. Cliccare su <i>Login utente amministratore</i> 3. Inserire i seguenti dati: <ol style="list-style-type: none"> a. Dominio: cpt b. Email: Admin c. Password: Admin01? 		
Risultati attesi:	Ci si dovrebbe trovare davanti alla pagina amministratore all'interno dell'applicativo WEB.		

Test Case:	TC-002	Nome:	Accesso amministratore con credenziali non corrette
Riferimento:	REQ-001		
Descrizione:	Il test è il medesimo di TC-001 con l'unica differenza che verranno provate delle credenziali non corrette per vedere se l'applicativo consente ugualmente l'accesso.		
Prerequisiti:	Pagina di login completata.		
Procedura:	<ol style="list-style-type: none"> 1. Cercare sul proprio browser https://saminfo.ch/i17aremat/EntrateUscite/ 2. Cliccare su <i>Login utente amministratore</i> 3. Inserire i seguenti dati: <ol style="list-style-type: none"> a. Dominio: cpt b. Email: Admi c. Password: Admin01? 4. Inserire i seguenti dati (se non ha funzionato): <ol style="list-style-type: none"> a. Dominio: cbt b. Email: Admin c. Password: Admin01? 5. Inserire i seguenti dati (se non ha funzionato): <ol style="list-style-type: none"> a. Dominio: cpt b. Email: Admin c. Password: Admin1? 		
Risultati attesi:	Non dovrebbe essere possibile l'accesso		

Test Case:	TC-003	Nome:	Accesso docente
Riferimento:	REQ-002		
Descrizione:	Si proverà ad effettuare l'accesso tramite l'utilizzo di un profilo docente.		
Prerequisiti:	Pagina di login completata e comunicazione con AD dello stabile.		
Procedura:	<ol style="list-style-type: none"> 1. Cercare sul proprio browser https://saminfo.ch/i17aremat/EntrateUscite/ 2. Inserire i dati di un docente 		
Risultati attesi:	Ci si dovrebbe trovare davanti alla pagina dedicata ai docenti all'interno dell'applicativo WEB.		

Test Case:	TC-004	Nome:	Accesso segreteria/direzione
Riferimento:	REQ-002		
Descrizione:	Si proverà ad effettuare l'accesso tramite l'utilizzo di un profilo segreteria/direzione.		
Prerequisiti:	Pagina di login completata e comunicazione con AD dello stabile.		
Procedura:	<ol style="list-style-type: none"> 1. Cercare sul proprio browser https://saminfo.ch/i17aremat/EntrateUscite/ 2. Inserire i dati di un membro della segreteria/direzione 		
Risultati attesi:	Ci si dovrebbe trovare davanti alla pagina dedicata alla segreteria/direzione all'interno dell'applicativo WEB.		

Test Case:	TC-005	Nome:	Accesso segreteria/direzione con credenziali errate
Riferimento:	REQ-002		
Descrizione:	Si proverà ad effettuare l'accesso tramite l'utilizzo di un profilo segreteria/direzione però con delle credenziali non esatte.		
Prerequisiti:	Pagina di login completata e comunicazione con AD dello stabile.		
Procedura:	<ol style="list-style-type: none"> 1. Cercare sul proprio browser https://saminfo.ch/i17aremat/EntrateUscite/ 2. Inserire dati casuali e tentare il login 		
Risultati attesi:	Dovrebbe comparire un messaggio di errore che non permette di accedere all'applicativo WEB.		

Test Case:	TC-006	Nome:	Permessi utente amministratore
Riferimento:	REQ-003		
Descrizione:	Con questo test si vuole controllare se l'utente amministratore possiede tutti i permessi che richiede.		
Prerequisiti:	Login completato, controllo degli utenti che fanno l'accesso		
Procedura:	<ol style="list-style-type: none"> 1. Cercare sul proprio browser https://saminfo.ch/i17aremat/EntrateUscite/ 2. Eseguire l'accesso con un utente amministratore 		
Risultati attesi:	Eseguito il login con utente amministratore dovranno comparire le seguenti voci nel menu a sinistra: <ul style="list-style-type: none"> • Dashboard Admin • Statistiche • Ricerca 		

Test Case:	TC-007	Nome:	Accesso persona
Riferimento:	REQ-004		
Descrizione:	Si vuole controllare se quando si entra in aula viene incrementato il numero di persone all'interno di essa.		
Prerequisiti:	Circuito completo e codice funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Spegner Arduino 2. Fare uscire una persona 3. Accendere arduino (Questo procedimento (accendere e spegnere arduino) viene fatto per evitare che l'Arduino conti la persona uscita, in questo modo non conterebbe la persona che entra.) 4. Fare entrare la persona 5. Eseguire accesso all'applicativo 6. Controllare le statistiche per vedere se il numero di allievi è incrementato 		
Risultati attesi:	Il numero di persone dovrebbe essere incrementato.		

Test Case:	TC-008	Nome:	Uscita persona
Riferimento:	REQ-004		
Descrizione:	Si vuole controllare se quando si esce dall'aula viene decrementato il numero di persone all'interno di essa.		
Prerequisiti:	Circuito completo e codice funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Fare uscire una persona 2. Eseguire accesso all'applicativo 3. Controllare le statistiche per vedere se il numero di allievi è decrementato 		
Risultati attesi:	Il numero di persone dovrebbe essere decrementato.		

Test Case:	TC-009	Nome:	Controllo margine di errore
Riferimento:	REQ-005		
Descrizione:	Provare a fare entrare più di una persona per vedere se viene visto		
Prerequisiti:	Circuito completo e codice funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Provare a far entrare più persone nello stesso momento 2. Controllare l'applicativo WEB 		
Risultati attesi:	Il numero di persone deve incrementare correttamente		

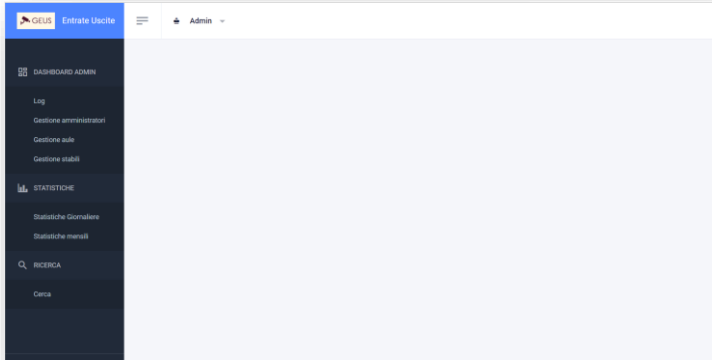
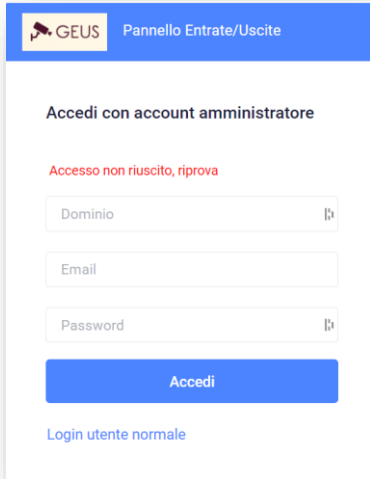
Test Case:	TC-010	Nome:	Inserimento dati nel database
Riferimento:	REQ-006		
Descrizione:	Controllare dati di un'entrata sull'applicativo WEB.		
Prerequisiti:	Applicativo funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Far entrare una persona 2. Controllare l'applicativo WEB sulla pagina Dashboard Admin -> Logs 		
Risultati attesi:	Deve essere visualizzata un'informazione appropriata.		

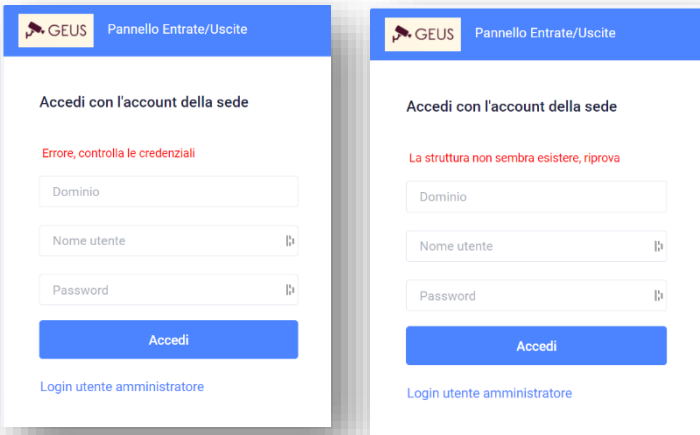
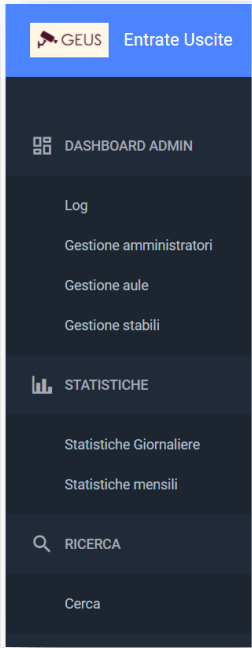
Test Case:	TC-011	Nome:	Controllo tabella log
Riferimento:	REQ-007 REQ-008		
Descrizione:	Provare ad applicare tutti i filtri		
Prerequisiti:	Applicativo web funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Eseguire l'accesso all'applicativo 2. Andare nella sezione Dashboard Admin 3. Andare nel sottomenu Log 4. Controllare tutti gli <i>Info</i> 5. Controllare tutti i <i>Warning</i> 6. Controllare tutti gli <i>Error</i> 7. Applicare un filtro con le date 		
Risultati attesi:	Tutti i filtri devono funzionare.		

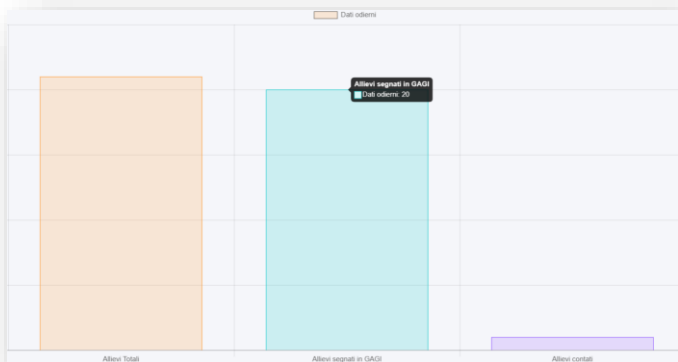
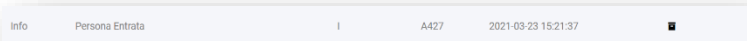
Test Case:	TC-012	Nome:	Statistica
Riferimento:	REQ-009		
Descrizione:	Controllare la pagina di statistica		
Prerequisiti:	Applicativo web funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Eseguire l'accesso all'applicativo 2. Andare nella sezione Statistiche 3. Andare nella sezione statistiche giornaliere 4. RISULTATO 1 5. Andare nella sezione statistiche mensili 6. RISULTATO 2 		
Risultati attesi:	<p>Risultato 1: Nelle statistiche giornaliere devono essere presenti le persone che attualmente sono all'interno della sede.</p> <p>Risultato 2: Nelle statistiche mensili invece vanno trovate le persone che sono presenti in aula in base ai vari giorni.</p>		

Test Case:	TC-013	Nome:	Archiviazione dei dati
Riferimento:	REQ-010		
Descrizione:	Provare ad archiviare un dato e vedere se influisce sui grafici		
Prerequisiti:	Applicativo web funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Eseguire l'accesso all'applicativo 2. Andare nella pagina con i log 3. Archiviare l'informazione di una persona uscita 4. Andare nelle statistiche giornaliere 5. Controllare che sia incrementato 		
Risultati attesi:	Andando nella pagina delle statistiche giornaliere il numero di persone all'interno dell'aula dovrebbe incrementare, questo perché si è contata una persona entrare ma essa non è uscita.		

5.2 Risultati test

Test case	Esito	Risultati	Data
TC-001	PASSATO	 <p>Viene mostrata questa pagina web nei browser più famosi in circolazione (Edge, Firefox, Chrome e Safari).</p>	23.03.2021
TC-002	PASSATO	 <p>Qualsiasi combinazione si provi viene fuori il seguente errore, che non permette l'accesso all'applicativo</p>	23.03.2021
TC-003	PARZIALMENTE PASSATO	Purtroppo questo test non può passare con l'applicativo caricato sul web host, ma con l'applicativo in locale funziona. Non avendomi dato un profilo docente per fare i test comunico con il server AD tramite il mio account e ricevo le informazioni che dovrebbe ricevere un docente qualsiasi, in questo modo mi trovo davanti all'applicativo visto dagli occhi di un docente.	23.03.2021
TC-004	PARZIALMENTE PASSATO	Vedesi descrizione TC-003	23.03.2021

TC-005	PASSATO	 <p>In base a cosa si sbaglia viene mostrato l'errore appropriato (nella prima si sono sbagliate le credenziali, nella seconda lo stabile).</p>	23.03.2021
TC-006	PASSATO	<p>Effettuato il login la schermata a sinistra si mostra così:</p> 	23.03.2021

TC-007	PASSATO	<div>Il numero di persone viene incrementato (immagine sottostante)</div> <div></div>	23.03.2021																																				
TC-008	PASSATO	<div>Il numero di persone scende, questo significa che il test è corretto.</div>	23.03.2021																																				
TC-009	NON PASSATO	<div>Purtroppo i sensori non riescono a gestire più di una persona che esce/entra contemporaneamente.</div>	23.03.2021																																				
TC-010	PASSATO	<div></div>	23.03.2021																																				
TC-011	PASSATO	<div><div>Si Stanno Visualizzando i Dati Da 10/10/2020 A 10/10/2021</div><table><thead><tr><th>Tipo</th><th>Descrizione</th><th>Azione</th><th>Aula</th><th>Data</th><th>Azioni</th></tr></thead><tbody><tr><td>Info</td><td>Persona entrata</td><td>I</td><td>A421</td><td>2021-02-01 13:15:00</td><td></td></tr><tr><td>Info</td><td>Persona entrata</td><td>I</td><td>A421</td><td>2021-02-01 13:16:00</td><td></td></tr></tbody></table><div><div>Archiviati</div><div>Visibili</div><div>Info</div><div>Error</div><div>Warning</div><div>Elimina tutti i filtri</div></div><table><thead><tr><th>Tipo</th><th>Descrizione</th><th>Azione</th><th>Aula</th><th>Data</th><th>Azioni</th></tr></thead><tbody><tr><td>Info</td><td>Persona entrata</td><td>I</td><td>A421</td><td>2021-02-01 13:15:00</td><td></td></tr><tr><td>Info</td><td>Persona entrata</td><td>I</td><td>A421</td><td>2021-02-01 13:16:00</td><td></td></tr></tbody></table></div>	Tipo	Descrizione	Azione	Aula	Data	Azioni	Info	Persona entrata	I	A421	2021-02-01 13:15:00		Info	Persona entrata	I	A421	2021-02-01 13:16:00		Tipo	Descrizione	Azione	Aula	Data	Azioni	Info	Persona entrata	I	A421	2021-02-01 13:15:00		Info	Persona entrata	I	A421	2021-02-01 13:16:00		23.03.2021
Tipo	Descrizione	Azione	Aula	Data	Azioni																																		
Info	Persona entrata	I	A421	2021-02-01 13:15:00																																			
Info	Persona entrata	I	A421	2021-02-01 13:16:00																																			
Tipo	Descrizione	Azione	Aula	Data	Azioni																																		
Info	Persona entrata	I	A421	2021-02-01 13:15:00																																			
Info	Persona entrata	I	A421	2021-02-01 13:16:00																																			

TC-012	PASSATO		23.03.2021
TC-013	PASSATO	 <p>Dopo aver archiviato una persona uscita (passa da 2 a 3).</p> 	23.03.2021

5.3 Mancanze/limitazioni conosciute

La mancanza più grande del progetto è sicuramente l'impossibilità di contare più entrate contemporaneamente. Ad esempio se due allievi entrano nel medesimo istante all'interno dell'aula le fotocellule capteranno unicamente un'entrata. Questo problema però è purtroppo dato dalle limitazioni delle fotocellule, le quali non riescono a fare un minimo di filtraggio in caso che entrino o escano più persone contemporaneamente.

L'unica soluzione che risolverebbe questa limitazione sarebbe optare per altre fotocellule, questo purtroppo non mi è stato possibile siccome sin dall'inizio del progetto ho avuto a disposizione unicamente queste fotocellule.

6 Consuntivo

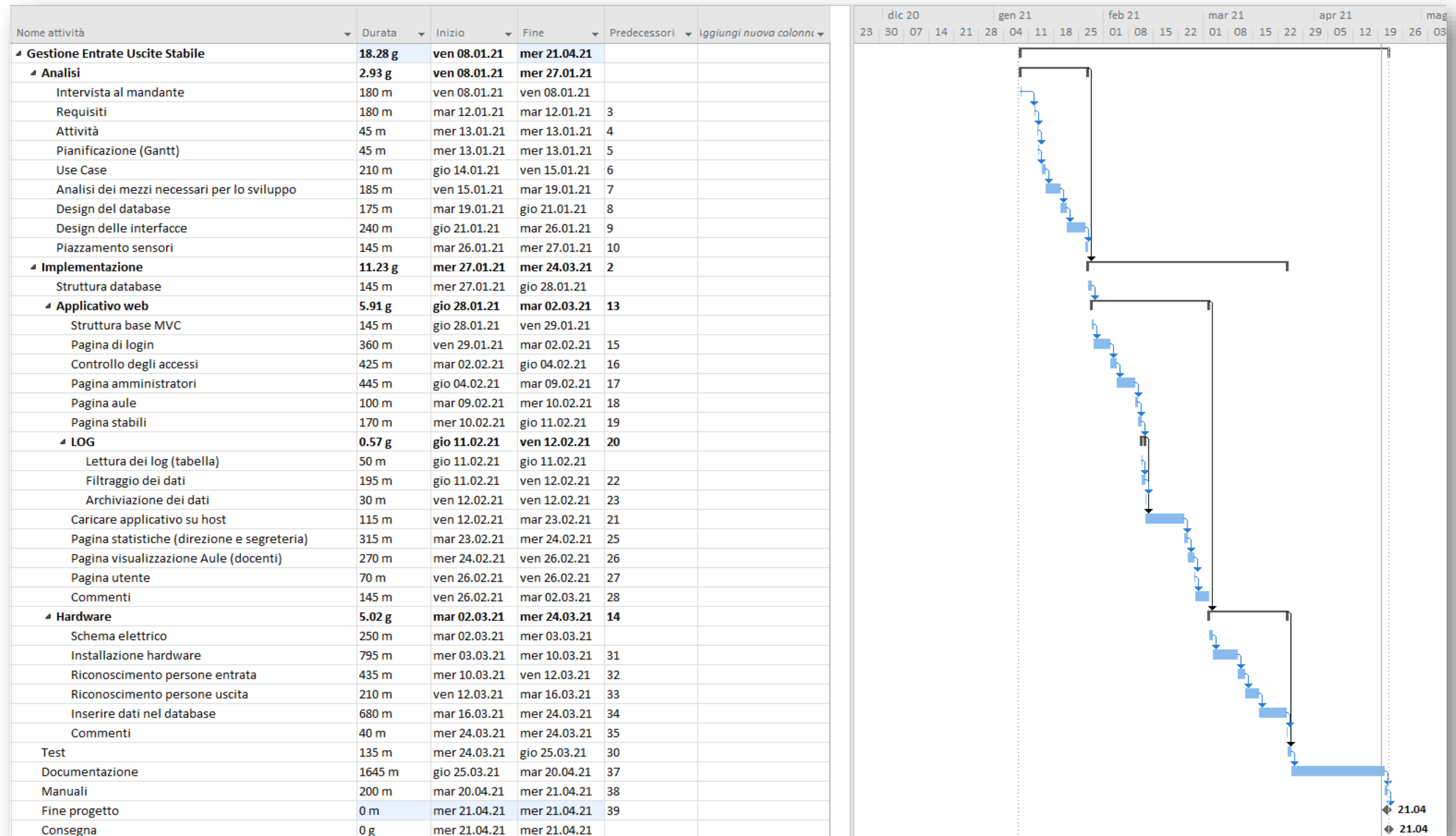


Figura 80 -- Gantt finale

7 Conclusioni

7.1 Sviluppi futuri

La miglioria più grande che si può fare è sicuramente l'upgrade con delle fotocellule più intelligenti, che permettano di migliorare il riconoscimento delle persone in entrata e che siano soprattutto più comode. Infatti ho trovato che le fotocellule utilizzate siano piuttosto scomode siccome andavano alimentate tramite un alimentatore esterno, non fornendo l'Arduino la tensione necessaria.

7.2 Considerazioni personali

In conclusione posso dire che il progetto mi ha insegnato a lavorare in modo estremamente più ordinato, sia lato di codice che lato dei diari. Infatti durante questo progetto a mio parere ho tenuto molta cura nei diari e in quello che andavo a svolgere durante ogni giornata di lavoro (ad eccezione ovviamente degli ultimi che contengono solo cose minime). Il risultato finale del progetto mi ha abbastanza soddisfatto, l'unica cosa che mi spiace è non aver a disposizione delle fotocellule più avanzate che permettessero di riconoscere più persone in entrata.

Infine come tempistiche posso dire che sono soddisfatto del progetto siccome sono riuscito a gestire al meglio il mio tempo, senza aver mai bisogno di tempo in più per riuscire a finire tutto né tempo che riempivo con pause inutili.

8 Bibliografia

8.1 Sitografia

- https://www.distrelec.ch/de/reflexions-lichtschranke-4m-pnp-panasonic-cy-192b/p/13763418?channel=b2c&price_gs=35.34714&source=googleps&ext_cid=shg00aqchde-na&ext_cid=shg00aqchde-P-CSS-SmartShopping&gclid=CjwKCAiAouD_BRBIEiwALhJH6A0-CIXoRicCBMSBi4Ut5w_7hSB0ge6h96qC90kvninGy0ZXUwGixocImoQAvD_BwE, Distrelec, 11-01-2021.
- <https://app.creately.com/>, 21.01.2021
- <https://www.mockflow.com/>, 26.01.2021
- www.arduino.cc, 27.01.2021
- www.looka.com, 28.01.2021
- [AD Explorer - Windows Sysinternals | Microsoft Docs](#), 28.01.2021
- <https://www.php.net/>, 29.01.2021
- <https://www.w3schools.com/>, 26.02.2021
- https://www.lelezapp.it/fatal-error-call-to-undefined-function-ldap_connect/, 29.01.2021
- <https://github.com/adLDAP2/adLDAP2>, 02.02.2021
- <https://www.chartjs.org/>, 12.02.2021
- <https://code-boxx.com/add-subtract-date-php/>, 25.02.2021

9 Glossario

Nome	Definizione
AD	Sta per Active Directory, è un insieme di servizi di rete adottato sui sistemi operativi Windows, per identificare ad esempio i computer, gli utenti, ...
Applicativo WEB	Si intende il sito web che verrà visualizzato a progetto completato.
Arduino	Piattaforma hardware dedicata a completare piccoli progetti (esso è composto da schede elettroniche con un microcontrollore).
Boolean	Valore che può assumere unicamente due significati, vero o falso.
Base_dn	OU di base dalla quale si parte per ricavare tutti gli utenti all'interno dell'Active Directory.
CN	Common Name, nome dell'utente o del gruppo in una struttura LDAP.
Database	Struttura separata in tabelle che contiene tutte le informazioni riguardanti un determinato applicativo/progetto.
Datetime	Valore che contiene giorno, mese, anno, ore, minuti e secondi in SQL.
Error (log)	Solitamente adottato nei log per salvare tutti gli errori che vengono captati.
GAGI	Applicativo WEB utilizzato dalle scuole per il salvataggio delle assenze di una specifica classe.
Gantt	Diagramma per pianificare un intero progetto e dividerlo in varie attività. Grazie ad esso si può avere un lavoro più organizzato.
Hardware	Parte fisica di un sistema, ad esempio in questo progetto un esempio di hardware sono i sensori ad infrarossi.
Info (log)	Adottato nei log per il salvataggio di eventi non dannosi per il sistema finale, ad esempio nel progetto l'entrata di una persona in un'aula.
Int	Tipo di dato che contiene unicamente valori numerici interi.
LDAP	Protocollo per l'interrogazione del server apposito che contiene tutti gli utenti, divisi in gruppi, riguardo una determinata azienda/scuola.
Log	Insieme di dati dedicati all'analisi.
OU	Organizational Unit, contiene i CN che fanno parte di un determinato gruppo.
PHP	Linguaggio di programmazione lato Server per la gestione degli applicativi WEB.
Time	Tipo di dato che salva unicamente ora e minuti in SQL.
Varchar	Tipo di dato per il salvataggio di stringhe in SQL.
Warning	Adottato nei log quando è presente un evento che non danneggia il sistema ma potrebbe farlo in un'operazione futura, ad esempio nel progetto staccare un sensore ad infrarossi causerà un errore in futuro.