Sensore cassetta delle lettere

# Indice

1 Indice 2

2 Introduzione 3

2.1 Informazioni sul progetto 3

2.2 Abstract 3

2.3 Scopo 3

3 Analisi 4

3.1 Analisi del dominio 4

3.2 Analisi e specifica dei requisiti 4

3.3 Use case 6

3.4 Pianificazione 7

3.5 Analisi dei mezzi 8

3.5.1 Software 8

3.5.2 Hardware 8

4 Progettazione 9

4.1 Design del database 9

4.2 Design delle interfacce 10

5 Implementazione 13

5.1 Web2Py 13

5.1.1 layout.html 13

5.1.2 index.html 14

5.1.3 notif\_list.html 15

5.1.4 notif\_calendar.html 16

5.1.5 user.html 17

5.1.6 devices.html 18

5.1.7 default.py 18

5.2 Arduino 25

5.2.1 File ReadWrite 25

5.2.2 File Main 28

6 Test 36

6.1 Protocollo di test 36

6.2 Risultati test 37

6.3 Mancanze/limitazioni conosciute 38

7 Consuntivo 39

8 Conclusioni 40

8.1 Sviluppi futuri 40

8.2 Considerazioni personali 40

9 Bibliografia 41

9.1 Sitografia 41

10 Indice delle figure 42

11 Allegati 43

# Introduzione

## Informazioni sul progetto

* Allievi: Alessandro Perri, Alex Ierardi e Diego Muniz
* Supervisore: Geo Petrini
* Classe: 3BB SAM Trevano, sezione informatica
* Data di inizio progetto: 27.01.23
* Data di consegna: 05.05.2023

## Abstract

**Situazione iniziale**

Inizialmente in questo progetto ci è stato chiesto di realizzare un sistema per il rilevamento di posta e pacchi in una cassetta delle lettere.

**Approccio**

Per questo progetto, saranno necessarie buone conoscenze per realizzare il server web e il database che dovranno comunicare, conoscenze nella programmazione web per poter realizzare la GUI, servirà saper utilizzare Arduino per il codice di raccolta e trasmissione dati dal Fishino.

**Risultati**

Per realizzare quello che ci è stato richiesto abbiamo utilizzato web2py che è un framework per applicazioni web. Grazie ad esso siamo riusciti a realizzare la GUI e il database che risiedono sul server web. Web2py utilizza Pyhton come linguaggio di programmazione, siccome nessuno di noi non lo aveva mai utilizzato prima abbiamo avuto la possibilità di ampliare il nostro bagaglio delle conoscenze

## Scopo

Scopi

* Didattici:
  + Saper creare e rispettare una progettazione.
  + Saper documentare il lavoro.
  + Saper creare i diari.
  + Saper creare Use-case.
* Operativi:
  + Saper creare una GUI web.
  + Saper utilizzare Python nel framework Web2py.

# Analisi

## Analisi del dominio

Per questo progetto è stato richiesto di sviluppare un sistema per il rilevamento della posta e dei pacchi in una cassetta delle lettere. Non appena il sistema rileverà una lettera, all’utente verrà inviata una notifica che indica l’arrivo della posta.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Realizzare codice per la trasmissione dei dati |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** |  |
| **002** |  |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Realizzare codice per la raccolta dei dati |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Elaborare un protocollo di trasmissione |
| **002** |  |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Realizzare l’hardware del dispositivo |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Progettare con un diagramma l’hardware |
| **002** |  |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Salvataggio all’interno di un database |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Scegliere il tipo di database |
| **002** |  |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Realizzare la GUI web |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Pagina delle notifiche |
| **002** | Pagina per la configurazione dei sensori |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Storico delle notifiche |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Visualizzazione in lista |
| **002** | Visualizzazione in calendario |

**Spiegazione elementi tabella dei requisiti**

**ID**: identificativo univoco del requisito.

**Nome**: breve descrizione del requisito.

**Priorità**: l'importanza con la quale deve essere svolto un requisito.

**Versione**: indica la versione del requisito.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

Figura 1: Use-Case

## Pianificazione

Di seguito c’è la pianificazione preventiva, nel Gantt preventivo è stimato il tempo necessario per completare le task del progetto.

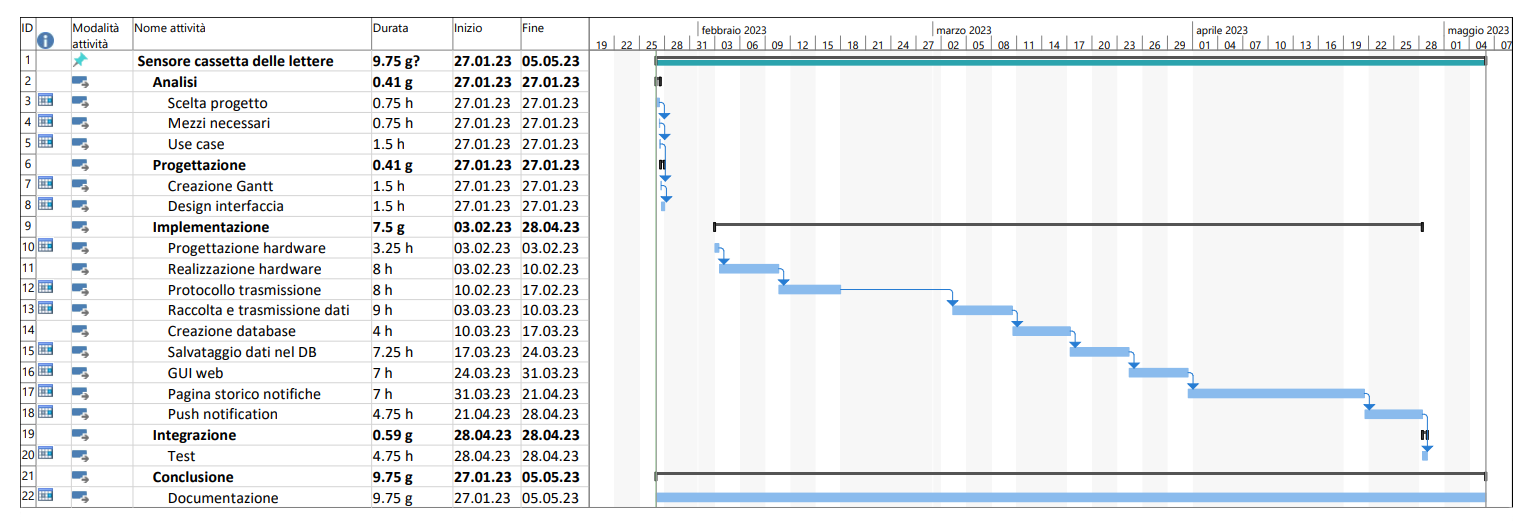


Figura 2: Gantt preventivo

## Analisi dei mezzi

Per realizzare il progetto è stato utilizzato un computer, Web2py come framework e un server per hostare il sito e salvare tutti i dati.

### Software

Per realizzare questo progetto sono stati utilizzati i seguenti software:

* OS Computer - Windows 10: Per la creazione di tutto il progetto.
* Web2py – 2.24.1: Utilizzato come framework.

### Hardware

* HDD e SSD per memorizzare le VM.
* Server per hostare la pagina e salvare il database.

Un computer con:

* CPU Intel Core i7-7700
* RAM 16 GB

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

## Design del database

Figura 3: Schema E-R

Questo è il nostro schema E-R per la realizzazione del database.

All’interno del nostro database abbiamo le seguenti tabelle:

* La tabella “User”, dove verranno registrati gli utenti che accederanno al sito.
* La tabella “Device”, dove verranno registrati i dispositivi utilizzati dall’utente.
* La tabella “Notification”, che invierà una notifica all’utente appena arriverà la posta.
* La tabella “Recorded\_value”, che salverà quando è arrivata la posta.

## Design delle interfacce

Il design del sito è strutturato nel seguente modo:

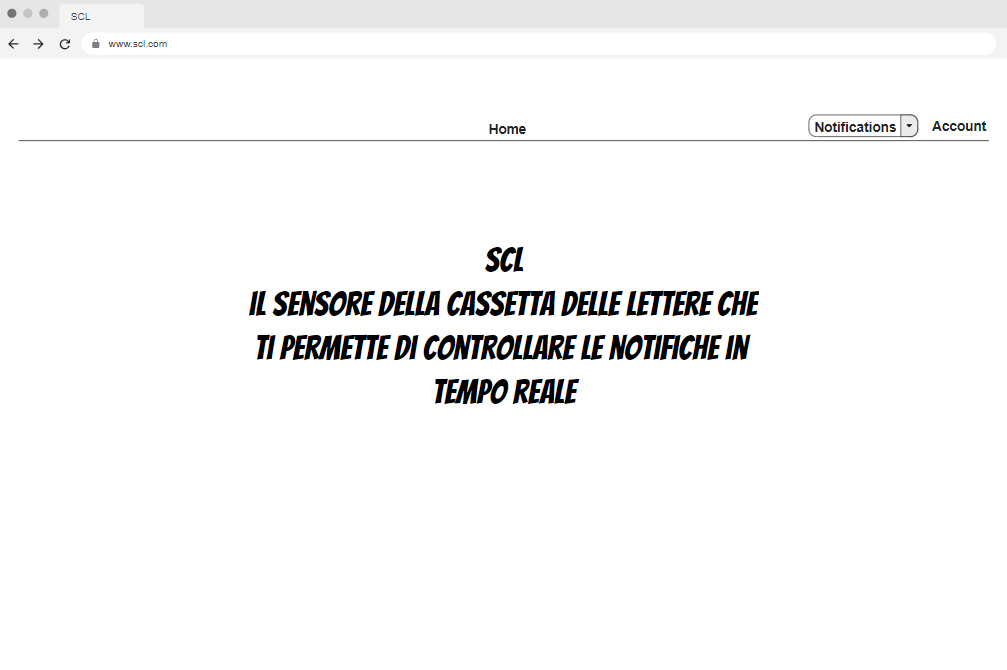
**Interfaccia home page**

Figura 4: Home page

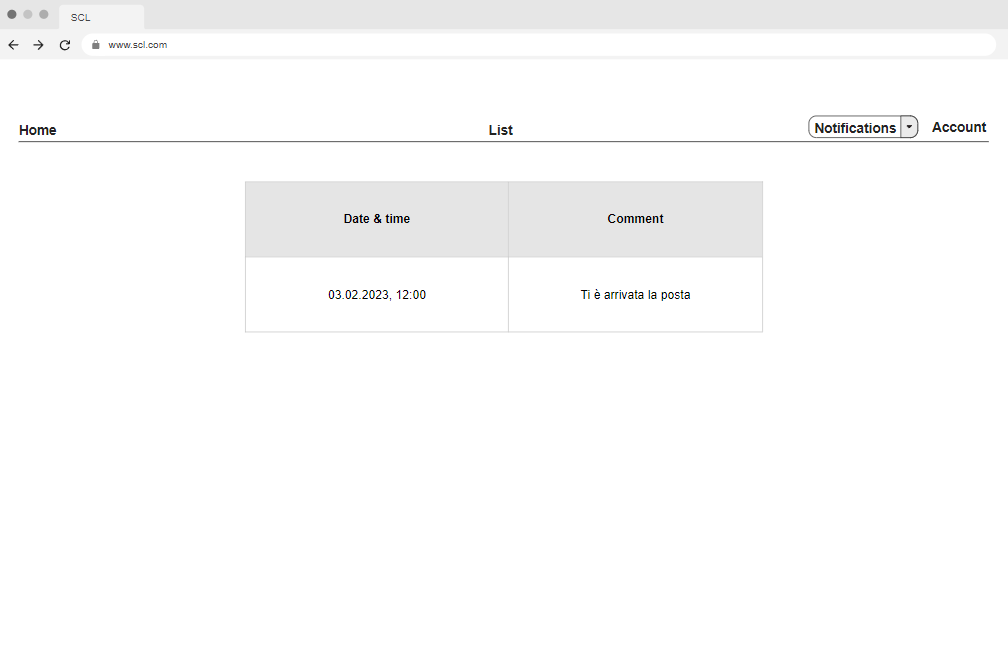
**Interfaccia notifiche in lista**

Figura 5: Notifiche in lista

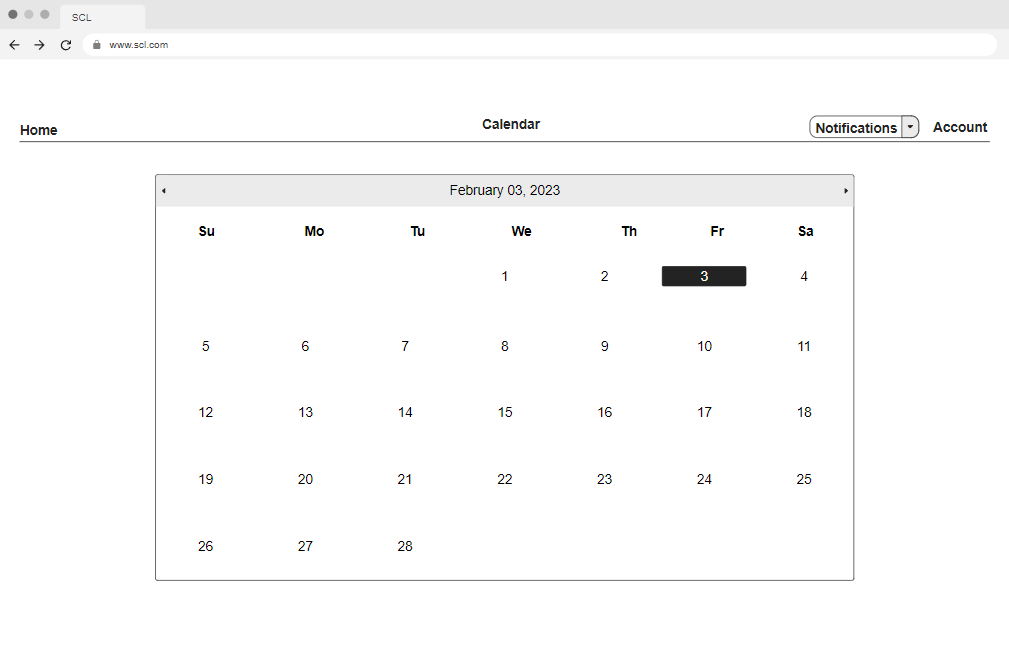
**Interfaccia notifiche nel calendario**

Figura 6: Notifiche in calendario

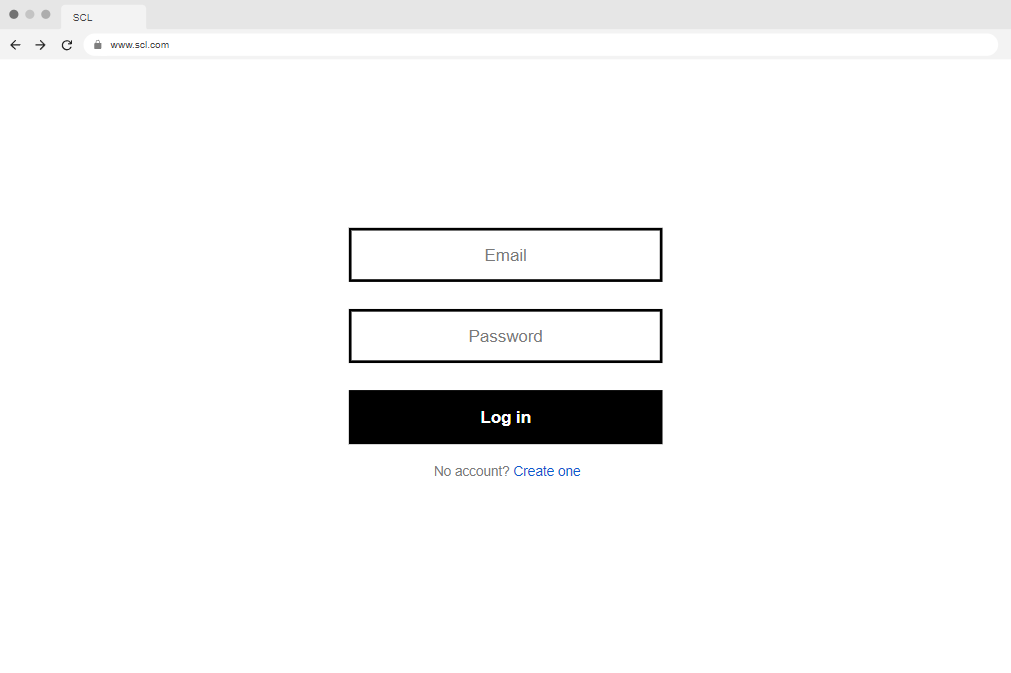
**Interfaccia di login**

Figura 7: Login

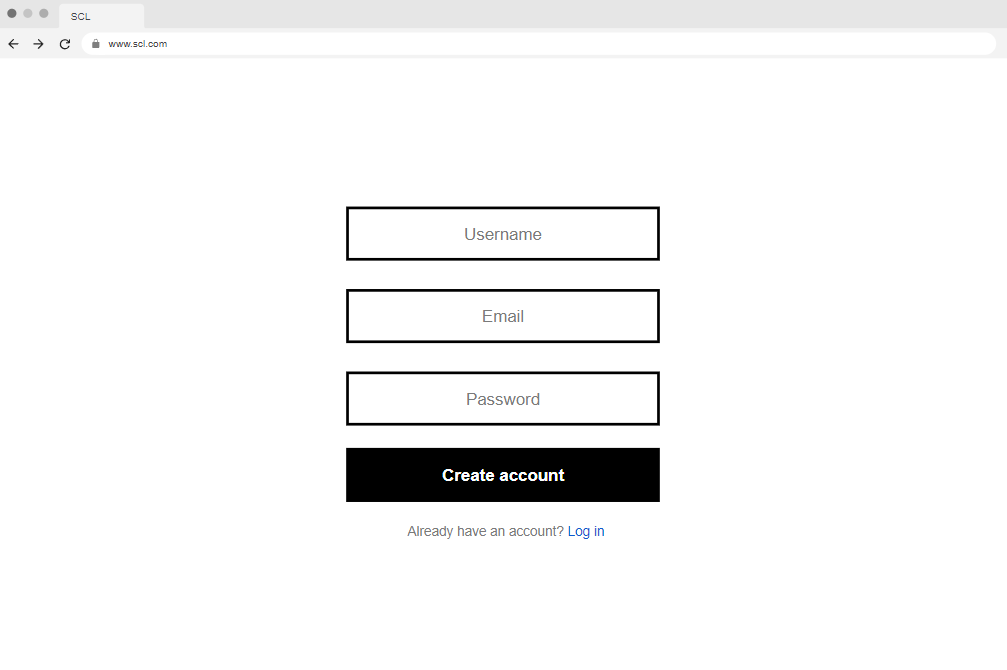
**Interfaccia creazione account**

Figura 8: Creazione account

# Implementazione

## Web2Py

### layout.html

La view layout.html rappresenta il layout di base per tutte le altre views. Questo layout contiene i meta tags e i link agli stylesheet, inclusi quelli di Bootstrap 4, che vengono utilizzati per la formattazione e lo stile del sito web.

Figura : Navbar

Inoltre, la navbar presente nel layout offre un facile accesso alle diverse views create, tra cui index.html, devices.html, notif\_list.html, notif\_calendar.html e user.html. Questi link sono utili per consentire agli utenti di navigare rapidamente tra le diverse pagine del nostro sito web.

### C:\Users\diego.muniz\Downloads\127.0.0.1_8000_scl_final_default_index (2).pngindex.html

Figura : Index.html

La view index.html presenta un design semplice e minimalista, essa rappresenta la prima pagina che gli utenti vedono quando accedono al sito web. Questa pagina è stata progettata per offrire un'esperienza di benvenuto accattivante ed efficace, presentando in modo chiaro e conciso le principali funzionalità del sito.

### notif\_list.html

Figura : Lista delle notifiche

La view notif\_list.html permette l’utente di visualizzare tutte le notifiche dei propri dispositivi in una semplice tabella, rendendo più facile per gli utenti analizzare i dati.

La tabella presenta le informazioni relative alle notifiche in un formato facile da leggere, tra cui la data di invio, il tipo di notifica e il dispositivo associato.

### notif\_calendar.html

La view notif\_calendar.html rappresenta un utile strumento per gli utenti poiché consente loro di visualizzare tutte le notifiche dei propri dispositivi in un comodo calendario. Questa view è stata creata utilizzando FullCalendar, che ci ha permesso di integrare facilmente un calendario nel nostro sito web.

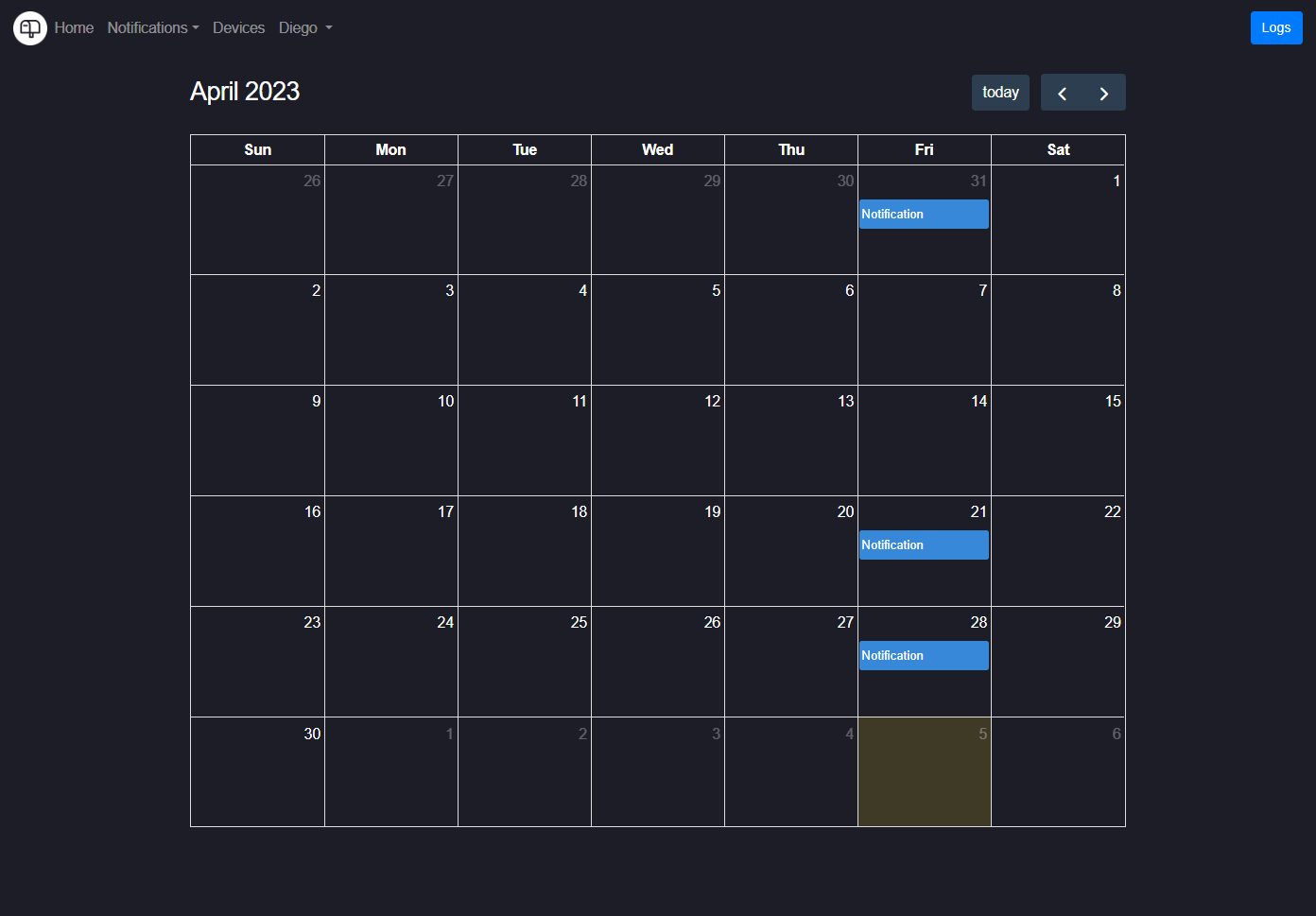


Figura : Notifiche nel calendario

### user.html

La view user.html rappresenta la pagina di login del sito, che consente agli utenti di accedere alle loro informazioni personali e di gestire il proprio account. La pagina di login è un componente cruciale del nostro sito web, poiché permette agli utenti di accedere a tutte le funzionalità principali del sito.

Inoltre, la view user.html offre anche la possibilità di registrarsi come nuovo utente, fornendo le proprie informazioni personali.



Figura : Log In

### devices.html

La view devices.html permette l’utente di visualizzare tutti i dispositivi registrati nel sistema. Ogni riga della tabella rappresenta un dispositivo e contiene le informazioni associate ad esso, come il model e la api key.

Inoltre, gli utenti possono visualizzare facilmente tutte le informazioni relative a un dispositivo selezionando la riga corrispondente nella tabella, e cercare per i dispositivi con la barra di ricerca apposita.

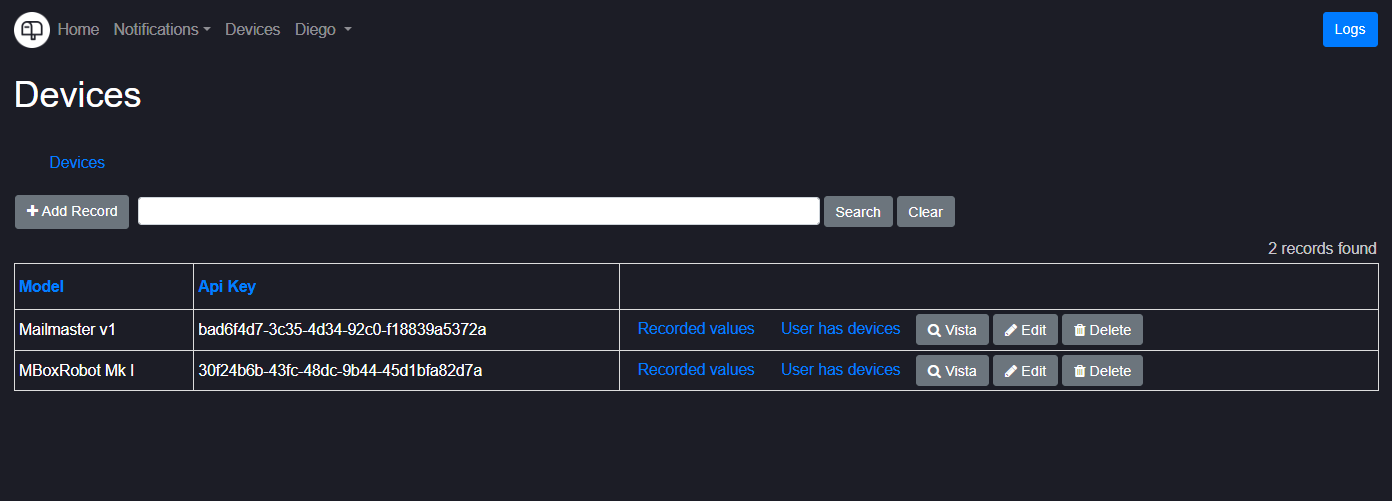


Figura : Devices

### default.py

Il file default.py rappresenta il controller principale della nostra applicazione il cui compito è quello di gestire le richieste degli utenti e di coordinare la visualizzazione delle diverse view.

Oltre a coordinare la visualizzazione delle view, il controller default.py contiene anche funzioni e metodi che permettono di manipolare i dati dell'applicazione, gestire le operazioni dell'utente e coordinare la comunicazione tra il modello e la view.

In termini di architettura MVC (Model-View-Controller), il controller default.py rappresenta il componente C del pattern, ovvero il controller che gestisce il flusso di dati tra il modello e la view. Grazie alla sua importanza nell'architettura dell'applicazione, il file default.py è il principale punto di accesso per la gestione e lo sviluppo del nostro progetto Web2Py.

Grazie al suo ruolo cruciale nell'architettura MVC, il controller default.py è uno dei file più importanti del nostro progetto.

**index()**

Questa funzione viene eseguita quando si accede alla home page del sito.

All'interno della funzione si imposta il ‘response.flash’ con il testo "First time on SCL?".   
Questo messaggio viene mostrato all'utente all'apertura della home page.

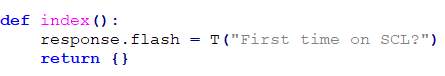


Figura 15. Metodo index

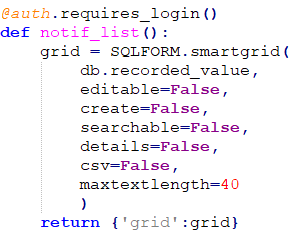
**notif\_list()**

Figura 16: Metodo notif\_list

Questa funzione viene eseguita quando si accede alla lista di notifiche del sito.

Genera una griglia (grid) che visualizza i valori registrati nel database ‘db.recorded\_value’.  
La griglia è configurata in modo che non sia possibile modificarne i contenuti (editable=False), né crearne di nuovi (create=False). Inoltre, la griglia non è ricercabile (searchable=False), non mostra i dettagli dei record (details=False), non consente di esportare i dati in formato CSV (csv=False) e limita il numero di caratteri visualizzati per i valori a 40 (maxtextlength=40).

Infine, la funzione restituisce un dizionario con la sola chiave ‘grid’, il cui valore è la griglia appena creata.

La decorazione ‘@auth.requires\_login()’ indica che l'accesso a questa pagina è consentito solo agli utenti autenticati.

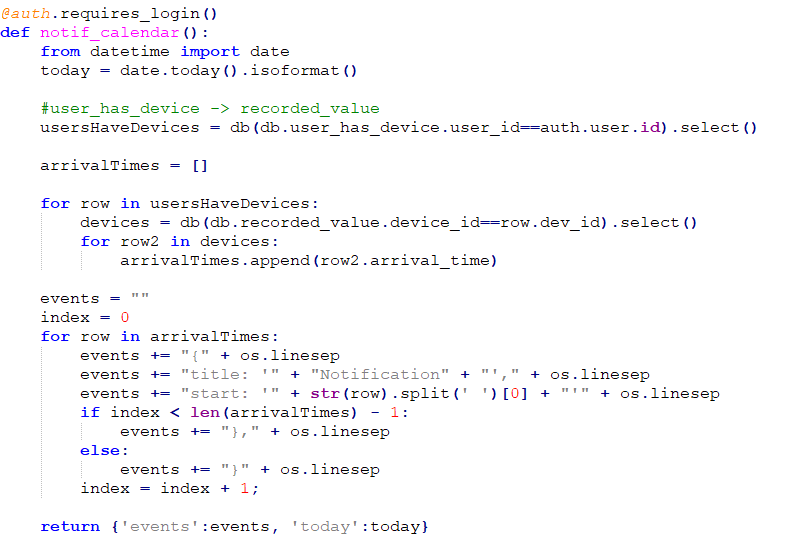
**notif\_calendar()**

Figura 17: Metodo notif\_calendar

Questa funzione viene eseguita quando si accede al calendario di notifiche del sito.

Recupera le informazioni sui dispositivi posseduti dall'utente attualmente autenticato, confrontando la tabella ‘user\_has\_device’ con l'id dell'utente registrato in ‘auth.user.id’.

Per ogni record trovato, recupera i valori registrati nella tabella ‘recorded\_value’ per il relativo id del dispositivo e aggrega gli orari di arrivo in una lista ‘arrivalTimes’.

Infine, costruisce una stringa di eventi del calendario nel formato richiesto, utilizzando la lista ‘arrivalTimes’ per popolare le date di inizio degli eventi. Questa stringa di eventi viene restituita insieme alla data attuale come dizionario.

La decorazione ‘@auth.requires\_login()’ indica che l'accesso a questa pagina è consentito solo agli utenti autenticati.

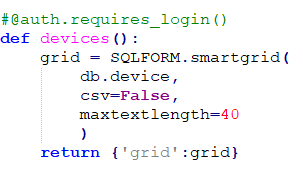
**devices()**

Figura 18: Metodo devices

Questa funzione viene eseguita quando si accede alla lista di devices del sito.

Genera una griglia (grid) che visualizza i valori registrati nel database ‘db.device’.  
La griglia è configurata in modo che non sia possibile di esportare i dati in formato CSV (csv=False) e limita il numero di caratteri visualizzati per i valori a 40 (maxtextlength=40).

Infine, la funzione restituisce un dizionario con la sola chiave ‘grid’, il cui valore è la griglia appena creata.

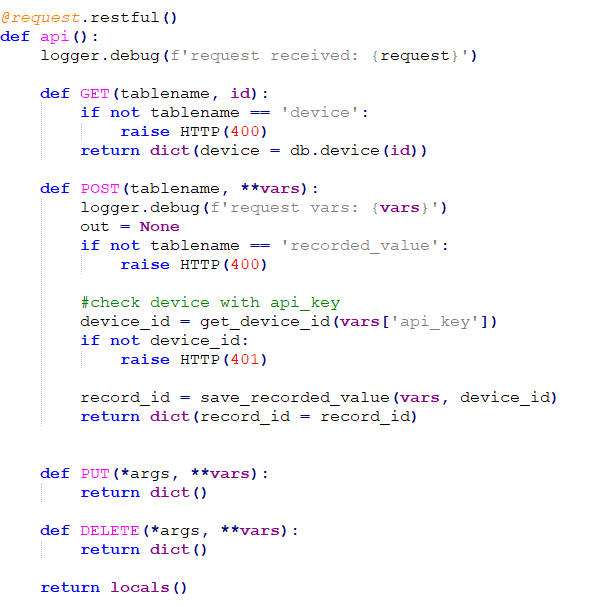
**api()**

Figura 19: Metodo api

È un servizio web RESTful che espone due metodi HTTP: GET, POST.

Nel metodo GET viene richiesto il parametro ‘tablename’ che deve essere uguale a ‘device’ e ‘id’ che rappresenta l'identificativo del dispositivo.   
Viene restituito il device dal database avente l'identificativo corrispondente.

Nel metodo POST viene richiesto il parametro ‘tablename’ che deve essere uguale a ‘recorded\_value’. Viene effettuato un controllo sull'api key del device per verificare l'autenticazione. Viene poi chiamata la funzione ‘save\_recorded\_value’ per salvare il valore registrato dal dispositivo nel database.   
Viene restituito l'identificativo del record appena creato.

Viene inoltre utilizzata la libreria `logger` per registrare gli accessi al servizio.

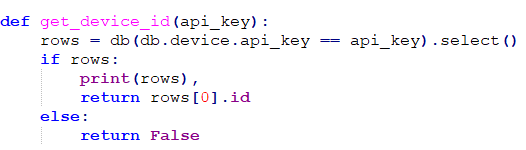
**get\_device\_id(api\_key)**

Figura 20: Metodo get\_device\_id

Riceve in input una stringa ‘api\_key’, che rappresenta la chiave di accesso dell'apparecchio, e cerca nel database l'eventuale presenza di un dispositivo associato a quella chiave.   
Se il dispositivo viene trovato, viene restituito il suo id, altrimenti viene restituito False.

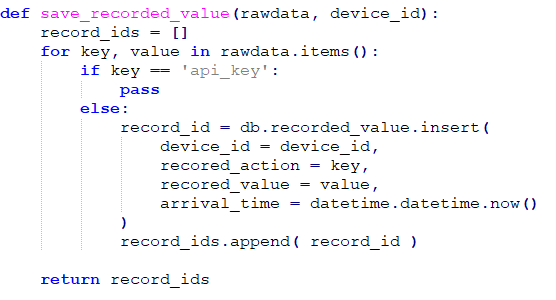
**save\_recorded\_value(rawdata, device\_id)**

Figura 21: Metodo save\_recorded\_value

Riceve in input una stringa ‘api\_key’, che rappresenta la chiave di accesso dell'apparecchio,   
e ‘rawdata’, un dizionario contenente coppie di chiavi e valori che rappresentano i dati da salvare nel database.

Cicla attraverso gli elementi del dizionario ‘rawdata’, e se la chiave è diversa da ‘api\_key’ inserisce il record nel database.

Il record inserito contiene le seguenti informazioni: l'id del dispositivo, l'azione registrata, il valore registrato, e il tempo di arrivo.

L'id del record appena inserito viene quindi aggiunto alla lista ‘record\_ids’.   
Al termine del ciclo, restituisce la lista ‘record\_ids’.

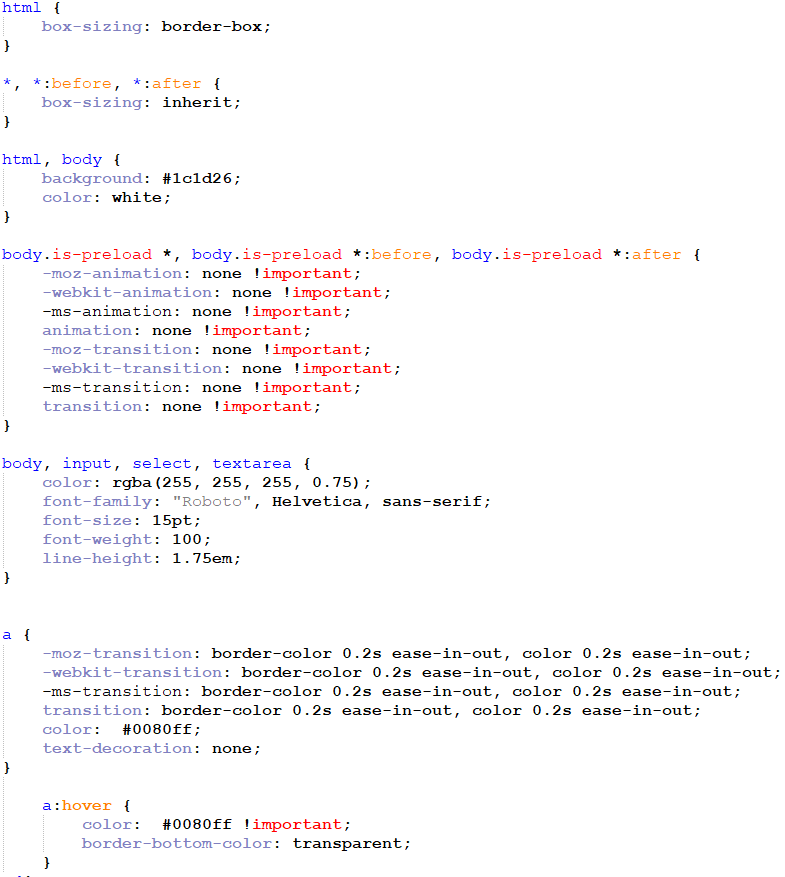
**default.css**

Figura : CSS di default

Il file default.css rappresenta il principale file di stile CSS della nostra applicazione. Esso definisce le regole di presentazione grafica e di formattazione del testo per tutte le view dell'applicazione.

## Arduino

### File ReadWrite

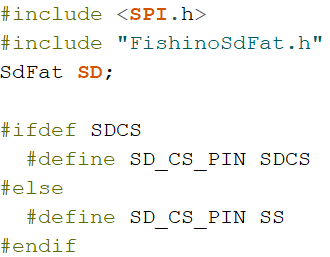
**Import e direttive**

Figura 23: Import e direttive

Il codice inizia importando le librerie "SPI.h" e "FishinoSdFat.h". Viene poi istanziata una classe "SdFat" chiamata SD. La riga successiva contiene una direttiva di precompilazione che definisce la costante "SD\_CS\_PIN" o a SDCS (Se presente) o a SS. La libreria "FishinoSdFat.h" è una versione personalizzata della libreria "SdFat.h" per poter funzionare con la scheda Fishino. Il codice in questo modo è in grado di gestire la lettura e la scrittura di dati sulla scheda SD montata sul Fishino.

**Scrittura del file**

Figura 24: Metodo setup

Il codice inizia con la dichiarazione di una variabile di tipo File chiamata myFile. Successivamente, nel setup, viene inizializzata la comunicazione seriale e viene eseguita la procedura di inizializzazione della scheda SD. Se la procedura non ha successo, viene stampato un messaggio di errore. Viene quindi creato un nuovo file di testo chiamato "config.txt" tramite la funzione SD.open(). Successivamente, il file viene rimosso (se esiste già) e quindi aperto nuovamente. Se l'apertura del file ha successo, viene costruita una stringa contenente una serie di valori separati da virgola. Questi dati di configurazione comprendono un nome di una rete Wi-Fi (MOD340), una password (passM340), un identificatore univoco (bad6f4d7-3c35-4d34-92c0-f18839a5372a), un URL per una API (Application Programming Interface) di registrazione dati, un indirizzo IP per un server remoto (164.128.168.41). Infine, il file viene chiuso e viene stampato un messaggio di successo o di errore a seconda del risultato dell'operazione.

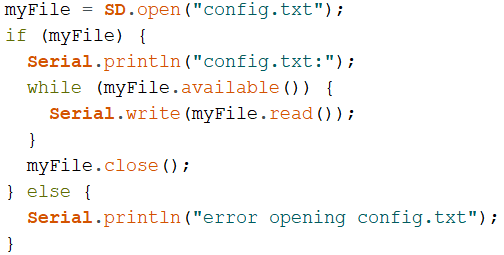
**Stampa del contenuto nel file**

Figura 25: Stampa contenuto file

Questo codice apre il file “config.txt” presente nel SD e se l’apertura è riuscita legge il contenuto del file e lo stampa sul terminale

### File Main

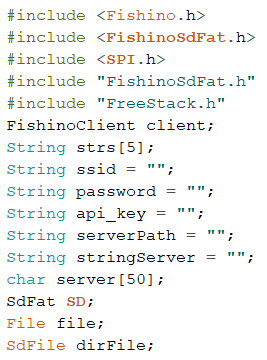
**Import e variabili globali**

Figura 26: Import e variabili globali

Il codice configura prima alcune variabili e librerie necessarie per l'utilizzo della scheda microSD e della connessione Wi-Fi tramite il modulo Fishino. In particolare, si importano le librerie "Fishino.h", "FishinoSdFat.h" e "SPI.h". Si definisce poi l'oggetto "client", di tipo "FishinoClient", e si usa successivamente per l'interazione con il server web. Vengono dichiarate tutte le variabili globali che serviranno all’interno del codice.

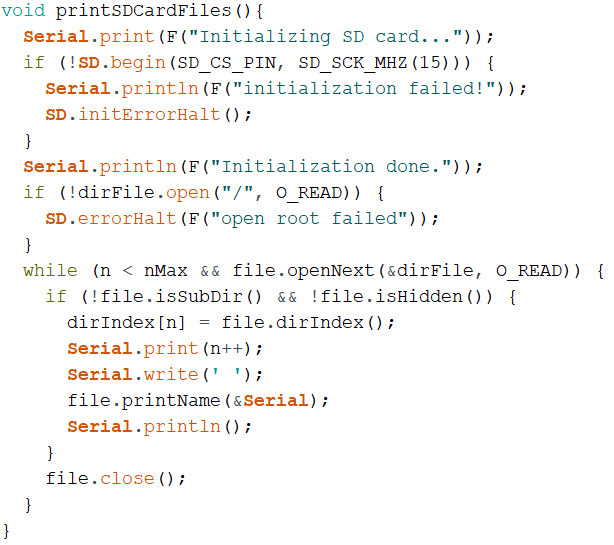
**Stampa informazioni file nel SD**

Figura 27: Metodo printSDCardFiles

All’interno di questa funzione viene effettuata l’inizializzazione della scheda SD e viene controllato che l’operazione sia stata eseguita con successo, altrimenti viene richiamata la funzione “SD.initErrorHalt()” che blocca l’esecuzione del programma. Per ogni file aperto viene salvato l'indice della directory corrente in cui si trova, viene poi stampato il nome del file sulla console seriale e viene incrementata la variabile "n". Infine, il file viene chiuso.

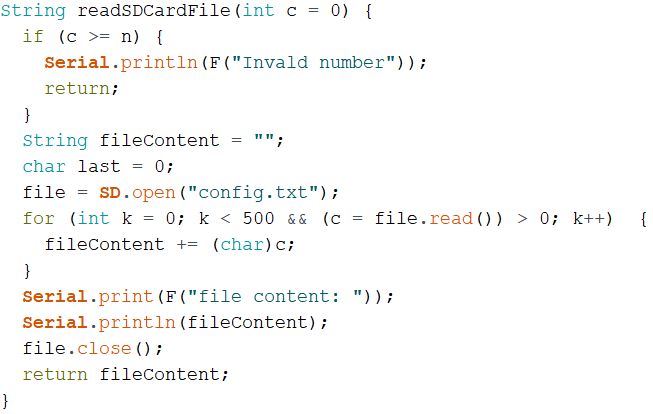
**Lettura del file nel SD**

Figura 28: Lettura del file SD

Questa funzione legge il contenuto del file chiamato “config.txt” dalla scheda SD fina ad un massimo di 500 caratteri e restituisce una stringa con il contenuto del file.

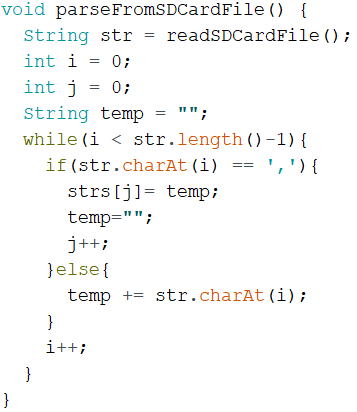
**Divisione dei contenuti**

Figura 29: Metodo parseFromSDCardFile

Questa funzione prende la stringa di ritorno del metodo “readSDCardFile” e divide i valori separati da virgola in elementi singoli e li inserisce dentro l’array di stringhe globale strs.

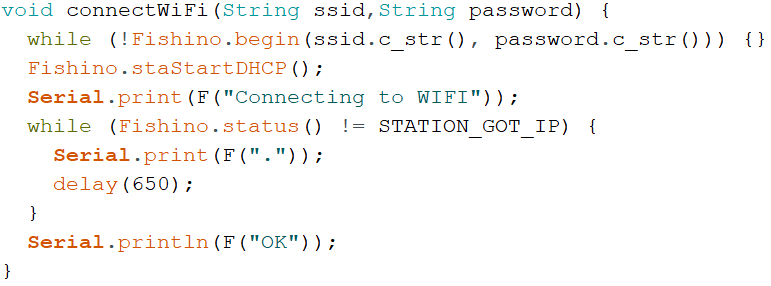
**Connessione al Wi-Fi**

Figura 30: Metodo connectWifi

Questa funzione si occupa di connettere la scheda Fishino ad una rete Wi-Fi utilizzando lo ssid e la password forniti come argomenti. Nella prima riga della funzione viene invocato il metodo "begin" della libreria Fishino, passando lo ssid e la password come argomenti, e viene utilizzato un loop while per attendere il completamento della connessione WiFi. Successivamente, viene chiamato il metodo "staStartDHCP" per inizializzare il client DHCP, il quale permette di ottenere un indirizzo IP in modo dinamico dalla rete WiFi. Infine, viene eseguita una serie di print sulla console seriale, mostrando l'avanzamento della connessione WiFi, finché lo status della connessione non diventa "STATION\_GOT\_IP" (ovvero quando l'IP viene assegnato alla scheda Fishino). Una volta completata la connessione, viene stampato un messaggio di "OK" sulla console seriale.

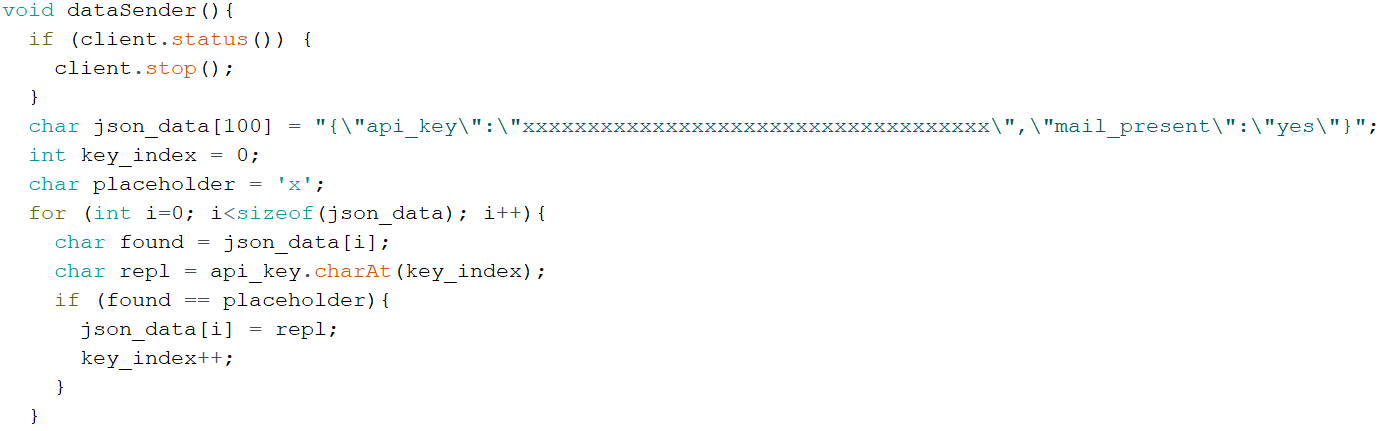
**Creazione json da inviare**

Figura 31: Metodo dataSender

Questo codice invia dati ad un server tramite una connessione client. Il formato dei dati inviati è un oggetto JSON che contiene un valore “api\_key” che deve essere qullo generato all’interno del sito. All’interno del ciclo quando il carattere trovato è ”x” lo rimpiazza con un carattere della chiave, abbiamo utilizzato questo metodo perché il concat delle stringhe non funzionava come doveva.

**Inoltro dei dati**

Figura 32: Inoltro dei dati

Questo codice è una funzione che esegue una richiesta POST ad un server HTTP. Utilizza la libreria Ethernet per stabilire una connessione al server e inviare un messaggio JSON tramite il protocollo HTTP utilizzando la porta 80. All'inizio, il codice controlla la connessione attraverso il metodo connect() della classe client. Se la connessione ha successo, il client stabilisce una connessione e procede con la richiesta. Viene quindi inviata la richiesta POST con POST serverPath HTTP/1.1, dove serverPath è l'URL del server al quale si vuole inviare la richiesta. Successivamente, vengono inviati alcune intestazioni tra cui Host, User-Agent e Content-Type che specificano il tipo di contenuto. Viene poi inviato il body della richiesta sotto forma di messaggio JSON. Infine, viene chiusa la connessione con il server, tramite il metodo stop().

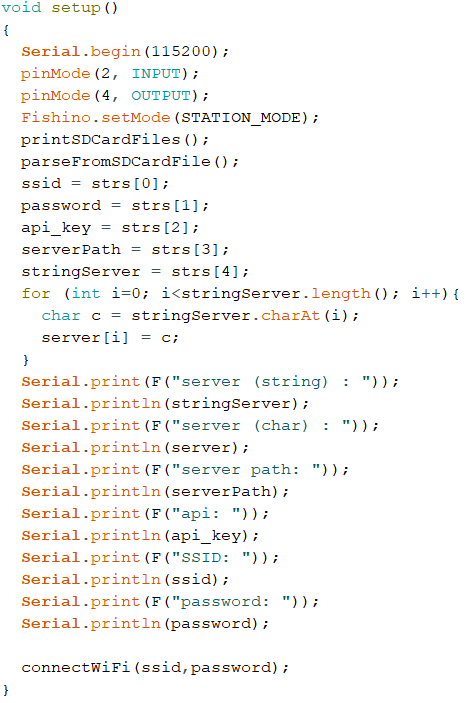
**setup()**

Figura 33: Metodo setup

Si inizia impostando la velocità della comunicazione seriale attraverso Serial.begin(115200). Viene poi dichiarato il pin 2 come input e il pin 4 come output attraverso pinMode(2, INPUT) e pinMode(4, OUTPUT). Viene poi impostata la modalità di connessione alla rete attraverso Fishino.setMode(STATION\_MODE). Dopodiché vengono richiamate le unzioni “printSDCardFiles()” e “parseFromSDCardFile()”. Vengono salvate le informazioni di strs all’interno di singole variabili globali, converto attraverso un for la stringa per il percorso del server da stringa ad array di caratteri, le stampo e infine richiamo il metodo per la connessione al Wi-Fi

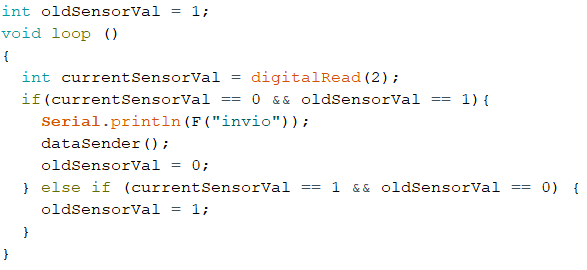
**loop()**

Figura 34: Metodo loop

Il codice inizia dichiarando una variabile "oldSensorVal" con valore iniziale di 1. Nella funzione "loop" viene poi letto il valore del sensore collegato al pin 2 e salvato nella variabile "currentSensorVal". Se il sensore passa da uno stato di alta a bassa attivazione (da HIGH a LOW), viene richiamata la funzione "dataSender". La variabile "oldSensorVal" viene poi aggiornata al nuovo stato del sensore (0). In alternativa, se il sensore passa da bassa a alta attivazione (da LOW a HIGH), l' "oldSensorVal" viene semplicemente aggiornato al nuovo stato (1).

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-01 | **Nome** | Trasmissione dei dati |
| **Descrizione** | Trasmissione dati dal Fishino al webserver | | |
| **Procedura** | 1. Inviare i dati dal Fishino 2. Riceverli sul server | | |
| **Risultati attesi** | Dati ricevuti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-002  REQ-02 | **Nome** | Raccolta dei dati |
| **Descrizione** | Raccolta dei dati trasmessi dal Fishino al webserver | | |
| **Procedura** | 1. Inviare i dati dal Fishino 2. Riceverli sul server 3. Salvare i dati | | |
| **Risultati attesi** | Dati salvati. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-003  REQ-03 | **Nome** | Hardware dispositivo |
| **Descrizione** | Hardware dispositivo completato | | |
| **Procedura** | 1. Montare l’hardware | | |
| **Risultati attesi** | L’hardware è completamente montato | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-004  REQ-04 | **Nome** | Salvataggio database |
| **Descrizione** | Salvataggio dei dati all’interno di un database | | |
| **Procedura** | 1. Registrare un nuovo utente | | |
| **Risultati attesi** | Utente registrato | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-005  REQ-05 | **Nome** | GUI web |
| **Descrizione** | Visualizzazione della GUI web | | |
| **Procedura** | 1. Aprire il sito web | | |
| **Risultati attesi** | La GUI è visualizzata correttamente | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-006  REQ-06 | **Nome** | Storico delle notifiche |
| **Descrizione** | Visualizzazione dello storico delle notifiche | | |
| **Procedura** | 1. Aprire la view “List” e “Calendar” dentro notifications | | |
| **Risultati attesi** | Le pagine sono visualizzate correttamente | | |

## Risultati test

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Nome** | **Stato** |
| TC-001 | Trasmissione dei dati | Passato |
| TC-002 | Raccolta dei dati | Fallito |
| TC-003 | Hardware dispositivo | Fallito |
| TC-004 | Salvataggio database | Passato |
| TC-005 | GUI web | Passato |
| TC-006 | Storico delle notifiche | Passato |

## Mancanze/limitazioni conosciute

Purtroppo all’interno del progetto abbiamo diverse mancanze, che sono:

* Mancanza della funzionalità di notifica di nuova posta tramite email. Il sito web non offre una funzione di notifica via email per i nuovi messaggi ricevuti.
* Mancanza della funzionalità di salvare i dati ricevuti dal server sul database.Il sito web non recupera i dati inviati tramite POST.
* Mancanza del Fishino montato nella cassetta della posta.

# Consuntivo

Rispetto alla pianificazione preventiva, nel Gantt consuntivo sono cambiati diversi tempi di diverse attività, questo era dovuto dal fatto che sono stati calcolati male alcuni tempi e l’importanza di alcune attività. Per questo motivo purtroppo non essendo stati nei tempi non siamo riusciti a finire il progetto

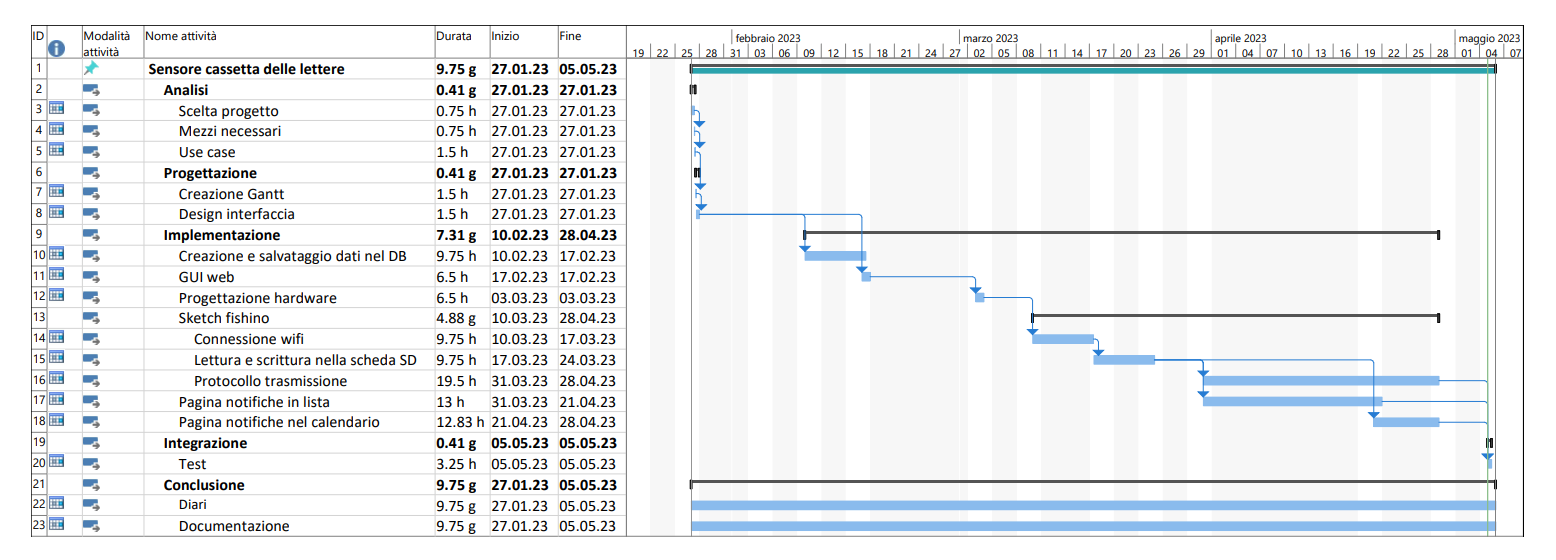


Figura 35: Gantt consuntivo

# Conclusioni

## Sviluppi futuri

Gli sviluppi futuri che abbiamo pensato per questo progetto sono:

* Completare la funzionalità di salvare i dati ricevuti tramite POST.
* Montare tutto l’hardware nella casetta delle lettere.
* Aggiunta della notifica tramite email quando arriva della posta.
* Esportare il sito web come applicazione per iOS e Android.
* Migliorare e rendere personalizzabile la GUI.

## Considerazioni personali

**Alex Ierardi**

Dopo questo progetto il mio bagaglio professionale è sicuramente aumentato siccome insieme al mio gruppo ci siamo ritrovati ad utilizzare applicazioni che non abbiamo mai utilizzato. Questo può essere una cosa positiva per quanto riguarda le nostre conoscenze personali, ma può essere anche un rallentamento per il progetto siccome ci siamo ritrovati a documentarci più del previsto. Dunque, mi è piaciuto svolgere questo progetto assieme ai mei colleghi ma avremmo dovuto utilizzare applicazioni che conosciamo.

**Alessandro Perri**

In conclusione, posso dire che questo progetto mi ha insegnato molto, sia a livello personale che a livello lavorativo. Sono contento di aver svolto questo progetto insieme ai miei compagni con cui mi sono trovato molto bene, ma allo stesso tempo sono deluso perché non siamo riusciti a terminare tutto quello che ci è stato richiesto. Mi sono divertito molto e sarebbe bello un giorno riuscire a terminarlo.

**Diego Muniz**

Posso dire che il progetto mi ha insegnato molto, ho imparato molto di come si fa e si porta avanti un progetto di questo peso sia sul lato teorico che pratico. Ho imparato come Usare web2py, come programmare con il Fishino e utilizzare le sue librerie, come usare FullCalendar, ecc.

# Bibliografia

## Sitografia

* <https://app.diagrams.net>, Draw.io, 03-02-2023
* <https://app.moqups.com>, Moqups, 03-02-2023
* <http://web2py.com/book>, Web2Py, 10-02-2023
* <https://fishino.it>, Fishino, 03-03-2023
* <https://docs.arduino.cc/software/ide-v1>, Arduino, 03-03-2023
* <https://fullcalendar.io/docs>, FullCalendar, 21-04-2023

# Indice delle figure

[Figura 1: Use-Case 6](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195831)

[Figura 2: Gantt preventivo 7](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195832)

[Figura 3: Schema E-R 9](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195833)

[Figura 4: Home page 10](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195834)

[Figura 5: Notifiche in lista 10](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195835)

[Figura 6: Notifiche in calendario 11](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195836)

[Figura 7: Login 11](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195837)

[Figura 8: Creazione account 12](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195838)

[Figura 9: Navbar 13](#_Toc134195839)

[Figura 10: Index.html 14](#_Toc134195840)

[Figura 11: Lista delle notifiche 15](#_Toc134195841)

[Figura 12: Notifiche nel calendario 16](#_Toc134195842)

[Figura 13: Log In 17](#_Toc134195843)

[Figura 14: Devices 18](#_Toc134195844)

[Figura 15. Metodo index 19](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195845)

[Figura 16: Metodo notif\_list 19](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195846)

[Figura 17: Metodo notif\_calendar 20](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195847)

[Figura 18: Metodo devices 21](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195848)

[Figura 19: Metodo api 22](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195849)

[Figura 20: Metodo get\_device\_id 23](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195850)

[Figura 21: Metodo save\_recorded\_value 23](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195851)

[Figura 22: CSS di default 24](#_Toc134195852)

[Figura 23: Import e direttive 25](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195853)

[Figura 24: Metodo setup 26](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195854)

[Figura 25: Stampa contenuto file 27](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195855)

[Figura 26: Import e variabili globali 28](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195856)

[Figura 27: Metodo printSDCardFiles 29](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195857)

[Figura 28: Lettura del file SD 30](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195858)

[Figura 29: Metodo parseFromSDCardFile 30](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195859)

[Figura 30: Metodo connectWifi 31](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195860)

[Figura 31: Metodo dataSender 32](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195861)

[Figura 32: Inoltro dei dati 33](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195862)

[Figura 33: Metodo setup 34](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195863)

[Figura 34: Metodo loop 35](file:///\\Cpt.local\shares\homes\allievi\SAM\IN\_folder-redir\alessandro.perri\Desktop\Documentazione%20-%20Sensore%20cassetta%20delle%20lettere.docx#_Toc134195864)

# Allegati

* Mandato e/o QdC.
* Prodotto.
* Diari di lavoro.
* Codici sorgente/documentazione.
* Use-case.
* Gui.
* Gantt preventivo.
* Gantt consultivo.