

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA  
CAMPUS DI CESENA

---

Scuola di Scienze  
Corso di Laurea in Ingegneria e Scienze Informatiche

MONITORAGGIO ATTIVO  
DI UN  
CLUSTER DI RASPBERRY PI

*Elaborato in*  
PROGRAMMAZIONE DI RETI

*Relatore*  
GABRIELE D'ANGELO

*Presentata da*  
STEFANO FALZARESI

---

Seconda Sessione di Laurea  
Anno Accademico 2015 – 2016



*Se vuoi qualcosa vai e inseguila...*



# Indice

<b>Introduzione</b>	<b>vii</b>
<b>1 Analisi</b>	<b>1</b>
1.1 Descrizione del problema . . . . .	1
1.2 Valutazione di mercato . . . . .	2
1.2.1 Site24x7 . . . . .	2
1.2.2 Nagios . . . . .	4
1.2.3 Zabbix . . . . .	5
1.2.4 Valutazione . . . . .	7
1.3 Scelta della soluzione . . . . .	7
<b>2 Struttura della Soluzione Proposta</b>	<b>9</b>
2.1 Progettazione . . . . .	10
2.2 Core . . . . .	11
2.3 Polling . . . . .	14
2.4 Comunicazione . . . . .	15
2.5 GUI . . . . .	17
2.5.2 Homepage . . . . .	18
2.5.3 Admin Page . . . . .	19
<b>3 Implementazione</b>	<b>23</b>
3.1 Core . . . . .	23
3.1.1 Interfaccia Web . . . . .	23
3.1.2 API Manager . . . . .	25
3.2 Polling e Comunicazione . . . . .	30
3.2.1 Raspberry Pi . . . . .	30
3.2.2 Server . . . . .	31
3.3 GUI . . . . .	37
3.3.1 Scripts . . . . .	38
3.3.2 Stile . . . . .	43
3.4 Accessibilità . . . . .	46

<b>4</b>	<b>Valutazioni Finali</b>	<b>47</b>
4.1	Analisi dei Risultati . . . . .	47
4.2	Considerazioni . . . . .	47
4.3	Sviluppi futuri . . . . .	48
	<b>Ringraziamenti</b>	<b>49</b>
<b>A</b>	<b>Elenco API</b>	<b>51</b>
<b>B</b>	<b>Guida all'installazione</b>	<b>63</b>
	<b>Sitografia</b>	<b>73</b>

# Introduzione

All'interno del campus di Cesena dell'università Alma Mater Studiorum è stato sviluppato un esempio di cluster computing basato sull'uso di 64 Raspberry Pi interconnessi tra loro tramite connessione ethernet e disposti all'interno di una struttura composta da mattoncini Lego a creare delle torri colorate. Infrastrutture di questo tipo richiedono che venga effettuata una rilevazione costante dello stato dei dispositivi che le costituiscono così da poter individuare tempestivamente possibili malfunzionamenti o un carico di lavoro non adeguato.

Nel corso di questa tesi verranno inizialmente analizzati i requisiti che un sistema di monitoraggio per cluster di questo tipo deve possedere e le funzionalità che dovrà mettere a disposizione dell'amministratore e degli utenti che vi dovranno interagire, in funzione ad essi si effettuerà una scelta della soluzione che verrà ritenuta più adatta allo svolgimento dei compiti richiesti.

La proposta che si deciderà di adottare verrà analizzata nel dettaglio così da identificare le caratteristiche fondamentali della struttura, il funzionamento delle parti che la compongono e la motivazione alla base di alcune delle scelte tecnologiche effettuate.

Nella parte conclusiva della trattazione ogni modulo sviluppato verrà attentamente sezionato ed analizzato. Per ogni sezione verranno indicati i linguaggi di programmazione utilizzati, la motivazione alla base delle scelte effettuate ed estratti implementativi che maggiormente caratterizzano la parte esposta.

A corredo sono state aggiunte a questa tesi due appendici, nella prima sono indicate tutte le API messe a disposizione dal sistema con una spiegazione delle funzionalità di ognuna ed una indicazione dei dati necessari al corretto funzionamento; la seconda invece contiene una guida all'installazione del sistema con indicazione dei prerequisiti richiesti e dei passi necessari per la messa in opera.



# Capitolo 1

## Analisi

La trattazione di questo capitolo sarà volta alla definizione del problema, alla ricerca ed alla conseguente selezione della soluzione più consona alle necessità espresse.

### 1.1 Descrizione del problema

Il dipartimento di Informatica Scienza e Ingegneria dell'Università di Bologna con il supporto finanziario di Ser.In.Ar da agosto 2013 <sup>[1]</sup> ha ideato, progettato ed in seguito prodotto un cluster di calcolo denominato “Raspèin”. Per lo sviluppo si è scelto, a differenza di altri cluster, di utilizzare come nodi dei Raspberry Pi Model B ver. 2 (da qui il nome Raspèin) così da avere un sistema compatto e dai costi contenuti con il quale però gli studenti potessero effettuare delle sperimentazioni reali su una struttura di calcolo distribuita. Inizialmente il sistema era composto da 32 nodi che sono poi stati raddoppiati così da ottenere un cluster formato da 64 Raspberry suddivisi in varie torri composte da mattoncini Lego<sup>®</sup>. Ora che il cluster Raspèin è completamente operativo e funzionante si è palesata la necessità di dotarlo di un sistema che ne permetta il monitoraggio in tempo reale, compito di questa tesi sarà quindi la ricerca, la selezione e l'implementazione di una soluzione in grado di risolvere questa problematica. D'accordo con il docente si è deciso di focalizzare la ricerca su alcuni aspetti fondamentali che la soluzione proposta dovrà obbligatoriamente possedere:

- Basso impatto prestazionale sui dispositivi periferici
- Possibilità di assegnare un range di valori “ammessi” per ogni sensore
- Possibilità di avere un sistema di alerting qualora venga rilevato un valore al di fuori del range indicato

- Avere una visualizzazione “mobile friendly”
- Permettere l’interfacciamento con applicazioni mobili native
- Costi di gestione contenuti

## 1.2 Valutazione di mercato

Una volta definiti i criteri si è avviata una ricerca di mercato per identificare i principali sistemi di monitoraggio disponibili, tra i risultati ottenuti è quindi stata fatta un’ulteriore selezione mantenendo solamente le soluzioni aventi le caratteristiche richieste dai requisiti esposti; si è deciso di valutare sia software a pagamento che software gratuiti analizzando innanzitutto le funzionalità offerte e valutandone poi in seguito l’impatto economico. Le soluzioni più interessanti verranno ora analizzate con attenzione per definirne pregi e difetti relativamente alla specifica implementazione.

### 1.2.1 Site24x7

#### **Site24x7**<sup>[2]</sup>

##### Caratteristiche

Site24x7<sup>®</sup> è un sistema di monitoraggio offerto da Zoho Corporation; questa è una soluzione di tipo cloud che non richiede quindi l’installazione di un server centralizzato presso la rete da monitorare, è una suite in abbonamento con varie soglie disponibili a partire da un minimo di 9\$ mensili (permettendo il monitoraggio fino ad un massimo di 10 dispositivi). Il sistema si presenta sin da subito molto completo ed articolato, al primo avvio una procedura guidata porta alla messa in monitoraggio di un host che potrà essere un Server o una risorsa web. La tipologia server richiede che sul dispositivo da monitorare sia installato un agent di connessione, questo, attraverso un codice univoco, effettuerà il binding con il nostro account che inizierà da quel momento a ricevere con periodicità definibile i dati dal dispositivo, l’agent è presente per la maggior parte dei sistemi operativi e, per effettuarne l’installazione, non sono necessarie particolari abilità.

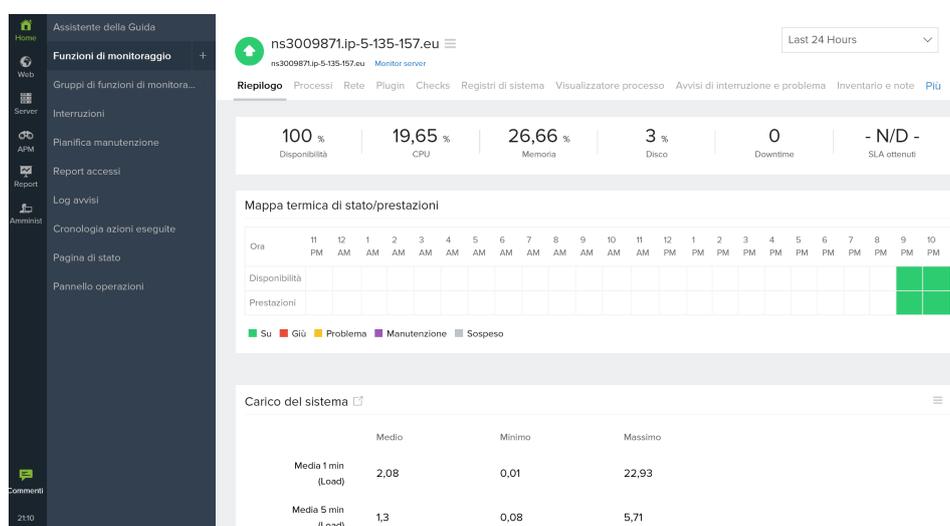


Figura 1.1: Pannello di controllo Site24x7

Il supporto tecnico al servizio è garantito da un call center sempre attivo, da dei webinar gratuiti e da delle guide molto complete che rendono inoltre la configurazione del sistema estremamente facile. Come da richieste questa soluzione permette di configurare delle soglie di allerta per alcune grandezze fondamentali quali il carico della CPU, lo spazio su disco o la memoria occupata, qualora vengano oltrepassate si riceverà una notifica attraverso Twitter, email, SMS o Messaggio istantaneo. Site24x7 mette inoltre a disposizione dei propri clienti un'applicazione mobile (disponibile su piattaforme iOS e Android) attraverso la quale è possibile accedere al proprio account e monitorare i dispositivi connessi.

## Valutazione

Dalle verifiche e dai test effettuati su questo strumento ho potuto constatare che l'offerta è molto completa e permette di rispondere a molti dei requisiti richiesti. Nonostante questo esistono però delle problematiche legate alla particolare natura delle nostre necessità, il problema principale di questa tipologia di servizio è rappresentato dal fatto che per collezionare i dati viene utilizzato un demone che, a volte, ha richiesto un utilizzo di CPU che si è attestato intorno al 5% su un server con una potenza di calcolo superiore a quella dei Raspberry che dovremmo andare ad analizzare; inoltre questo sistema essendo basato su servizi in cloud richiede che i dispositivi monitorati facciano costantemente delle comunicazioni tramite Internet andando ad incidere in questo modo sul traffico generato dalla rete. In ultimo una delle richieste era poter monitorare varie tipologie di grandezze sui dispositivi quali ad esempio la tem-

peratura del sistema, su Site24x7 questa tipologia di controllo non è presente e sarebbe quindi necessario svilupparla, da zero.

## 1.2.2 Nagios

### **Nagios**<sup>[3]</sup>

#### Caratteristiche

Nagios<sup>®</sup> è un prodotto sviluppato dalle Nagios Enterprises e basato sull'architettura Nagios Core fornita in licenza GPLv2 (*GNU General Public License*), l'azienda produttrice ha deciso di fornire due diverse tipologie di servizio, una completamente gratuita attraverso Nagios Core ed una a pagamento attraverso Nagios XI. Le differenze tra le due versioni sono irrilevanti nell'ottica di questo progetto e quindi risulta essere sufficiente analizzare le funzionalità messe a disposizione da Nagios Core. Per il funzionamento del sistema è indispensabile la presenza di un computer con sistema operativo Linux che fungerà da server web e server applicativi, l'installazione risulta essere relativamente facile<sup>[4]</sup> ma richiede comunque un minimo di conoscenza di Linux e dei suoi comandi principali; in circa 20 minuti il sistema dovrebbe essere pronto per essere poi configurato. Tutta la configurazione è basata su dei file di testo nei quali vengono indicati i vari host (che possono anche essere suddivisi in gruppi), i servizi da monitorare su ognuno di essi e il metodo di notifica da utilizzare nel qual caso avvenga un evento problematico.

```
define host{
    use         generic-host      ; Inherit default values from a template
    host_name   remotehost       ; The name we are giving to this host
    alias       Some Remote Host ; A longer name associated with the host
    address     192.168.1.50     ; IP address of the host
    hostgroups  allhosts        ; Host groups this host is associated with
}
define command{
    name        check_http
    command_name check_http
    command_line $USER1$/check_http -I $HOSTADDRESS$ $ARG1$
}
```

Listato 1.1: Esempio di file di configurazione

Questa soluzione mette a disposizione dei propri utenti una moltitudine di plugin che permettono l'utilizzo di particolari funzionalità non nativamente im-

plementate, uno di questi è ad esempio NRPE (*Nagios remote plugin executor*, fondamentale per il controllo di sistemi remoti (nel caso specifico i Raspberry).

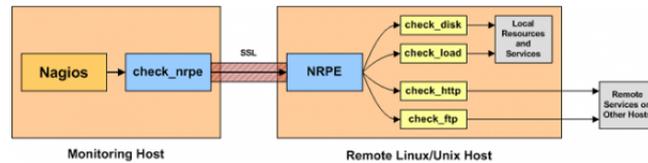


Figura 1.2: funzionamento NRPE<sup>[5]</sup>

Tra i vari plugin però non è stato possibile trovare qualcosa che permetta la comunicazione attraverso twitter e dispositivi mobili.

## Valutazione

La natura open source del progetto lo ha reso inizialmente e per vari anni uno dei software maggiormente utilizzati, la community molto attiva ha inoltre permesso la creazione di molti plugin capaci di coprire svariate necessità, negli ultimi due anni però la mancanza di nuove funzionalità e di un restyling grafico hanno portato questo software a perdere sempre più quote di mercato fino ad attestarsi ad un 20%<sup>[6]</sup> circa e vedendo la conseguente nascita ed affermazione di soluzioni più innovative. Allo stato di sviluppo attuale questo software non mette a disposizione dei plugin funzionanti per l'utilizzo di Twitter o per il monitoraggio del sistema su dispositivi mobili; non riuscendo quindi a rispondere completamente a tutte le richieste effettuate sarebbe necessario da parte nostra implementare tali funzionalità all'interno di nuovi plugin ma ciò richiede l'acquisizione di specifiche competenze in merito ed un considerevole tempo di sviluppo.

### 1.2.3 Zabbix

**ZABBIX**<sup>[7]</sup>

#### Caratteristiche

Zabbix<sup>®</sup> è un software sviluppato dalla Zabbix SIA dagli inizi del 2005, questo prodotto viene rilasciato con licenza GPLv2 ma, se utilizzato a scopi commerciali, il produttore invita ad acquistare un servizio di supporto a pagamento. È un sistema di tipo centralizzato che richiede quindi un server raggiungibile dai dispositivi oggetto del monitoraggio. L'installazione del server

principale richiede una piattaforma LAMP (*Linux Apache MySQL Phpmyadmin*) basata su una qualsiasi distribuzione Linux, per agevolare l'installazione sono resi disponibili un Live CD o una macchina virtuale preconfigurata anche se il fornitore consiglia quest'ultima tipologia di approccio solo per effettuare dei test e non per la vera messa in produzione. Al termine dell'installazione la configurazione del sistema può essere effettuata interamente attraverso l'interfaccia web di Zabbix, per compiere questa fase non è disponibile una vera e propria procedura guidata ma, seguendo il manuale disponibile online<sup>[8]</sup>, i passi per la creazione di un host e dei relativi "sensori" non risultano essere estremamente complessi ed è inoltre possibile, una volta terminata la configurazione del primo dispositivo, creare un layout personalizzato da poter poi esportare per la creazione dei successivi. La comunicazione tra server e dispositivi può essere fatta in modo passivo con il server che attende i dati dagli altri host o attivo con il server che invia ad uno o più host le richieste di un determinato valore; la comunicazione server-client può avvenire attraverso un agent preventivamente installato sui client (che poi rimarrà continuamente in esecuzione) o "agentless" attraverso dei trap snmp (*simple network management protocol*). Per ogni risorsa monitorata è possibile definire delle soglie che, nel caso in cui vengano oltrepassate, mettono in moto la procedura di alerting che si occuperà di avvisare l'amministratore della rete dell'avvenuto guasto.

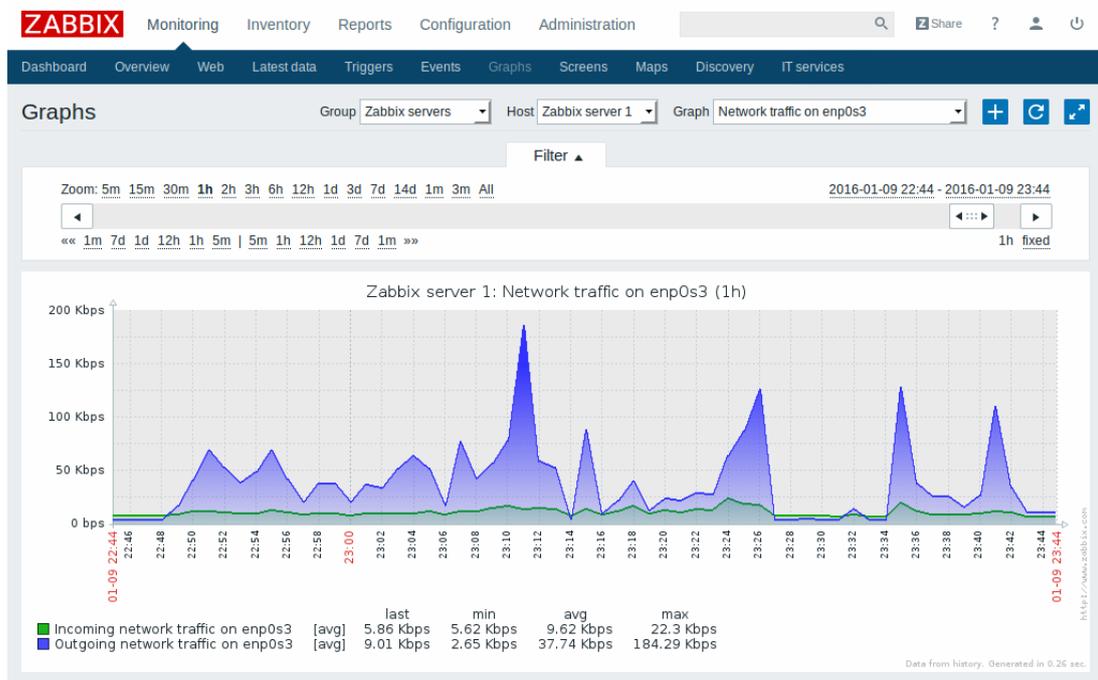


Figura 1.3: Dashboard di Zabbix

Le tipologie di notifica offerte nativamente sono: email, SMS (richiede un modem GSM), Jaber o Ez Texting, è però possibile definire degli script personalizzati attraverso linguaggio bash (*bourne again shell*) o python. Non è presente un'applicazione ufficiale per smartphone o tablet ma, grazie alle API (*application programming interface*) messe a disposizione, varie soluzioni di terze parti sono comunque ottenibili dagli application store.

### 1.2.4 Valutazione

Zabbix è tra le migliori soluzioni gratuite presenti sul mercato, una grafica moderna unita ad una rapida curva d'apprendimento hanno portato questo prodotto ad essere una delle vere alternative a Nagios sul quale sta ormai prendendo il sopravvento. Grazie alla sua comunità molto attiva è possibile trovare svariati template già fatti per il monitoraggio di particolari dispositivi o per aggiungere funzionalità al sistema. Nonostante ciò però non sono ancora attualmente disponibili un sistema di alerting attraverso twitter o un sistema per il monitoraggio della temperatura di sistema; probabilmente è possibile implementare queste funzionalità grazie all'utilizzo degli alert script personalizzati e delle API disponibili ma questo richiederebbe ovviamente del tempo sia per lo studio del sistema che per la creazione ed il testing di quanto necessario.

## 1.3 Scelta della soluzione

Dall'analisi effettuata sulle tre soluzioni sopra elencate quella con la maggior parte delle caratteristiche richieste risulta essere Site24x7 che però a fronte di un costo non indifferente di 780\$ annui non fornisce tutte le funzionalità richieste ed inoltre la tipologia di comunicazione basata sul continuo scambio di dati attraverso la rete Internet collide con due delle basi fondamentali su cui si basa questo progetto, il basso impatto sulle prestazioni di ogni singolo Raspberry e lo spreco minimo di banda. Tutte le altre soluzioni invece non rispondendo completamente alle richieste fatte richiederebbero, da parte di chi implementerà il sistema, un impegno non indifferente per l'apprendimento approfondito dello stesso così da poter aggiungere, mantenere ed adattare nel tempo alle mutevoli necessità i moduli mancanti. Queste conoscenze acquisite andrebbero sì ad accrescere il bagaglio di competenza di chi le andrà a studiare ma risulterebbero poco spendibili nel futuro trasformando così il tempo utilizzato quasi in un tempo sprecato. Per tutte queste motivazioni credo che, nonostante la grande quantità di funzionalità disponibili, nessuna delle proposte attualmente presenti sul mercato possa ritenersi adatta ad effettuare il monitoraggio del cluster e ritengo quindi che, sia a livello di formazione

che a livello di future implementazioni, sia preferibile sviluppare una soluzione personalizzata che abbracci completamente le necessità precedentemente descritte.

## Capitolo 2

# Struttura della Soluzione Proposta

Questo capitolo tratterà in dettaglio la soluzione selezionata evidenziandone le funzionalità e la composizione. Verrà effettuata una suddivisione per moduli ognuno dei quali verrà approfondito mostrandone funzionalità e struttura.

## 2.1 Progettazione

Come visto precedentemente le necessità molto specifiche che il progetto possiede ci hanno portato a scegliere di sviluppare una soluzione completamente personalizzata piuttosto che adattare un software già sul mercato, i due pilastri portanti che hanno guidato l'intera fase di progettazione sono stati l'uso limitato di risorse sui dispositivi periferici e la possibilità di espandibilità futura. Per fare ciò si è quindi scelto di ideare un sistema che avesse un Core principale al quale poter collegare uno o più moduli, il dialogo tra Core e moduli periferici verrà poi effettuato utilizzando le più moderne tecnologie web. Questo tipo di struttura è pensata per essere la base da cui partire per implementare di volta in volta sempre nuove funzionalità permettendo però di riutilizzare ciò che era stato precedentemente fatto. In questa relazione verrà trattata la prima versione del sistema, i moduli sviluppati per questa versione sono i seguenti:

**Core:** immagazzina i dati ricevuti e li scambia con gli altri moduli

**Polling:** acquisisce i dati dai dispositivi li verifica e li invia al Core

**Comunicazione:** si occupa della comunicazione tra sistema utenti ed amministratori

**GUI:** rende il progetto accessibile all'utente finale

Attraverso la Figura 2.1 viene mostrata l'organizzazione di questi moduli e l'interazione presente tra di essi, è possibile notare come il Core sia al centro dell'intero sistema e come quindi gli altri moduli periferici dialoghino quasi esclusivamente con esso.

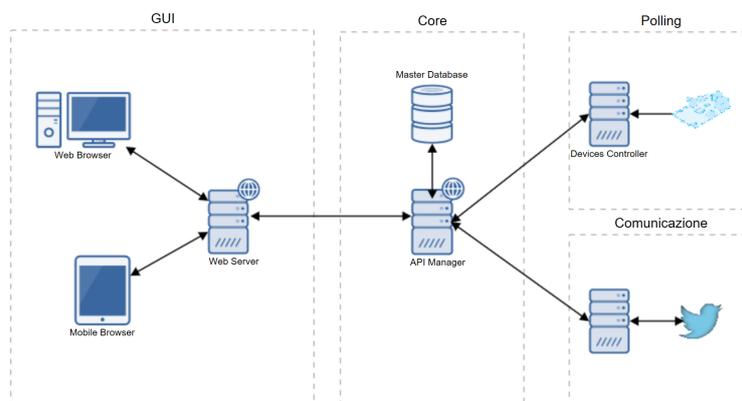


Figura 2.1: Struttura del progetto

La struttura a moduli utilizzata permette inoltre di suddividere le funzionalità tra vari dispositivi o server, questa tipologia di approccio permette di garantire un'elevata scalabilità e flessibilità del sistema; per molte parti inoltre è possibile effettuare una ulteriore segmentazione interna permettendo così di sfruttare ad esempio servizi già presenti sulla rete di destinazione (ad esempio potrebbe essere riutilizzato un server dei database o un server web). Per meglio analizzare e definire l'intero sistema entreremo ora più nel dettaglio delle quattro parti evidenziate.

## 2.2 Core

Il Core rappresenta la base di tutto il sistema ed ogni altro modulo per poter funzionare deve interagire con esso, le sue funzioni principali sono quindi immagazzinare i dati, organizzarli e scambiarli con le altre parti del sistema in modo semplice e standardizzato. Volendo permettere l'utilizzo di questo modulo anche a servizi di terze parti (quali ad esempio un'applicazione mobile) si è deciso di dotarlo di un sistema di autenticazione basato su dei token e, ad ogni richiesta di interazione, ogni client dovrà quindi allegare anche il proprio token univoco; ad ognuno di essi è assegnato inoltre un livello permettendo così di inibire l'accesso ad una o più risorse. In Figura 2.2 viene delineata la struttura del Core e le due parti fondamentali che lo compongono:

**API Manager:** riceve ed interpreta le richieste

**Authenticator:** verifica le credenziali fornite e il livello delle stesse

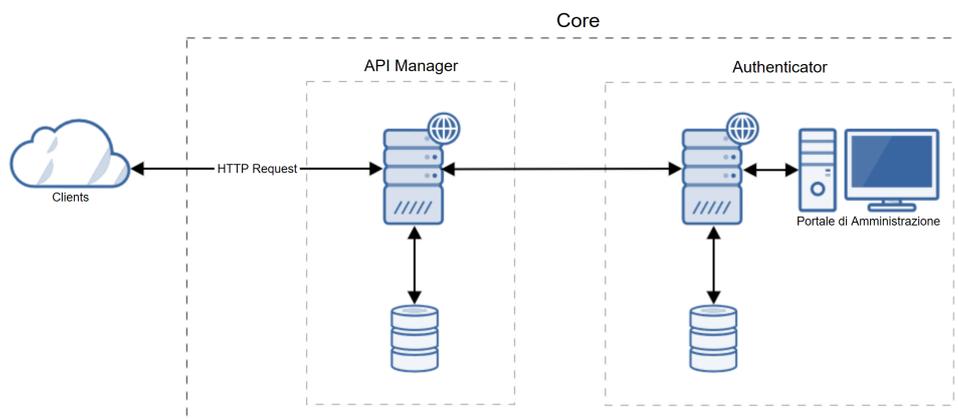


Figura 2.2: Struttura interna del Core

### 2.2.1 Funzionamento

Attraverso il protocollo HTTP (*Hypertext Transfer Protocol*) i client interrogano le REST API (*Representational State Transfer*) messe a disposizione dall'API Manager, le tipologie di richiesta supportate sono:

**GET** per la lettura di dati

**POST** per la scrittura di dati

Per questa specifica installazione si è scelto di effettuare lo scambio di dati attraverso il protocollo HTTP che risulta essere non sicuro in quanto lo scambio di dati che avviene tra client e server è totalmente in chiaro, sarà quindi necessario a livello di firewall di rete imporre delle limitazioni al traffico che può essere instradato verso il server che farà da API Manager così che la comunicazione avvenga tra due dispositivi all'interno di una rete trusted (che potrà essere una rete locale o una rete creata attraverso una VPN (*Virtual private network*)).

In alternativa, nel caso in cui si decida di rendere tale servizio totalmente accessibile anche dall'esterno sarà necessario utilizzare un protocollo maggiormente sicuro come ad esempio HTTPS (*Hypertext transfer protocol over secure socket layer*) che prevede la trasmissione di pacchetti HTTP attraverso un tunnel crittografato, per fare ciò sarà necessario configurare in modo adeguato il server web che fornisce le API e acquistare da delle CA (*Certification Authority*) un certificato digitale che ne attesti l'identità.

In Figura 2.3 viene mostrato il diagramma di sequenza relativo alla ricezione di una richiesta; appena questa viene ricevuta l'API Manager ne verifica la tipologia ed in seguito passa la chiave di autenticazione e il nome dell'API richiesta all'authenticator che, con i dati forniti, interroga il proprio database e verifica che la chiave fornita risulti correttamente censita e che non sia stata revocata, se questo controllo ha esito positivo viene verificato quindi, sempre attraverso un'interrogazione al database, il livello di autenticazione necessario per usufruire dell'API, se vi è corrispondenza tra il livello della chiave e quello necessario all'uso della funzione l'esito positivo verrà inviato alla sessione dell'API manager che aveva effettuato la chiamata il quale lo interpreterà e di conseguenza valuterà la successiva operazione da svolgere.

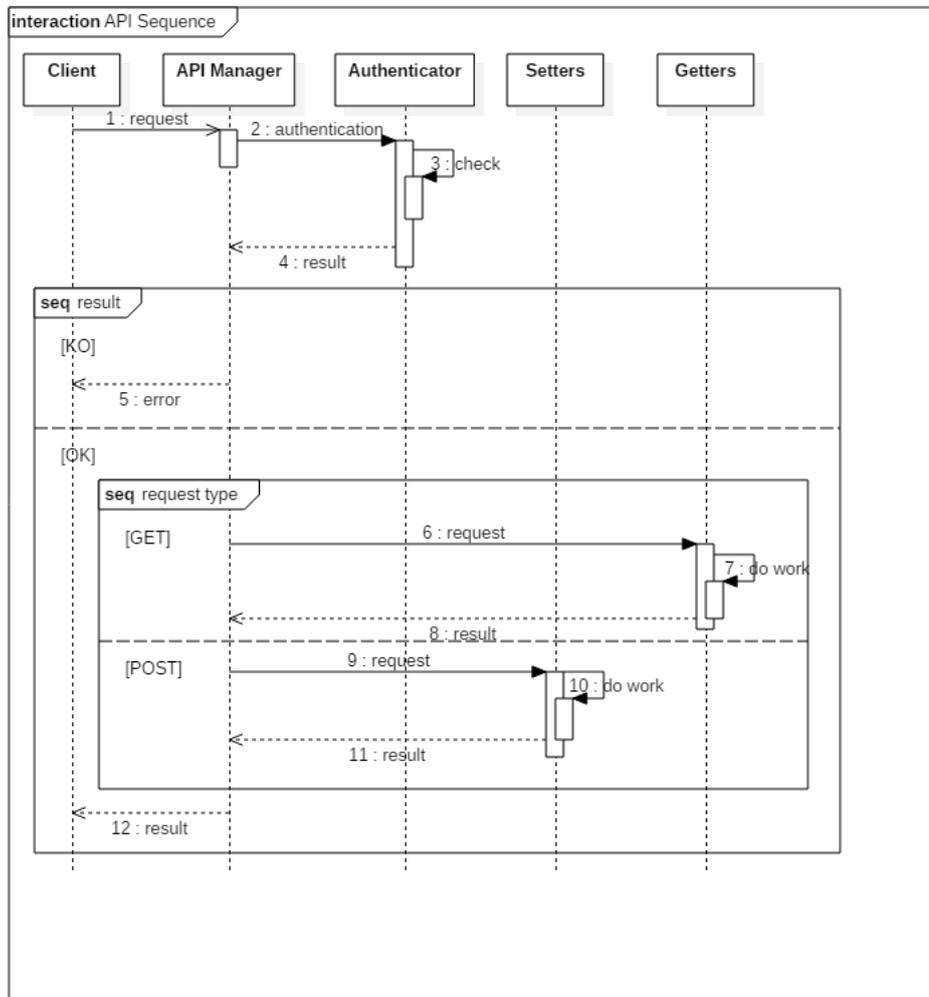


Figura 2.3: Diagramma di sequenza, ricezione di una richiesta

Al fine di permettere il corretto funzionamento di questo servizio è necessario che l'amministratore del sistema possa gestire tutti i token di autenticazione messi a disposizione degli utilizzatori, per fare ciò è stato creato un apposito portale web attraverso il quale sarà quindi possibile crearli e modificarli; come è possibile notare dalla Figura 2.4 in fase di creazione vengono richiesti esclusivamente una descrizione ed un livello da assegnare alla chiave, se il processo termina in modo corretto verrà mostrata la chiave alfanumerica generata automaticamente ed utilizzabile per l'autenticazione alle API.

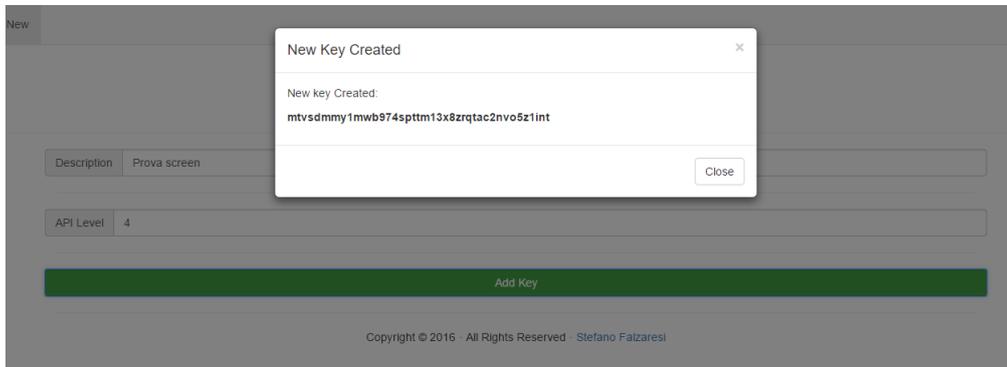


Figura 2.4: Nuova chiave correttamente generata

Vista la delicatezza di questa procedura si è deciso di dotare di un ulteriore livello di sicurezza le operazioni che prevedono una interazione con il database, ogni funzione sviluppata identifica quindi prima di effettuare qualsiasi operazione, la pagina di provenienza della chiamata e verifica che tale pagina appartenga ad un referer valido, in caso contrario la computazione verrà interrotta.

## 2.3 Polling

Lo scopo principale del modulo di polling è la raccolta ed interpretazione dei dati provenienti dai dispositivi e il successivo invio al Core. Come mostrato in Figura 2.5 gli elementi che compongono questo modulo sono due:

**Controller:** interagisce con Core e dispositivi

**Dispositivi:** generano file di diagnostica

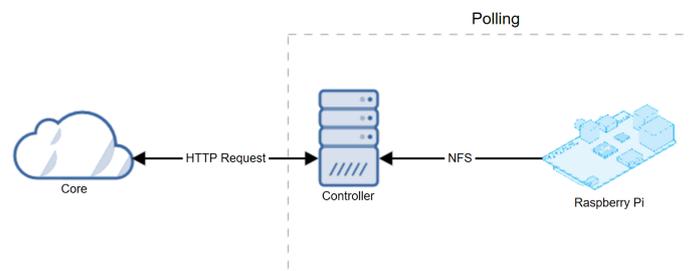


Figura 2.5: Struttura del modulo di polling

### 2.3.1 Funzionamento

Il controller avvia il modulo e richiede al Core i dati di configurazione necessari per poter essere operativo come ad esempio la cartella da cui dovranno essere prelevati i dati dei dispositivi. Questi ultimi invece, avendo risorse limitate, non dialogano direttamente con il Core e quindi all'interno del codice sviluppato contengono già tutte le informazioni necessarie a rilevare in modo periodico i valori dei sensori presenti e esportare questo dato all'interno del controller; attraverso protocollo NFS (*Network file system*), i dati rilevati sono inseriti, unitamente all'identificativo del dispositivo, all'interno di un file opportunamente formattato. Ad intervalli prefissati il controller verifica all'interno della cartella comunicatagli in fase di avvio se sono presenti dei nuovi file e, se così fosse, li analizza singolarmente effettuandone il parsing ed identificando il Raspberry che lo ha generato e le grandezze presenti al suo interno, una volta verificata online l'effettiva esistenza del dispositivo i dati rilevati vengono inviati al Core che li renderà poi fruibili agli altri moduli.

## 2.4 Comunicazione

Questo modulo svolge tutte le operazioni di controllo e comunicazione del sistema, l'unico elemento che ne compone la struttura è il controller il quale richiede i dati al Core e li analizza per poi valutare se è necessario effettuare l'invio di una comunicazione.

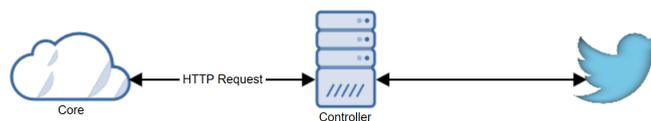


Figura 2.6: Infrastruttura di comunicazione

### 2.4.1 Funzionamento

Come il modulo di polling anche quello di comunicazione all'avvio contatta il Core per prendere tutti i dati a lui necessari per l'interfacciamento, nella fattispecie vengono importati tutti i dati necessari all'invio di messaggi tramite Twitter tra i quali è presente anche il nickname dell'amministratore di sistema. Le funzionalità specifiche messe a disposizione da questo modulo sono tre:

**invio messaggi di errore** all'amministratore

**risposta a query specifiche**

## aggiornamento dello stato

L'invio di messaggi di errore all'amministratore viene effettuato al rilevamento di un valore non corretto per uno o più sensori presenti su un Raspberry, per fare ciò il modulo contatta in modo periodico il Core richiedendo i dati di tutti i Raspberry a cui risulta collegato e per ognuno analizza i dati dei sensori attivi, se il range indicato non è corretto viene inviato un messaggio attraverso le API fornite da Twitter e, nello stesso istante, viene pubblicato anche un tweet sulla pagina del sistema.

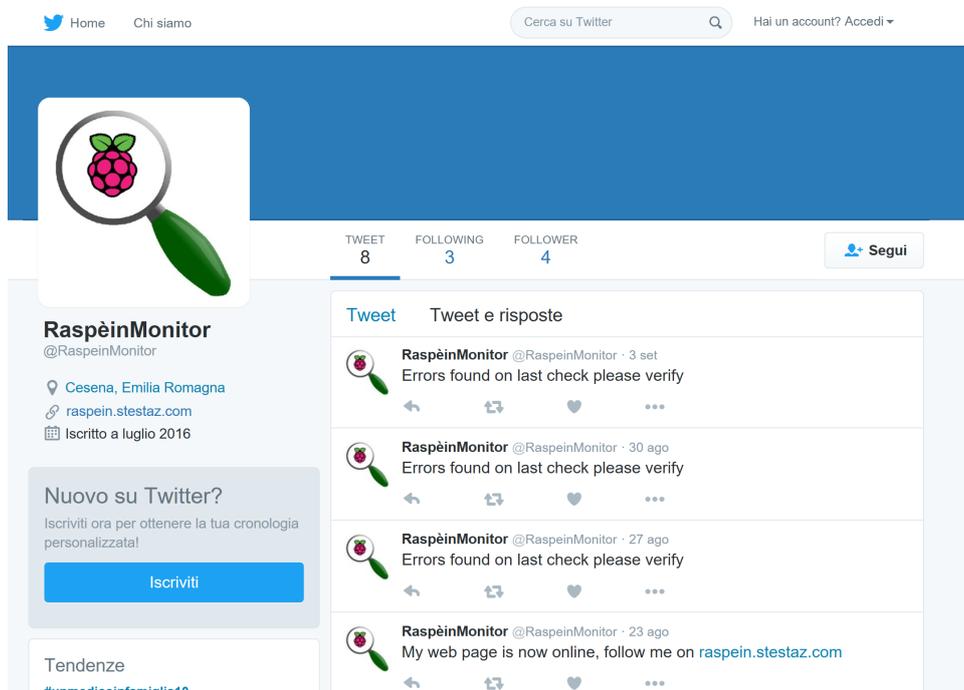


Figura 2.7: Esempio di comunicazione effettuata su Twitter [9]

La seconda funzionalità di questo modulo permette agli utenti finali di richiedere, tramite Twitter, lo stato di un particolare dispositivo, per un vincolo di sicurezza posto dalle API del social network questa funzione è disponibile esclusivamente agli utenti che risultano essere follower dell'account @RaspeinMonitor; scrivendo infatti a questo account un messaggio privato con la dicitura "status nome Raspberry" si avrà in risposta l'elenco dei sensori attivi con l'ultimo valore rilevato e la data di rilevamento. Per fare ciò il controller verifica in modo periodico i messaggi ricevuti e li analizza iterativamente richiedendo poi in diretta al Core i dati richiesti dall'utente.

L'ultima funzione messa a disposizione è esclusivamente di tipo "social"; essa

permette infatti all'amministratore di scrivere un messaggio privato a @RaspeinMonitor richiedendo di postare un aggiornamento di stato sul profilo pubblico del sistema; per farlo è sufficiente inviare un messaggio privato con la scritta "update messaggio da pubblicare" ed il controller effettuerà la pubblicazione del messaggio verificando preventivamente che il mittente sia effettivamente l'amministratore.

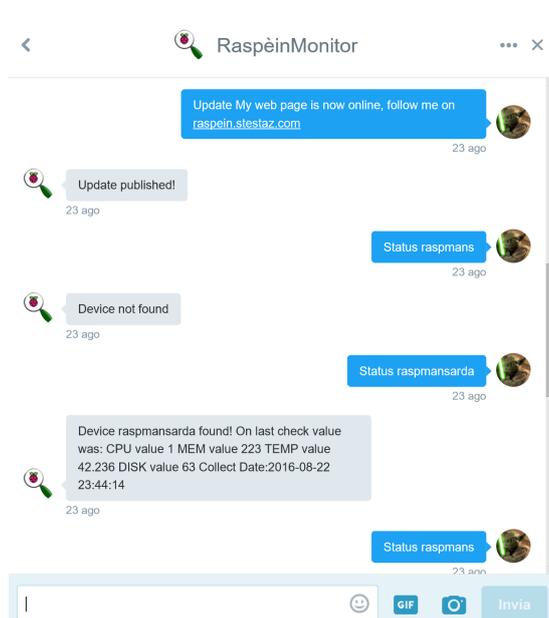


Figura 2.8: Scambio di messaggi tra amministratore e modulo di comunicazione

## 2.5 GUI

La GUI (*Graphical user interface*) è, come indica anche il nome stesso, quella parte del sistema che si interfaccia direttamente con l'utente finale. Come mostrato in Figura 2.9 questo modulo è composto esclusivamente dal server web Apache al quale i client si collegano.

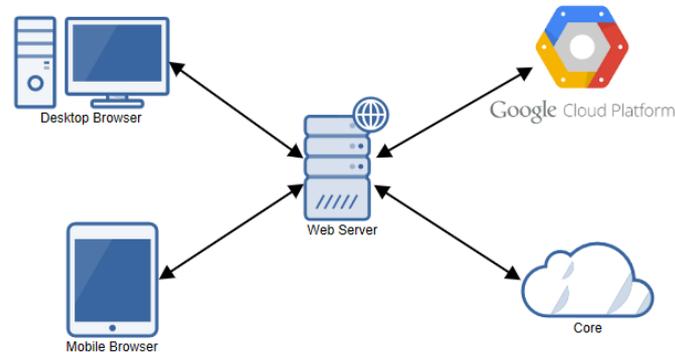


Figura 2.9: Interfacciamento della GUI con client e Core

### 2.5.1 Funzionamento

Essendoci tra le richieste di base la necessità che questo sistema fosse visualizzabile sia su computer che su dispositivi mobili (con particolare attenzione ai tablet) la scelta progettuale è stata quella di utilizzare nella costruzione delle pagine il framework Bootstrap<sup>[10]</sup>. Bootstrap prevede la suddivisione della pagina web secondo una griglia che poi viene adattata e modificata al cambiare delle dimensioni del display che la sta visualizzando rendendo così la pagina “responsive”. Per lo sviluppo di questo progetto web si è scelto di utilizzare le tecnologie che più rappresentano l’attuale era della navigazione, per la creazione delle pagine è stato utilizzato il linguaggio HTML (*HyperText markup language*) nella sua ultima quinta versione, la formattazione delle pagine è invece resa possibile grazie ad un importante utilizzo di fogli di stile in linguaggio CSS3<sup>[11]</sup> (*Cascading style sheet*). Si è scelto infine di popolare il contenuto delle pagine non con PHP, uno dei più classici sistemi server side, ma sfruttando degli script jQuery<sup>[12]</sup> e la tecnica AJAX (*Asynchronous JavaScript an XML*) così da rendere il caricamento dei contenuti asincrono a quello della pagina e della grafica e dando inoltre all’utente la possibilità di interagire con la pagina in modo più dinamico.

### 2.5.2 Homepage

La homepage mostra lo stato attuale di ogni Raspberry posizionandoli all’interno delle torri che costituiscono il cluster Raspèin, all’utente è permesso cliccare su ognuno dei valori rilevati per vederne attraverso un grafico l’andamento nelle ultime 24 ore, cliccando invece sul Lego che identifica il Raspberry è possibile selezionare un arco temporale da analizzare ed un sensore,

verrà così mostrato un grafico con i dati richiesti.

Tutti i valori mostrati e quelli presenti all'interno dei grafici vengono prelevati dal Core attraverso le API messe a disposizione. Per disegnare invece i grafici viene utilizzato un servizio web offerto da Google chiamato Google Charts <sup>[13]</sup>.

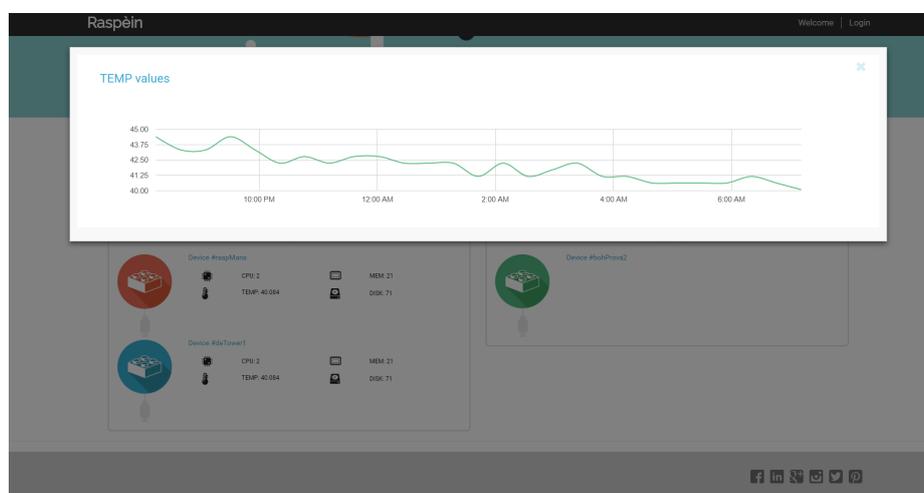


Figura 2.10: Homepage, visualizzazione di un grafico<sup>[14]</sup>

### 2.5.3 Admin Page

Vi è poi la pagina admin raggiungibile solamente previa autenticazione ed attraverso la quale l'amministratore può completamente configurare il sistema e modificare i dati utilizzati dagli altri moduli.

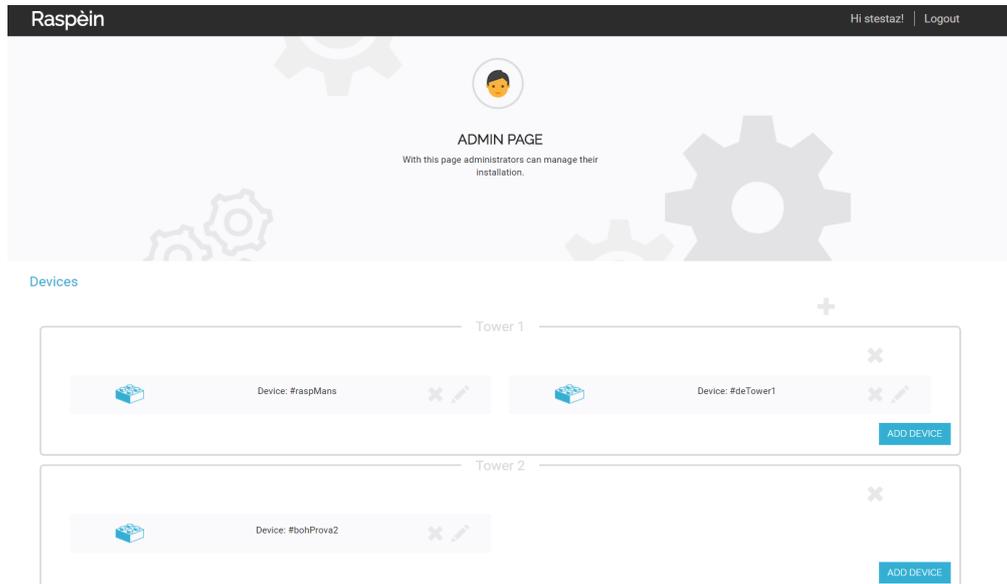


Figura 2.11: Pagina di amministrazione<sup>[15]</sup>

La pagina è formata da due sezioni:

**Devices:** da cui gestire torri e dispositivi

**System:** da cui gestire le impostazioni di sistema

Attraverso le icone presenti all'interno della prima sezione è possibile aggiungere o rimuovere delle torri e, per ognuna, è possibile aggiungere, modificare o rimuovere i dispositivi al suo interno.

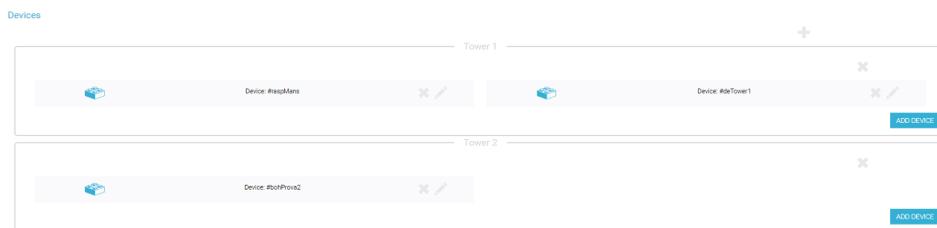


Figura 2.12: Pagina admin, gestione delle torri

Per procedere alla fase di aggiunta di un dispositivo è sufficiente premere il tasto corrispondente ed inserire i dati dello stesso nella modale che verrà visualizzata, i dati richiesti sono il colore che si vuole assegnare al dispositivo (così da modificare il colore dell'icona presente in homepage), un nome univoco ed una posizione all'interno della torre, è poi necessario indicare per i sensori

attivi il range entro il quale il dispositivo è in uno stato di normalità e fuori dal quale si riceverà invece una notifica.

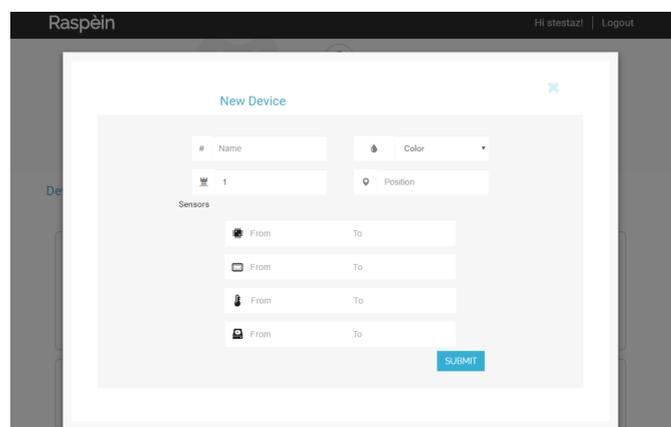


Figura 2.13: Pagina admin, aggiunta di un dispositivo

Attraverso system è invece possibile configurare tutte le variabili di sistema che consentono il corretto funzionamento degli altri moduli, contiene quindi le informazioni relative a Twitter ed al percorso in cui i dispositivi salvano i propri file di diagnostica.



Figura 2.14: Pagina admin, sezione “system”



# Capitolo 3

## Implementazione

Nel capitolo seguente verrà mostrato il metodo implementativo utilizzato per produrre il sistema precedentemente illustrato.

### 3.1 Core

Le parti costituenti il Core sono due, una che effettua l'autenticazione degli utenti e una che esegue l'effettiva computazione delle richieste, entrambe le parti sono web-based quindi le tecnologie ed i linguaggi utilizzati sono propri di questo ambiente.

#### 3.1.1 Interfaccia Web

L'interfaccia web, da utilizzarsi per la creazione e modifica dei token di autenticazione, è stata prodotta utilizzando il linguaggio HTML unito al framework Bootstrap ed a CSS3 per lo stile.

Per l'interfaccia si è scelto di operare principalmente utilizzando degli script jQuery, dovendo però interagire con un database relazionale è stato necessario utilizzare anche PHP, per fare ciò la tecnica utilizzata è stata AJAX con richieste POST, nel Listato 3.1 è mostrato un esempio rappresentante la chiamata ad una funzione PHP e la conseguente interpretazione del risultato di tipo Json, la funzione viene abilitata alla pressione di un elemento sulla pagina che abbia id "addBtn", avendo utilizzato jQuery ogni elemento sulla pagina può essere utilizzato per far scaturire un evento differentemente da quanto invece succederebbe con una pagina sviluppata esclusivamente in linguaggio HTML, inoltre si può notare come i dati poi utilizzati non debbano per forza essere all'interno di uno stesso form ma, come successo per l'evento click, possono essere attinti da qualsiasi elemento presente nel documento.

I selettori messi a disposizione da questo linguaggio sono vari, quelli maggiormente utilizzati nello sviluppo di questo progetto sono i selettori di “classe” o di “id”, un punto apposto all’inizio del nome identifica la volontà di selezionare una classe, un cancelletto invece prevede la selezione di un oggetto per id.

```
$("#addBtn").click(function(){
    var desc=$("#description").val();
    var lvl=$("#level").val();
    desc = desc.replace("'", "");
    $.ajax({
        url: 'php/newKey.php',
        type: 'POST',
        data: {"desc":desc,"lvl":lvl},
        dataType: 'json',
    })
    .done(function (result) {
        $("#newKeyP").text(result);
    })
    .fail(function () {
        $("#newKeyP").text("Error try again");
    });
});
```

Listato 3.1: Chiamata AJAX ad una funzione PHP

Nel Listato 3.2 viene invece mostrata la funzione PHP relativa allo script sopra sopra analizzato la cui funzione è generare una nuova chiave di autenticazione, la particolarità di questa funzione, proposta poi anche nelle successive funzioni dell’authenticator, sta nel controllo preventivo del referer della funzione, viene cioè identificato l’url richiedente e viene verificato che sia corrispondente a quello aspettato per la funzione in oggetto, questo accorgimento è stato preso per ridurre le possibilità di un attacco esterno. Volendo rendere il più possibile “portabile” questo progetto si è inoltre deciso di salvare l’url radice di ogni pagina all’interno di un file di configurazione, tale url verrà poi reso disponibile a tutte le funzioni che lo necessitano, nello stesso file sono inoltre inseriti i dati fondamentali di accesso al database locale.

Al termine della verifica di attendibilità del referer la funzione procede alla creazione di un nuovo token di autenticazione, basandosi su un set di caratteri alfanumerici ogni carattere viene estratto in modo casuale dal set fino a comporre una stringa avente una lunghezza fissa di 40 caratteri.

La chiave generata ed il suo livello vengono poi salvati all’interno del database e ed alla funzione AJAX chiamante viene restituito un risultato contenente la chiave.

```
<?php
include("../inc/settings.php");

$ref = $_SERVER["HTTP_REFERER"];
if( $ref === $referrer."newKey.php"){
    $description = $_POST["desc"];
    $level = $_POST["lvl"];

    $myConn = new mysqli($db_server,$db_user,$db_pass,$db_name);
    do{
        $string = '';
        $characters = 'abcdefghijklmnopqrstuvwxyz0123456789';
        for ($i = 0; $i < 40; $i++) {
            $string .= $characters[rand(0, strlen($characters) - 1)];
        }
        $query = "insert into apis
            values(DEFAULT,'".$string."','".$description."','".$level.",1)";
        $toRet = $string;
    }while(!($myConn->query($query)));
    error_log("ritorno ".$toRet);
    echo json_encode($toRet);
}else{
    die;
}
```

Listato 3.2: Funzione PHP di creazione nuova chiave

L'applicativo web mette inoltre a disposizione dell'operatore altre funzioni che permettono la visualizzazione e modifica di chiavi precedentemente generate, anche in questo caso l'interazione è del tipo jQuery - AJAX - PHP.

### 3.1.2 API Manager

il gestore delle API è la sezione più corposa di questa parte ed è stata interamente sviluppata utilizzando come linguaggio di programmazione PHP orientato ad oggetti, sono stati quindi creati vari oggetti ognuno dei quali svolge una funzione specifica e rende disponibile vari metodi per potervi interagire ed operare quindi più ad alto livello.

L'aspetto fondamentale che un set di API deve possedere è dato dalla semplicità di utilizzo, è necessario quindi fare in modo che richiamando uno specifico url il sistema possa riconoscere automaticamente il corretto servizio da erogare; per poter ottenere questo risultato è stato necessario abilitare sul server web Apache il modulo "mod\_rewrite"<sup>[16]</sup> questo modulo permette infatti, tramite

delle regular expression, di identificare l'url richiesto e reindirizzarlo ad una pagina di destinazione senza però perdere l'url iniziale.

Per permettere il corretto funzionamento di questo modulo è necessario che alcune funzioni all'interno della configurazione del server web siano correttamente configurate, nel Listato B.3 è indicato il codice necessario a permettere innanzitutto l'esecuzione dello script di istruzione del modulo, questa configurazione è legata esclusivamente alla cartella in cui il modulo deve essere utilizzato.

```
<Directory "/var/www/wss/">
    Options Indexes FollowSymLinks
    AllowOverride All
    Allow from All
</Directory>
```

Listato 3.3: Particolare del file di configurazione di Apache

Lo script di istruzione andrà composto all'interno del file “.htaccess” ed il codice necessario al corretto funzionamento, creato prendendo spunto da manualistica e da ricerche online <sup>[17]</sup> è il seguente.

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ /wss/index.php [QSA,L]
```

Listato 3.4: File .htaccess utilizzato

Con il comando “RewriteEngine” viene abilitato il motore del modulo per questa sezione del sito; con il comando “RewriteCond” invece viene definita la condizione che attiva o meno il modulo, nell' esempio mostrato la regola viene attivata se la risorsa richiesta non è un file presente sul server, infine il comando “RewriteRule” definisce cosa è necessario fare se la regola precedente è stata soddisfatta, l'operazione da fare in questo caso è un redirect alla pagina index.php.

Il file index.php deve poter soddisfare quindi tutte le richieste inoltrate, per fare ciò all'interno dello stesso vengono istanziate tutte le classi necessarie, l'implementazione prevede la presenza di tre classi differenti:

**Tools:** per la gestione dell'autenticazione

**Getters:** contenente tutte le funzioni di tipo GET

**Setters:** contenente tutte le funzioni di tipo POST

## Tools

Questo elemento si occupa di realizzare l'autenticazione del client, è composto da due funzioni:

**explane\_url:** effettua il parsing dell'url ricevuto identificando al suo interno l'API richiesta

**apiAuth:** verifica se la key ricevuta esiste a sistema e, in caso affermativo, se il suo livello è sufficiente per l'utilizzo dell'API.

```
class tools {

    function explane_uri($string) {
        $string = str_replace("/wss/", '', $string);
        if ($string != "") {
            $vet = explode('/', $string);
            return $vet[0];
        } else
            return null;
    }

    function apiAuth($key,$reqApi){
        include ("/var/www/wss/inc/settings.php");
        $keyManagerConn = new mysqli($db_api_server,
            $db_api_username,
            $db_api_password,
            $db_api_db);
        $apiManagerConn = new mysqli($db_model_server,
            $db_model_username,
            $db_model_password,
            $db_model_db);

        $response =0;
        $keyLevel = 0;

        $getLevelQuery = "select a.level from apis a
            where a.enable=1 and a.key='".$key."'";
        if( $levelRes = $keyManagerConn->query($getLevelQuery)){

            if($levelRes->num_rows == 1){
                $rowLevel = $levelRes->fetch_assoc();
                $keyLevel = $rowLevel["level"];
            }
        }
    }
}
```

```

$keyManagerConn->close();
if($keyLevel !=0){
    $getApiData = "select a.level,a.enable from availableApis
        a
        where a.name='".$reqApi.'"";
    if($apiData = $apiManagerConn->query($getApiData)){
        if($apiData->num_rows ==1){
            $rowData = $apiData->fetch_assoc();
            if($rowData["enable"] == 1){
                if($rowData["level"]<=$keyLevel){
                    $response = 2;
                }
                else{
                    $response = 1;
                }
            }
        }
    }
}
$apiManagerConn->close();
return $response;
}
}

```

Listato 3.5: Specifica dell'autenticazione

## Getters

Questa classe risponde a tutte le richieste di tipo GET effettuate dai client, al suo interno si compone di un costruttore e di una serie di funzioni specifiche che sono elencate nell'appendice di questa tesi.

Ogni funzione presuppone una interazione con il database, nell'esempio allegato vengono mostrati il costruttore e una funzione che restituisce le informazioni di uno specifico Raspberry organizzate all'interno di un elemento Json.

```

class getters {
    protected $modelDbConn;

    function __construct(){
        include("/var/www/wss/inc/settings.php");
        $this->modelDbConn = new mysqli($db_model_server,
            $db_model_username,
            $db_model_password,

```

```

        $db_model_db);
    }

    function getRaspi($args){
        $return = null;
        $query = "select * from raspberry";
        if(isset($args["id"])){
            if(is_numeric($args["id"])) {
                $query .= " where enable=1 and id=" . $args["id"];
            }
        }
        if($tempRes = $this->modelDbConn->query($query)){
            $return[] = array("result"=>"success");
            while($row = $tempRes->fetch_assoc()){
                $return[] = $row;
            }
        }else{
            $return[] = array("result"=>"error");
        }
        $this->modelDbConn->close();
        return json_encode($return);
    }
}

```

Listato 3.6: Parte della classe Getters con costruttore ed una funzione

## Setters

Questo oggetto, strutturalmente simile a getters si occupa di gestire tutte le richieste ricevute e correttamente autenticate aventi metodo POST, questa tipologia di chiamate è volta ad effettuare soprattutto aggiornamento o inserimento di nuovi dati all'interno del database, un esempio è rappresentato dalla seguente funzione che permette di aggiungere, per un sensore specifico, il valore da esso rilevato.

```

function addData($args){
    $return = -1;
    if(isset($args["id"]) && isset($args["value"])){
        if(is_numeric($args["id"]) && is_numeric($args["value"])){
            $insData = "insert into sensorData
                (sensorId,value,date) ".
                "values(".$args["id"].",
                ".$args["value"].",

```

```
        NOW()");
        if($this->modelDbConn->query($insData)){
            $return = $this->modelDbConn->insert_id;
        }
    }
}
$this->modelDbConn->close();
return json_encode($return);
}
```

Listato 3.7: Estratto dell'oggetto Setters

## 3.2 Polling e Comunicazione

In questa implementazione specifica si è scelto di portare all'interno dello stesso server sia il modulo di Polling che il modulo di comunicazione, lo sviluppo di questi due moduli prevede l'utilizzo di due differenti linguaggi di programmazione:

**Bash** per la creazione dei file su Raspberry

**Java** per le operazioni svolte dal server

### 3.2.1 Raspberry Pi

Su Raspberry Pi il linguaggio scelto è stato bash in quanto risulta essere di uso comune e con costi computazionali limitati, questo è fondamentale per un dispositivo in cui la capacità di calcolo è limitata e che, quando il cluster sarà operativo, sarà necessario sfruttare al 100%.

Per non appesantire ulteriormente il sistema si è scelto di evitare l'installazione di demoni all'interno del dispositivo ma di sfruttare invece "crontab" un demone già attivo ed in funzione normalmente sui sistemi operativi Linux e che ha come unico scopo l'esecuzione programmata di alcune operazioni. Nella fattispecie a crontab viene richiesto di avviare lo script "monBatch.sh" il cui scopo è rilevare lo stato del Raspberry e creare un file con una struttura ben definita così che possa essere analizzato dal server.

Anche sulla creazione del file si è cercato in tutti i modi di ridurre al minimo i byte utilizzati così da avere dimensioni contenute e tempi di trasferimento rapidi, la struttura del file prevede che per ogni riga venga indicato un sensore definito da l'id della propria tipologia e dal valore rilevato, nella prima riga del file dovrà inoltre essere inserito il nome del Raspberry che ha creato il file, una

volta generato il file viene salvato in una directory apposita che verrà indicata anche nei parametri di configurazione dell'installazione così che il server possa identificarla senza possibilità di errore.

```
#!/bin/bash

#con questo script estraggo i dati di funzionamento di raspberry
#e li metto su file
FREECPU=$(vmstat | awk 'FNR==3{print $15}')
FREEMEM=$(free -m | grep Mem | awk '{print $4}')
FREEHD=$(df | grep /dev/root | awk '{print $5}' | sed s"/. $//")
TEMP=$(cat /sys/class/thermal/thermal_zone0/temp)
echo "0-raspMans" > /home/pi/Documents/RaspeinMon/Data/testRes
echo "1-$FREECPU >>/home/pi/Documents/RaspeinMon/Data/testRes
echo "2-$FREEMEM >>/home/pi/Documents/RaspeinMon/Data/testRes
echo "3-$TEMP >>/home/pi/Documents/RaspeinMon/Data/testRes
echo "4-$FREEHD >>/home/pi/Documents/RaspeinMon/Data/testRes
```

Listato 3.8: Script per l'estrazione dello stato di un Raspberry

### 3.2.2 Server

Per produrre la parte server si è scelto di effettuarne lo sviluppo utilizzando Java, uno dei più utilizzati linguaggi di programmazione orientata ad oggetti, la scelta di questo linguaggio è stata dettata in primis dall'elevata portabilità offerta da questo linguaggio di programmazione ed inoltre anche per la grande esperienza pregressa maturata in università.

Per cercare di rendere l'applicativo maggiormente snello e reattivo si è scelto di utilizzare una struttura a Thread, all'avvio viene quindi avviato un "MainThread" il quale effettua l'avvio di ulteriori due thread:

**MessageCrawlerThread** per gestire la comunicazione

**RaspeyDataCollectorThread** per il polling dei dati provenienti dai dispositivi

tutti i thread vengono interrotti in modo controllato se l'utente richiede di terminare l'esecuzione attraverso ad esempio il comando CTRL+C.

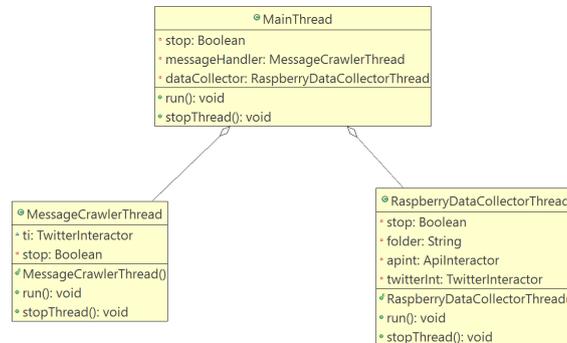


Figura 3.1: Organizzazione dei thread

```

public class MainThread extends Thread{
    private Boolean stop;
    private MessageCrawlerThread messageHandler;
    private RaspberryDataCollectorThread dataCollector;
    @Override
    public void run() {
        this.stop=false;
        ApiInteractor apint = new ApiInteractor();
        ServerConfiguration serverConf = new
            ServerConfigurationImpl(apint,1);
        TwitterInteractor twitterInt = new
            TwitterInteractor(serverConf.getTwitterConsumerKey(),
                serverConf.getTwitterConsumerSecret(),
                serverConf.getTwitterAccessToken(),
                serverConf.getTwitterAccessTokenSecret(),
                serverConf.getTwitterContactName()
            );
        this.messageHandler = new MessageCrawlerThread(twitterInt);
        this.dataCollector = new
            RaspberryDataCollectorThread(serverConf.getFolder(),
                apint,twitterInt);

        messageHandler.start();
        dataCollector.start();
        while (!this.stop){
            try {
                Thread.sleep(10000);
            }
        }
    }
}
  
```

```
        } catch (InterruptedException e) {
            System.out.println("Preso eccezione nel main");
            messageHandler.stopThread();
            e.printStackTrace();
        }
    }
}
public void stopThread(){
    this.stop=true;
    messageHandler.stopThread();
}
}
```

Listato 3.9: Classe MainThread

## Comunicazione

La parte di comunicazione è svolta dall'oggetto `MessageCrawlerThread` questo thread si esegue in modo periodico la funzione `parseMessages` della classe "TwitterInteractor". Questo metodo nel momento in cui viene invocato verifica attraverso le API di Twitter e la libreria `Twitter4J` <sup>[18]</sup> se sono presenti dei messaggi privati per l'account impostato (i dati sono prelevati dal Core e vengono passati nel costruttore di `TwitterInteractor`) e, nel caso ci fossero, vengono analizzati per individuare se è presente uno dei comandi riconosciuti. Nel caso in cui il comando richiesto preveda di prelevare dei dati dal Core viene istanziata la classe `ApiInteractor` che provvede ad eseguire le operazioni di GET e POST sul Core.

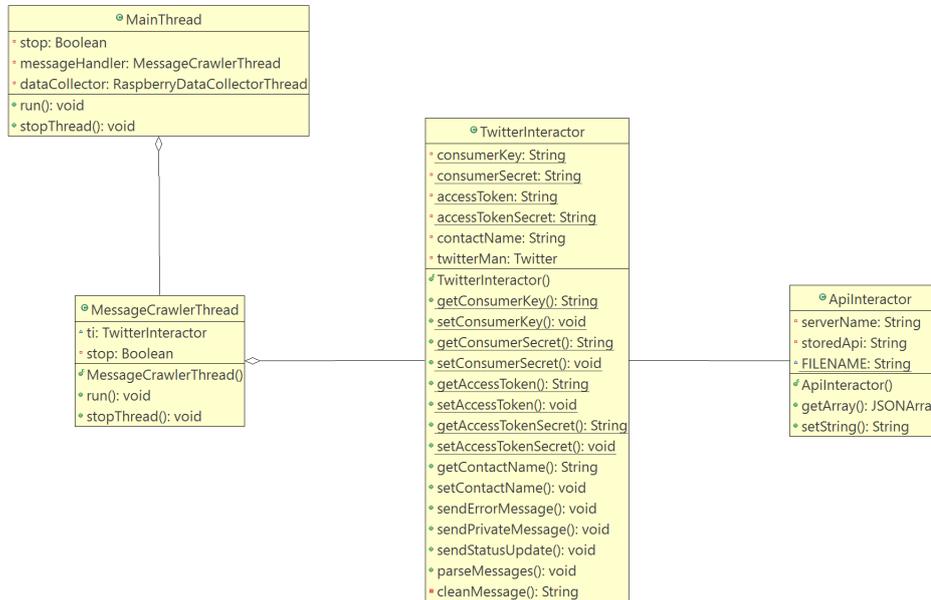


Figura 3.2: Diagramma delle classi della parte di comunicazione

## Polling

La parte che effettua il polling dei dati dai dispositivi è svolta dall'oggetto "RaspberryDataCollectorThread".

Vista la crescente complessità di questa parte si è scelto di creare delle interfacce e delle implementazioni delle stesse per ogni tipologia di operazione da eseguire cosicché la classe principale possa lavorare più ad alto livello e futuri interventi di manutenzione risultino più agevoli. Il diagramma delle classi sottostante illustra le relazioni presenti tra i vari oggetti e la tipica distinzione interfaccia - classe presente in organizzazioni di questa tipologia.

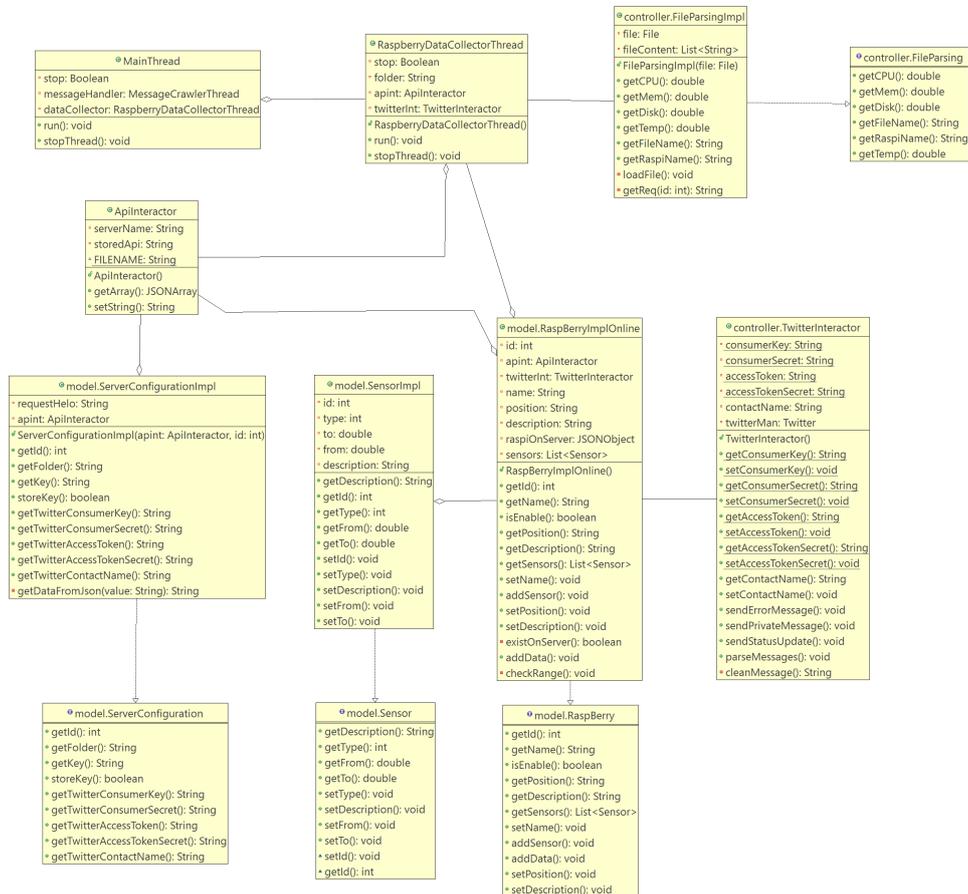


Figura 3.3: Diagramma delle classi della parte di polling

Il costruttore della classe `RaspberryDataCollectorThread` prevede che vengano passate alla stessa gli elementi necessari all'espletamento delle proprie funzioni, questi elementi sono:

**Folder** necessario per identificare la cartella in cui vengono salvati i file dai dispositivi

**ApiInteractor** per l'interazione con il Core

**TwitterInteractor** per l'invio di messaggi di allerta

Estendendo la superclasse `Thread` l'oggetto `RaspberryDataCollectorThread` viene messo in esecuzione richiamando il metodo `.start` che produce l'esecuzione di "run"; questo metodo in modo periodico verifica la presenza di file

all'interno della directory inizialmente definita e, attraverso l'oggetto "FileParsingImpl", effettua il parsing di ogni singolo file presente creando per ognuno un nuovo oggetto di tipo "RaspberryImplOnline", copia speculare del dispositivo ricevuto attraverso le API. Per ogni dispositivo viene poi verificata la presenza dei corretti sensori e, per ognuno, aggiunto il valore rilevato. Durante l'aggiunta dei dati è presente un particolare metodo privato "checkRange" che verifica se il valore rilevato risulta compreso nel range di valori definito dall'amministratore attraverso la GUI e, nel caso vi sia un errore, invia un messaggio privato al gestore indicandogli l'errore rilevato, in aggiunta viene effettuato un aggiornamento di stato sul profilo del sistema.

```
private void checkRange(Sensor sens, double value){
    int error=0;
    if(sens.getFrom() != sens.getTo()){
        if(value > sens.getTo() || value < sens.getFrom()){
            error ++;
            try {
                String sensType="";
                switch (sens.getType()){
                    case 1:
                        sensType = "CPU";
                        break;
                    case 2:
                        sensType = "MEM";
                        break;
                    case 3:
                        sensType = "TEMP";
                        break;
                    case 4:
                        sensType = "DISK";
                        break;
                }
                twitterInt.sendMessage("Il dispositivo Raspberry
                    avente nome "+this.name+
                    " e posizione "+this.position
                    +" ha rilevato un valore al di fuori del range
                    impostato sulla risorsa "
                    +sensType + " il valore doveva essere compreso
                    tra "+sens.getFrom()+ " e " + sens.getTo()
                    +" ed invece e "+ value);
            } catch (TwitterException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
    }

    }else{
        //Valore nei limiti
    }
}
if(error !=0){
    try {
        twitterInt.sendStatusUpdate("Errors found on last check
            please verify");
    } catch (TwitterException e) {
        e.printStackTrace();
    }
}
}
```

Listato 3.10: Metodo checkRange estratto da RaspberryImplOnline

### 3.3 GUI

L'ultima sezione sviluppata è la GUI, per l'utilizzatore dell'applicativo risulta essere la parte che maggiormente ne influenza il gradimento e quindi, per riuscire al meglio, deve essere pensata e sviluppata mettendosi dalla parte dell'utilizzatore stesso. Partendo da questo concetto base si è scelto di sviluppare l'interfaccia grafica secondo i nuovi dettami dello sviluppo web che prediligono pagine pulite e semplici e automaticamente adattabili a dispositivi aventi display di tutte le dimensioni.

Inizialmente è stata valutata la possibilità di utilizzare dei CMS (*content management system*) per la creazione delle pagine e del loro contenuto ma poi, analizzando quelli presenti sul commercio, è stato chiaro che le necessità del progetto erano al di fuori di quanto attualmente disponibile e che l'adattamento degli attuali sistemi avrebbe richiesto lo sviluppo di particolari moduli personalizzati; si è scelto quindi di partire da zero e produrre interamente scripts, layout grafico e pagine html.

Le pagine che compongono il sito sono due, la homepage a cui accedono tutti gli utenti finali e la parte admin a cui possono accedere esclusivamente gli amministratori.

In entrambe le pagine il linguaggio di programmazione utilizzato maggiormente è stato l'HTML prediligendolo al PHP.

### 3.3.1 Scripts

Come per la parte del Core vista inizialmente si è optato per l'utilizzo di script jQuery e AJAX per richiedere i dati alle API in modo asincrono, ogni pagina ha un suo set di scripts apposito così che i dati caricati dal dispositivo siano esclusivamente quelli necessari alla visualizzazione di quella pagina. Durante lo sviluppo di questa parte ci si è immediatamente accorti che l'utilizzo di script per intercettare eventi su oggetti generati in modo dinamico non si poteva basare sui classici paradigmi utilizzati in pagine statiche, per risolvere questo problema quindi si è dovuto operare utilizzando delle particolari funzioni di jQuery che basano gli eventi non sull'oggetto cliccato ma sul body della pagina, nel Listato 3.11 l'evento "click" sull'oggetto con classe "deviceIcon" viene rilevato dal body della pagina invece che dalla classe stessa.

```
$(document).ready(function(e) {  
  
    $("body").on("click", ".deviceIcon",function(){  
        populateModal($(this).siblings(".devId").text());  
    });  
});
```

Listato 3.11: Script della homepage

Come detto la pagina viene popolata dinamicamente e quindi al caricamento della stessa viene inviata una richiesta alle API con i dati di tutti i dispositivi attivi sull'installazione e in questo modo la pagina viene disegnata, nei listati successivi vengono mostrate tre delle funzioni che eseguono queste operazioni. "GetSystemsAndRaspi" interroga una funzione PHP locale chiamata "getConfiguration.php" che rileva, da un file di configurazione presente nel webserver, il link del Core (così che per esigenze di installazione il Core possa essere spostato), il token di autenticazione e l'id dell'installazione; con questi dati viene poi richiamata una API sul Core che restituisce sottoforma di Json tutti i dispositivi presenti nell'installazione indicata e, per ognuno di questi, viene definita la struttura in torri attraverso la chiamata alle funzioni "addTower" e "addDeviceToTower".

```
function getSystemsAndRaspi(){  
$.ajax({  
    url: '/scripts/getConfiguration.php',  
    type: 'POST',  
    dataType: 'json',  
    success: function(res){  
        var key = res.key;
```

```
var id = res.id;
var server = res.server;
$.ajax({
  url: server+'/getRaspiFromInstId/',
  type: 'GET',
  data: {key:key,instId:id},
  dataType: 'json',
  success: function(resApi){
    var tower = 0;
    var i = 1;
    if(resApi[0].result === "success"){
      for(i=1; i<resApi.length;i++){
        if(tower != resApi[i].tower){
          tower = resApi[i].tower;
          addTower(tower);
        }
        addDeviceToTower(tower,
          resApi[i].name,
          resApi[i].id,
          resApi[i].color);
      }
    }
  },
  error: function(){
    alert("Dati Errati API");
  }
});
},
error: function(){
  alert("Dati Errati");
}
});
}
```

Listato 3.12: Funzione GetSystemsAndRaspi

Le funzioni `addTower` e `addDeviceToTower` hanno lo scopo di generare la pagina modificando il codice HTML della stessa, per farlo utilizzano il metodo “append” che permette di aggiungere del codice dopo un dato elemento, nella fattispecie sono stati creati due elementi aventi come classe “towerContainer” per le torri e “deviceContainer” per i dispositivi.

Una ulteriore problematica incontrata nell’interazione con oggetti creati dinamicamente è stata l’utilizzo del metodo “val” fornito da jQuery, questo metodo non è infatti utilizzabile in quanto la pagina viene creata dopo che il dom (ele-

mento che rappresenta il contenitore di tutta la pagina) è stato caricato, la soluzione intrapresa prevede l'aggiunta di elementi nascosti con i dati necessari inseriti all'interno del testo e recuperabili quindi attraverso il metodo "text".

```
function addTower(towerId){
    $(".towerContainer").append(
        "<div id='tower'+towerId+' ' class='col-xs-12 col-sm-6'>"+
        "<fieldset class='colContainer'>"+
        "    <legend class='towerLegend '>Tower
        "+towerId+"</legend>"+
        "    <div class='row'>"+
        "        <div class='deviceContainer'>"+
        "        </div>"+
        "</div></fieldset></div>");
}
function addDeviceToTower(towerId,deviceName,deviceId,deviceColor){
    var tower = "#tower"+towerId;
    var device = "device"+deviceId;
    $(tower).children().find(".deviceContainer").append(
        "<div class='row'>"+
        "    <div class='col-xs-12 device'>"+
        "        <form class='form-inline' role='form'>"+
        "            <div class='col-xs-3 myDevIconCont'>"+
        "                <span>"+
        "                    <img src='img/'+deviceColor+
        "                    'Icon.png' aria-hidden='true' "+
        "                    class='deviceIcon img-responsive' "+
        "                    data-toggle='modal' "+
        "                    data-target='#myStatsModal' "+
        "                    alt='device icon'>"+
        "                <p class='hidden devId'>"+deviceId+"</p>"+
        "            </span></div>"+
        "            <div class='col-xs-9'>"+
        "                <div class='row'>"+
        "                    <div class='col-xs-12'>"+
        "                        <p id='"+device+" ' "+
        "                        class='deviceInfo'>"+
        "                            Device #"+deviceName+"</p>"+
        "                    </div>"+
        "                </div>"+
        "            </div>"+
        "        </form>"+
        "    </div>"+
        "</div>");
```

```

    setRaspiLastValues(deviceId,deviceName);
}

```

Listato 3.13: Funzioni AddTower e AddDeviceToTower

All'interno della homepage sono stati inseriti dei grafici così da permettere agli utenti di visualizzare in tempo reale lo stato di un particolare sensore o di un particolare dispositivo; per lo sviluppo di questa sezione si è scelto di utilizzare un servizio già presente sulla rete e gratuitamente offerto da Google chiamato Google Charts<sup>[13]</sup>.

Attraverso i Google Charts è possibile disegnare dei grafici in funzione a una base dati fornita con una particolare formattazione; nella pagina i grafici sono inseriti all'interno di una modale e vengono quindi caricati in modo asincrono alla pressione del tasto, questo permette di ridurre il tempo di caricamento della pagina anche se ovviamente rallenta leggermente l'apertura del grafico stesso che andrà quindi a comporsi nell'istante in cui l'utente ne richiede la visione; la funzione che recupera i dati odierni dal Core e li passa a Google Charts è chiamata "populateChart" ed è mostrata nel listato seguente, esiste inoltre una ulteriore funzione chiamata "populateChartFromTo" che si occupa di disegnare il grafico relativo ai valori in un range di tempo selezionato dall'utente e che prevede un funzionamento simile alla precedente.

```

$.ajax({
    url: server+'/getTodayData/',
    type: 'GET',
    data: {key:key,id:sensorId},
    dataType: 'json',
    success: function(resData){
        if(resData[0].result === "success"){
            var chartData = new google.visualization.DataTable();
            var f=1;
            chartData.addColumn("datetime","date");
            chartData.addColumn("number",sensType+" Value");
            for(f=1; f<resData.length;f++){
                chartData.addRow(resData.length-1);
                chartData.setCell(f-1,1,resData[f].value);
                chartData.setCell(f-1,0,new
                    Date(resData[f].date));
            }
            google.charts.setOnLoadCallback(drawChart(chartData,
                "curve_chart"));
            $("#modalH3").text(sensType + " values");
        }
    },

```

```
error: function(){
    alert("Dati Errati API");
}
});
```

Listato 3.14: Funzione populateChart

Gli script presenti all'interno della zona admin prevedono, a differenza di quanto accade nella homepage, che l'amministratore possa interagire in modo attivo con il sistema, da questa sezione è infatti possibile modificare la configurazione, aggiungere o rimuovere dispositivi. Il numero di script utilizzati sarà quindi maggiore rispetto alla homepage e prevede un approccio simile a quanto analizzato basato su jQuery ed AJAX, un esempio è mostrato nel Listato 3.15 che mostra la procedura utilizzata per la rimozione da interfaccia e da database di un particolare dispositivo

```
function removeDevice(deviceId){
    var id = deviceId.substring(7);
    $.ajax({
        url: '/scripts/getConfiguration.php',
        type: 'POST',
        dataType: 'json',
        success: function(res){
            var key = res.key;
            var server = res.server;
            var instId = res.id;
            $.ajax({
                url: server+'/editRaspi/',
                type: 'POST',
                data: {key:key,
                    id:id,
                    enable:0
                },
                dataType: 'json',
                success: function(resApi){
                    if(resApi != -1){
                        $(deviceId).remove();
                    }else{
                        alert("unable to remove");
                    }
                },
                error: function(){
                    alert("Dati Errati API");
                }
            });
        }
    });
}
```

```
        });  
    },  
    error: function(){  
        alert("Dati Errati");  
    }  
});  
}
```

Listato 3.15: Rimozione di un dispositivo

### 3.3.2 Stile

Per lo sviluppo delle pagine si è scelto di utilizzare un approccio mobile-oriented, uno dei framework attualmente maggiormente utilizzati <sup>[19]</sup> per questa tipologia di progetti è Bootstrap, questo framework basato su delle librerie JavaScript e fogli di stile CSS prevede una suddivisione degli elementi componenti la pagina in classi particolari.

Il CSS viene suddiviso utilizzando delle “media query” che permettono di definire l’aspetto di un determinato elemento in funzione alla dimensione dello schermo, nell’esempio mostrato si può notare come due elementi siano visualizzati in modo diverso nel caso in cui il dispositivo abbia un display con larghezza fino a 767pixel o su un display con larghezza compresa tra 768 e 991pixel.

```
@media (max-width: 767px) {  
    .containerLogin{  
        margin:2%;  
    }  
    .headerImg{  
        display:none;  
    }  
}  
  
@media (min-width: 768px) and (max-width: 991px) {  
    .containerLogin{  
        margin:4%;  
    }  
    .headerImg{  
        display:block;  
    }  
}  
}
```

Listato 3.16: Esempio di media query

Oltre a definire questa suddivisione tramite media query la particolarità di Bootstrap risiede nel suddividere la pagina in una griglia<sup>[20]</sup> che, a seconda delle dimensioni dello schermo, potrà essere:

**xs:** extra small

**sm:** small

**md:** medium

**lg:** large

questo tipo di concetto è stato inizialmente complesso da interpretare e da utilizzare in quanto gli elementi non vengono più gestiti all'interno della pagina in funzione al loro tipo ma principalmente per la classe a cui appartengono.

Con Bootstrap la pagina viene quindi suddivisa in dodici colonne e, ogni elemento che la compone può essere dimensionato utilizzando una o più colonne, per migliorare l'adattabilità però è inoltre possibile definire che un oggetto sia largo un quantitativo di colonne differente tra le varie tipologie di visualizzazione evitando così posizionamenti erronei o sprechi di spazio, per fare ciò è necessario quindi definire tutte queste particolarità in fase di progettazione ed associare poi ad ogni elemento le classi più adatte. Tutti gli elementi che si trovano all'interno di un oggetto avranno a loro volta una dimensione che potrà andare da uno a dodici colonne indifferentemente dalle dimensioni del padre, la dimensione però in pixel di ogni colonna risulterà inferiore alla corrispondente nel padre.

Nell'esempio viene mostrato come una torre nella homepage abbia una dimensione di dodici colonne in un display di piccole dimensioni "col-xs-12" ma assuma poi una dimensione di sei colonne per display di dimensioni che vanno da small a salire "col-sm-6", se non differentemente indicato le dimensioni maggiori ereditano la classe delle inferiori così che non serva obbligatoriamente riscrivere la classe per ogni dimensione.

Nello stesso esempio si può notare la suddivisione per righe "row" utilizzata solitamente con questo framework, questa suddivisione permette di meglio gestire il movimento degli oggetti al cambio di risoluzione.

Un'ultima particolarità può essere rilevata nell'oggetto immagine, in questo caso è presente un "data-toggle" che assegna all'immagine l'evento di comparsa di una form modale avente id "myStatsModal".

```
<div id="tower2" class="col-xs-12 col-sm-6">
  <fieldset class="colContainer">
    <legend class="towerLegend ">Tower 2</legend>
    <div class="row">
      <div class="deviceContainer">
```

```
<div class="row">
  <div class="col-xs-12 device">
    <form class="form-inline" role="form">
      <div class="col-xs-3 myDevIconCont">
        <span>
          
          <p class="hidden devId">10</p>
        </span>
      </div>
      <div class="col-xs-9">
        <div class="row">
          <div class="col-xs-12">
            <p id="device10"
                class="deviceInfo">Device
                #bohProva2</p>
          </div>
        </div>
      </div>
    </form>
  </div>
</div>
</fieldset>
</div>
```

Listato 3.17: Utilizzo di bootstrap nella homepage

L'utilizzo di Bootstrap ha permesso di semplificare la creazione di un sito responsive ma il design pensato per le varie pagine si discosta molto dagli standard del framework, per ottenere quindi il risultato voluto è stato necessario creare dei fogli di stile personalizzati che vengono caricati in cascata ai CSS di Bootstrap così da essere applicati per ultimi agli elementi come da linguaggio CSS.

Per non interferire con la struttura di Bootstrap si è inoltre deciso di utilizzare, ove possibile, delle classi personalizzate per ogni tipologia di elemento evitando quindi di usare le classi già proprie del framework. La suddivisione dello stile è stata fatta in più CSS in modo che ogni pagina carichi esclusivamente quanto necessario per la visualizzazione fatta eccezione per i contenuti comuni

quali testata e piede che si ripetono in entrambe le pagine. Anche per questi elementi si sono utilizzate le media query così da rendere le pagine sempre correttamente dimensionate; l'impatto grafico e l'usabilità da computer sono stati inoltre inseriti dei particolari effetti quando il mouse si trova su un elemento, un esempio è mostrato di seguito.

```
.deviceIcon:hover{
  background-color:rgba(52, 175, 211,0.2);
}
.deviceInfo{
  font-family: Roboto;
  color: #34afd3;
  font-weight: 500;
}
```

Listato 3.18: Esempio di stile sull'elemento con classe deviceIcon

## 3.4 Accessibilità

Grazie alle nuove tecnologie sempre più persone hanno accesso alle pagine web esistenti online, per questo è importante che queste siano fruibili dalla maggior parte di utenti possibile compreso chi possiede delle disabilità, attraverso dei tool online <sup>[21]</sup> tutte le pagine create sono state rese compatibili con lo standard WCAG 2.0 (*Web content accessibility guidelines*) livello AAA che, su direttiva del W3C <sup>[22]</sup> (*World wide web consortium*) definisce quelle che solo le linee guida per permettere l'accesso ai contenuti web anche per utenti che presentano disabilità.

# Capitolo 4

## Valutazioni Finali

### 4.1 Analisi dei Risultati

L'intento di questa tesi era riuscire a dotare un cluster, formato da vari Raspberry Pi, di un sistema di monitoraggio utilizzando il quantitativo minimo di risorse e permettendo che il sistema potesse interagire con il mondo esterno. Analizzando la soluzione prodotta ritengo che il risultato voluto sia stato ottenuto, seppure ci sia spazio per ulteriori miglioramenti il software sviluppato permette di effettuare il monitoraggio non solo di un cluster Raspberry ma in generale di un qualsiasi dispositivo o insieme di dispositivi basato su sistema operativo Linux. L'impatto prestazionale è stato, come da richieste, ridotto al minimo e ciò permette quindi di essere applicato anche a dispositivi con risorse limitate (ad esempio Beagle Board).

Utilizzando esclusivamente tecnologie open source la soluzione non prevede costi derivanti da licenze o da contratti di manutenzione quindi può ritenersi a costo nullo se non quello dato dall'intervento umano per mantenere o ampliare il sistema.

Per finire ritengo sia soddisfatto anche l'intento di voler creare un progetto che definisse le basi da cui partire per lo studio e l'implementazione di nuove funzionalità, la struttura modulare basata su un nucleo centrale permetterà infatti di sviluppare sempre nuove funzionalità o migliorare quelle esistenti.

### 4.2 Considerazioni

Personalmente ho trovato il percorso di realizzazione del progetto molto interessante e formativo, dover sviluppare delle soluzioni su diversi ambiti della programmazione da quella web a quella più a basso livello e su tipologie di host più disparati che vanno da piccoli dispositivi quali i Raspberry Pi a

server di maggiori dimensioni ha richiesto che io acquisissi nuove competenze e migliorassi quelle esistenti.

Questo processo di crescita, soprattutto per lo sviluppo web, si nota chiaramente valutando le differenze presenti tra i due siti web creati per questo progetto, il primo per la gestione delle API key è molto più standard ed utilizza concetti più semplici, il secondo che rende il progetto visibile al pubblico ha una grafica completamente personalizzata ed utilizza delle interazioni molto più complesse.

### 4.3 Sviluppi futuri

Per questa tesi si era valutata la possibilità di sviluppare anche una applicazione per dispositivi mobili ma si è poi preferito concentrare maggiormente l'attenzione sulla creazione dell'infrastruttura permettendo comunque la fruizione del progetto attraverso browser web anche su smartphone e tablet, lo sviluppo di un'applicazione nativa potrebbe però rappresentare una delle possibili evoluzioni del sistema con la possibilità anche di espandere questo pensiero a dispositivi indossabili.

Sarebbe inoltre interessante studiare ed implementare diverse tipologie di comunicazione tra il sistema e gli utenti utilizzando ad esempio email, SMS o altri metodi.

In ultimo si potrebbe valutare l'implementazione di questo sistema anche su infrastrutture di maggiori dimensioni, infatti sviluppando un servizio per il polling di dispositivi Windows (utilizzando ad esempio Java o C#), potrebbe essere possibile monitorare tutti i computer presenti sulla rete, inoltre creando un modulo che sfrutta il protocollo SNMP tale controllo potrebbe essere esteso a dispositivi di rete quali stampanti o switch.

# Ringraziamenti

Vorrei ringraziare innanzitutto il mio relatore Gabriele d'Angelo per l'opportunità offerta e la profonda dedizione che mette nel suo mestiere, la passione che lo muove rende ogni studente orgoglioso di aver frequentato il suo corso.

Ringrazio Mariachiara per avermi sopportato e supportato sin dall'inizio in questa avventura iniziata quasi per scherzo ma felicemente portata a termine.

Ringrazio l'azienda per cui lavoro che ha saputo sopportare i miei giorni di ferie passati a lezione o sui libri.

Ringrazio infine gli "...Aspiranti ingegneri...", abbiamo passato tre anni insieme in cui ci siamo conosciuti, siamo cresciuti ed insieme abbiamo affrontato ogni esame nella buona e nella cattiva sorte.

L'ultimo ringraziamento e forse il più importante va al "Dream Team" ed alla "Somma e potente Chiara", le sere passate assieme intorno ad un tavolo e la loro amicizia sono state il carburante che è riuscito a portarmi qui, nonostante chi diceva che non ce l'avrei potuta fare, nonostante la stanchezza e gli impegni, gran parte del mio percorso è merito loro e non potrò mai ringraziarli a sufficienza, saranno per sempre i miei ragazzi che ho visto crescere ed io sarò per sempre "quello anziano".

Grazie di cuore a tutti.



# Appendice A

## Elenco API

In questa sezione verranno elencate le API fornite dal Core e per ognuna verranno indicati i dati necessari ed il risultato aspettato. Per ognuna è indispensabile passare come argomento "key" il token di autenticazione quindi tale argomento verrà tralasciato.

### A.1 Getters

Tutte le API di tipo "getter" e che prevedono esclusivamente la lettura di dati dal Core devono essere richieste tramite metodo GET.

#### A.1.1 getRaspi

Questa funzione fornisce uno specifico Raspberry o un elenco di Raspberry

##### Parametri

**id** rappresenta l'id dell'installazione richiesta (*opzionale*)

##### Risposta

in risposta viene restituito un Json con il campo "result" come primo elemento, il valore di questo elemento potrà essere "error" in caso di errore o "success" se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo del dispositivo

**name** nome del dispositivo

**tower** torre all'interno della quale il dispositivo è inserito

**position** posizione occupata all'interno della torre

**description** descrizione del dispositivo (attualmente non utilizzata)

**instId** id dell'installazione a cui fa riferimento

**color** colore che dovrà avere nella visualizzazione

### A.1.2 **getRaspiFromInstId**

Questa funzione fornisce l'elenco dei Raspberry associati ad uno specifico id di installazione.

#### **Parametri**

**instId** rappresenta l'id dell'installazione richiesta

#### **Risposta**

in risposta viene restituito un Json con il campo "result" come primo elemento, il valore di questo elemento potrà essere "error" in caso di errore o "success" se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo del dispositivo

**name** nome del dispositivo

**tower** torre all'interno della quale il dispositivo è inserito

**position** posizione occupata all'interno della torre

**description** descrizione del dispositivo (attualmente non utilizzata)

**instId** id dell'installazione a cui fa riferimento

**color** colore che dovrà avere nella visualizzazione

### A.1.3 **getRaspiFromName**

Questa funzione fornisce uno specifico Raspberry effettuandone la ricerca per nome.

#### **Parametri**

**name** rappresenta il nome del Raspberry da ricercare

## Risposta

in risposta viene restituito un Json con il campo “result” come primo elemento, il valore di questo elemento potrà essere “error” in caso di errore o “success” se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo del dispositivo

**name** nome del dispositivo

**tower** torre all’interno della quale il dispositivo è inserito

**position** posizione occupata all’interno della torre

**description** descrizione del dispositivo (attualmente non utilizzata)

**instId** id dell’installazione a cui fa riferimento

**color** colore che dovrà avere nella visualizzazione

### A.1.4 getSensors

Questa funzione ritorna l’elenco dei sensori associati ad uno specifico Raspberry.

## Parametri

**id** rappresenta l’id del Raspberry del quale si richiedono i sensori

## Risposta

in risposta viene restituito un Json con il campo “result” come primo elemento, il valore di questo elemento potrà essere “error” in caso di errore o “success” se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo del sensore

**type** codice rappresentante la tipologia di sensore

**typeDesc** descrizione della tipologia di sensore

**description** descrizione del sensore (attualmente non utilizzata)

**fromValue** valore minimo da cui il sensore si trova in stato di normalità

**toValue** valore massimo entro il quale il sensore si trova in stato di normalità

### A.1.5 `getData`

Questa funzione restituisce l'elenco dei valori rilevati da un determinato sensore.

#### Parametri

**id** rappresenta l'id del sensore di cui si richiedono i rilevamenti

#### Risposta

in risposta viene restituito un Json con il campo "result" come primo elemento, il valore di questo elemento potrà essere "error" in caso di errore o "success" se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo della rilevazione

**sensorId** id del sensore

**value** valore rilevato

**date** timestamp della rilevazione

### A.1.6 `getTodayData`

Questa funzione fornisce le rilevazioni effettuate da un determinato sensore nelle ultime 24 ore, viene restituita solo una rilevazione ogni cinque per evitare un eccessivo quantitativo di dati, per avere un risultato completo vedere la funzione `getTodayDataFull`.

#### Parametri

**id** rappresenta l'id del sensore di cui si richiedono i rilevamenti

#### Risposta

in risposta viene restituito un Json con il campo "result" come primo elemento, il valore di questo elemento potrà essere "error" in caso di errore o "success" se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo della rilevazione

**sensorId** id del sensore

**value** valore rilevato

**date** timestamp della rilevazione

### A.1.7 **getTodayDatFull**

Questa funzione fornisce le rilevazioni effettuate da un determinato sensore nelle ultime 24 ore.

#### **Parametri**

**id** rappresenta l'id del sensore di cui si richiedono i rilevamenti

#### **Risposta**

in risposta viene restituito un Json con il campo "result" come primo elemento, il valore di questo elemento potrà essere "error" in caso di errore o "success" se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo della rilevazione

**sensorId** id del sensore

**value** valore rilevato

**date** timestamp della rilevazione

### A.1.8 **getFromToData**

Questa funzione fornisce le rilevazioni effettuate da un determinato sensore in una fascia temporale specifica, il risultato ottenuto viene limitato ad una rilevazione su dieci per evitare quantitativi troppo elevati di dati, è comunque presente la funzione getFromToDataFull per avere un risultato completo.

#### **Parametri**

**id** rappresenta l'id del sensore di cui si richiedono i rilevamenti

**from** indica la data da cui si vogliono i dati

**to** indica la data fino alla quale si richiedono i dati

## Risposta

in risposta viene restituito un Json con il campo “result” come primo elemento, il valore di questo elemento potrà essere “error” in caso di errore o “success” se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo della rilevazione

**sensorId** id del sensore

**value** valore rilevato

**date** timestamp della rilevazione

### A.1.9 getFromToDataFull

Questa funzione fornisce le rilevazioni effettuate da un determinato sensore in una fascia temporale specifica.

#### Parametri

**id** rappresenta l’id del sensore di cui si richiedono i rilevamenti

**from** indica la data da cui si vogliono i dati

**to** indica la data fino alla quale si richiedono i dati

## Risposta

in risposta viene restituito un Json con il campo “result” come primo elemento, il valore di questo elemento potrà essere “error” in caso di errore o “success” se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo della rilevazione

**sensorId** id del sensore

**value** valore rilevato

**date** timestamp della rilevazione

### A.1.10 getLastData

Questa funzione restituisce l’ultimo valore rilevato da un determinato sensore

## Parametri

**id** rappresenta l'id del sensore di cui si richiedono il valore

## Risposta

in risposta viene restituito un Json con il campo "result" come primo elemento, il valore di questo elemento potrà essere "error" in caso di errore o "success" se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo della rilevazione

**sensorId** id del sensore

**value** valore rilevato

**date** timestamp della rilevazione

### A.1.11 getConfiguration

Questa funzione restituisce i Parametri di configurazione di una determinata installazione

## Parametri

**id** identifica l'id dell'installazione

## Risposta

in risposta viene restituito un Json con il campo "result" come primo elemento, il valore di questo elemento potrà essere "error" in caso di errore o "success" se la ricerca ha dato esito positivo, in questo caso i seguenti dati seguiranno:

**id** identificativo dell'installazione

**apiKey** token assegnato a questa installazione

**folder** cartella nella quale il Controller dei dispositivi andrà a cercare i file da analizzare

**twitterConsumerKey** API key utilizzata per l'invio di messaggi su twitter

**twitterConsumerSecret** codice segreto utilizzato per le API

**twitterAccessToken** token utilizzato per l'invio di messaggi su twitter

**twitterAccessTokenSecret** codice segreto utilizzato per il token

**twitterContactName** nickname dell'amministratore del sistema

## A.2 Setters

Tutte le API di tipo "setter" che prevedono cioè la scrittura di dati nel Core devono essere istanziate con metodo POST

### A.2.1 addRaspi

Aggiunge un Raspberry al sistema.

#### Parametri

**name** nome del dispositivo

**tower** torre all'interno della quale il dispositivo è inserito

**position** posizione occupata all'interno della torre

**description** descrizione del dispositivo

**instId** id dell'installazione a cui fa riferimento

**color** colore che dovrà avere nella visualizzazione

#### Risposta

verrà restituito l'id del Raspberry aggiunto o -1 in caso di errore

### A.2.2 addSensor

Aggiunge un sensore ad un Raspberry precedentemente creato

#### Parametri

**raspId** id del Raspberry a cui aggiungere il sensore

**sensType** id identificante la tipologia di sensore

**description** descrizione del dispositivo

**from** valore di soglia minimo per il sensore

**to** valore di soglia massimo per il sensore

### Risposta

verrà restituito l'id del sensore creato o -1 in caso di errore

### A.2.3 addData

Permette di aggiungere un rilevamento ad un sensore.

### Parametri

**id** identificativo del sensore

**value** valore rilevato

### Risposta

verrà restituito l'id dell'inserimento o -1 in caso di errore

### A.2.4 editRaspi

Questa funzione permette di modificare un Raspberry precedentemente creato.

### Parametri

Fatta esclusione per l'id tutti gli altri dati sono opzionali

**id** id del Raspberry

**name** nome del dispositivo

**tower** torre all'interno della quale il dispositivo è inserito

**position** posizione occupata all'interno della torre

**description** descrizione del dispositivo

**enable** stato del dispositivo, 0 indica disattivo (e quindi non visibile) 1 indica attivo e quindi visibile

**color** colore che dovrà avere nella visualizzazione

**Risposta**

verrà restituito 1 se la modifica ha avuto esito positivo o -1 se la modifica ha avuto esito negativo

**A.2.5 editSensor**

Questa funzione permette di aggiornare un sensore precedentemente creato

**Parametri**

Fatta esclusione per l'id tutti gli altri dati sono opzionali

**id** id del sensore

**description** descrizione del sensore

**type** identificativo della tipologia di sensore

**from** valore di soglia minimo per il sensore

**to** valore di soglia massimo per il sensore

**Risposta**

verrà restituito 1 se la modifica ha avuto esito positivo o -1 se la modifica ha avuto esito negativo

**A.2.6 moveSensor**

Questa funzione permette di spostare un sensore da un Raspberry ad un altro

**Parametri**

**sensId** id del sensore

**raspId** id del Raspberry di destinazione

**Risposta**

verrà restituito 1 se la modifica ha avuto esito positivo o -1 se la modifica ha avuto esito negativo

### A.2.7 **setEnvironment**

Questa funziona imposta la configurazione di una determinata installazione, se la configurazione non esiste verrà creata con i Parametri inviati

#### **Parametri**

**instId** identificativo dell'installazione

**apiKey** token assegnato a questa installazione

**folder** cartella nella quale il Controller dei dispositivi andrà a cercare i file da analizzare

**twitterConsumerKey** API key utilizzata per l'invio di messaggi su twitter

**twitterConsumerSecret** codice segreto utilizzato per le API

**twitterAccessToken** token utilizzato per l'invio di messaggi su twitter

**twitterAccessTokenSecret** codice segreto utilizzato per il token

**twitterContactName** nickname dell'amministratore del sistema

#### **Risposta**

verrà restituito 1 se la modifica ha avuto esito positivo o -1 se la modifica ha avuto esito negativo



# Appendice B

## Guida all'installazione

### B.1 Pre-Requisiti

Questo progetto è stato sviluppato in ambiente Linux prevedendo principalmente l'utilizzo di due tecnologie:

**JVM (*Java virtual machine*):** sul server di polling

**Web:** su API e GUI

è quindi necessario che prima di procedere all'installazione del sistema si verifichi che i seguenti prerequisiti siano soddisfatti.

#### B.1.1 JVM

Sul server che riceverà i dati dai Raspberry e che ne effettuerà l'invio al Core è necessario verificare che sia correttamente installato Java, per fare ciò è sufficiente eseguire da console il comando “java -version”, la versione minima richiesta è la 7 e quindi il comando dovrebbe dare un esito come quello indicato in figura

A screenshot of a terminal window titled "192.168.200.142 - PuTTY". The terminal shows the following text:

```
root@raspberrypi:~# java -version
java version "1.7.0_95"
OpenJDK Runtime Environment (IcedTea 2.6.4) (7u95-2.6.4-1-deb8u1+rpil)
OpenJDK Zero VM (build 24.95-b01, mixed mode)
root@raspberrypi:~#
```

Figura B.1: Versione di Java installata

nel caso in cui venga visualizzato un errore indicante “comando non trovato” sarà necessario effettuare l’installazione dell’ambiente Java denominato “JRE” (*Java runtime environment*); in distribuzioni basate su kernel debian è sufficiente eseguire il comando “sudo apt-get install default-jre”.

## B.1.2 Web

Molti dei moduli sviluppati sono basati su tecnologie web, per fare ciò è quindi necessario verificare per ognuno di essi che i seguenti prerequisiti siano soddisfatti.

### Server Web

È necessario innanzitutto che sia installato un web server, in ambiente Linux i più comuni sono Apache<sup>[23]</sup> e Nginx<sup>[24]</sup> per la cui installazione si rimanda alle rispettive guide fornite dai produttori.

### DBMS

Dovendo immagazzinare dei dati questo progetto necessita di un DBMS (*Database management system*), tutto lo sviluppo è stato effettuato utilizzando il DBMS relazionale MySQL<sup>[25]</sup>, per l’utilizzo con altri RDBMS sarà necessario analizzare ed adattare query ed oggetti utilizzati per la connessione ai database.

Per verificare che MySQL sia correttamente installato digitare il comando “mysql -version”.

Nel caso in cui sia necessario effettuare l'installazione in un ambiente con kernel Debian è sufficiente digitare i seguenti comandi:

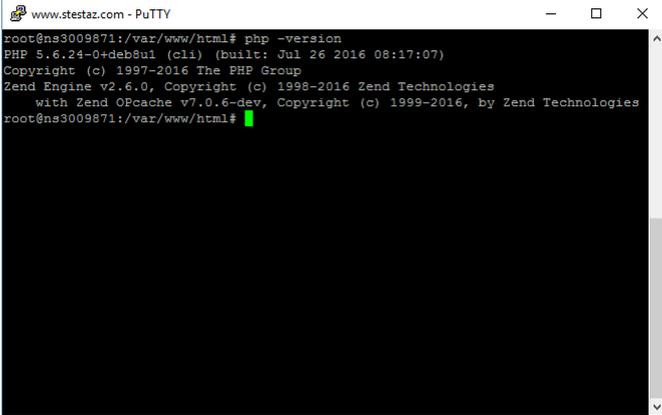
```
sudo apt-get update
sudo apt-get install mysql-server libapache2-mod-auth-mysql
php5-mysql
```

Listato B.1: Installazione Mysql

Per altri ambienti si rimanda invece alle guide disponibili sul sito di MySQL<sup>[27]</sup>.

## PHP

Verificare che sul server sia installato PHP almeno alla versione 5, per effettuare questa verifica è sufficiente, da console, digitare il comando “php -version” ed identificare la versione installata come da figura



```
www.stestaz.com - PuTTY
root@ns3009871:/var/www/html# php -version
PHP 5.6.24-0+deb8u1 (cli) (built: Jul 26 2016 08:17:07)
Copyright (c) 1997-2016 The PHP Group
Zend Engine v2.6.0, Copyright (c) 1998-2016 Zend Technologies
    with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies
root@ns3009871:/var/www/html#
```

Figura B.2: Versione di PHP installata

Nel caso in cui venga utilizzato Apache è inoltre necessario accertarsi che il modulo “mod\_rewrite” sia attivo, per fare ciò è sufficiente creare una file PHP con il seguente codice:

```
<?php
    phpinfo();
?>
```

Listato B.2: Php Info

aprendo da browser web la pagina corrispondente verranno mostrate delle informazioni dettagliate dell'installazione PHP che si sta utilizzando, nella sezione “loaded modules” dovrebbe essere mostrato come da Figura B.3 il suddetto

modulo.

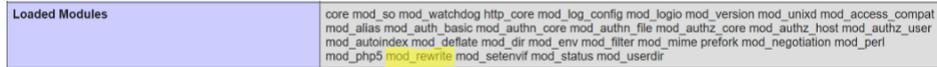


Figura B.3: Moduli Attivi

Nel caso in cui il modulo non risulti effettivamente attivo è possibile abilitarlo digitando il comando “a2enmod rewrite”.

Per permetterne il corretto funzionamento è necessario che alcune funzioni all’interno della configurazione del server web siano correttamente configurate, nel Listato B.3 è indicato il codice necessario a permettere innanzitutto l’esecuzione dello script di istruzione del modulo, questa configurazione è legata esclusivamente alla cartella in cui il modulo deve essere utilizzato.

```
<Directory "/var/www/wss/">
    Options Indexes FollowSymLinks
    AllowOverride All
    Allow from All
</Directory>
```

Listato B.3: Particolare del file di configurazione di Apache

Lo script di istruzione andrà composto all’interno del file “.htaccess” ed il codice necessario è il seguente:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ /wss/index.php [QSA,L]
```

Listato B.4: File .htaccess utilizzato

Nel caso in cui si preferisca utilizzare come server web Nginx sarà necessario modificare la configurazione del sistema inserendo la seguente direttiva:

```
location / {
    if (!-e $request_filename){
        rewrite ^(.*)$ /wss/index.php break;
    }
}
```

Listato B.5: Rewrite rule Nginx

Si consiglia inoltre di installare un tool di gestione del database basato su PHP come ad esempio phpMyAdmin<sup>[25]</sup>, questo permetterà una più agevole manutenzione dei database.

## B.2 Installazione

Nelle pagine a seguire verranno indicati i passaggi necessari all'installazione dell'intero sistema, per semplicità si è scelto di effettuare una suddivisione della guida e dei file di installazione intorno ai moduli sviluppati. Tutto il materiale necessario all'installazione è inoltre disponibile sul repository del progetto <sup>[29]</sup>.

### B.2.1 Admin

Per permettere il funzionamento della parte admin è necessario innanzitutto installare il database corrispondente, all'interno della cartella “database” è presente il file “apiKeys.sql” che contiene tutti i comandi necessari alla creazione del database ed al popolamento dello stesso con dei dati di prova iniziali.

1. accedere a phpMyAdmin
2. autenticarsi con le credenziali di MySQL
3. andare nella sezione “importa”
4. nella sezione “File Da Importare” premere sfoglia e selezionare il file apiKeys.sql
5. premere il tasto “Esegui” lasciando invariato il resto del contenuto

É necessario ora procedere alla configurazione dei parametri necessari a far sì che le pagine PHP e gli script che compongono questo sito possano correttamente interagire con il database.

1. entrare all'interno della cartella “website\inc”
2. aprire il file “settings.php” con un editor di testo
3. configurare i parametri al suo interno indicando:

**db\_server:** nome o indirizzo del server DBMS

**db\_user:** nome di un utente con privilegi di lettura e scrittura sul database

**db\_pass:** password per l'utente indicato

**db\_name:** nome del database da utilizzare, questo valore dovrebbe rimanere invariato nel caso in cui sia stata utilizzata la procedura automatizzata di creazione del database

**referrer:** indicare l'url a cui il sito risponderà, questo dato è fondamentale per il corretto accesso al sistema

#### 4. salvare il file e chiudere l'editor

Terminata la configurazione di database e parametri di connessione è possibile proseguire posizionando i file necessari all'interno del server web che dovrà rendere disponibile questo modulo, per fare ciò è sufficiente copiare l'intero contenuto della cartella "website" all'interno del server stesso.

É ora possibile verificare il corretto funzionamento del sistema, per fare ciò accedere all'url indicato precedentemente nel campo "referrer" e digitare come credenziali di accesso:

**Username:** admin

**Password:** admin

se la configurazione è stata effettuata correttamente si verrà reindirizzati alla pagina in cui gestire le api key di autenticazione. In caso contrario verificare i log del server web per identificare il problema.

Se la configurazione ha avuto esito positivo si consiglia di creare immediatamente una nuova API e disabilitare quella inserita di default, tale chiave sarà poi necessaria nei passaggi successivi.

### B.2.2 WSS

Per permettere il funzionamento di questa parte è fondamentale innanzitutto che sia stato configurato correttamente il modulo "Admin" in quanto entrambe le parti condividono un database. All'interno della cartella "database" è presente il file "raspeinModel.sql" in cui sono presenti tutti i comandi necessari alla creazione del database delle API fornite, effettuarne l'import seguendo i seguenti passaggi:

1. accedere a phpMyAdmin
2. autenticarsi con le credenziali di MySql

3. andare nella sezione “importa”
4. nella sezione “File Da Importare” premere sfoglia e selezionare il file raspeinModel.sql
5. premere il tasto “Esegui” lasciando invariato il resto del contenuto

Come precedentemente fatto è ora necessario configurare attraverso un editor di testo il file “settings.php” presente all’interno del percorso “website\inc”, i parametri necessari sono:

**db\_api\_server:** nome o ip del server in cui è installato il db contenente le api key

**db\_api\_username:** nome di un utente con privilegi di lettura sul precedente database

**db\_api\_password:** password dell’utente precedentemente indicato

**db\_api\_db:** nome del database in cui sono salvate le api key

**db\_model\_server:** nome del server in cui è stato installato il database “raspeinModel”

**db\_model\_username:** nome di un utente con privilegi di lettura e scrittura sul database raspeinModel

**db\_model\_password:** password dell’utente appena indicato

**db\_model\_db:** nome del database che si desidera utilizzare in alternativa a raspeinModel

Al termine di questa configurazione è possibile spostare tutto il contenuto della cartella “website” all’interno del server web che fornirà le funzionalità del Core, è importante verificare che questo server sia accessibile tramite protocollo http o https dagli altri moduli, in caso contrario il sistema non sarà funzionante. Questo modulo fa uso delle funzionalità di rewrite quindi è necessario verificare che la configurazione sia stata effettuata correttamente, per fare ciò è possibile richiedere l’API “getConfiguration” digitando l’url “http://url\_del\_server/wss/getConfiguration” senza indicare alcun parametro; il browser dovrebbe indicare “Errore 500”, questo significa che il modulo ha funzionato correttamente in quanto l’url è stato identificato e passato alle API che però non hanno potuto verificare l’autenticazione in quanto non è stato inviato il token di autenticazione.

### B.2.3 Polling e Comunicazione

Terminate le precedenti configurazioni si può passare alla parte riguardante i moduli di polling e comunicazione, per il funzionamento di questi moduli è innanzitutto necessario verificare che il server adibito alla loro esecuzione abbia accesso verso l'esterno in quanto deve poter interagire con le API di Twitter. È inoltre necessario creare un'applicazione twitter attraverso l'apposito portale<sup>[30]</sup>, sarà questa a permettere l'invio di messaggi attraverso il social network.

#### Device

Nei dispositivi da controllare è necessario innanzitutto creare uno script bash che rilevi i valori da monitorare e li salvi su un file che possa essere letto dal server, per semplicità è stato già creato un file di esempio, è possibile trovarlo all'interno del percorso "Polling e Comunicazione\Device" con il nome monBatch.sh.

Per ogni dispositivo lo script andrà modificato indicando:

**DEVICENAME:** nome del dispositivo

**DIRECTORY:** destinazione del file in cui salvare il risultato, questa directory deve essere accessibile al server di polling

Per terminare è necessario configurare crontab attraverso il comando "crontab -e" in modo che esegua lo script appena creato in modo periodico, un esempio è il seguente che mostra cosa indicare per effettuare una esecuzione costante con periodo di 5 minuti.

```
#run mon batch every 5 min
*/5 * * * * /home/pi/Documents/RaspeinMon/monBatch.sh
```

Listato B.6: Esempio di crontab

#### Server

Per poter funzionare la parte server necessita che sia messo in esecuzione un file jar, il modo più comodo per fare questo è creare un demone da avviare all'accensione del server. Innanzitutto è necessario configurare il file "settings.conf" indicando:

**ApiKey:** il token da utilizzare per autenticarsi con il Core

**ServerName:** url al quale risponde il Core

**instID:** id dell'installazione che si intende utilizzare

Posizionare quindi il file all'interno della cartella “/root” del server e rinominarlo in “.settings.conf”.

Se si ha dimestichezza con la creazione di servizi in sistemi Linux sarà necessario creare un demone appunto che esegua il file jar contenuto nella cartella. Nel caso in cui invece non si abbia tale dimestichezza eseguire i seguenti passaggi:

1. copiare i file “deviceMonitor” e “deviceMonitor.jar” all'interno della cartella “/etc/init.d”
2. eseguire il comando “update-rc.d deviceMonitor defaults”

Così facendo si è creato un servizio secondo i parametri inseriti all'interno del file “deviceMonitor” che, all'avvio del server, esegue il file “deviceMonitor.jar”, come per gli altri servizi sono disponibili i comandi stop/start/restart.

## B.2.4 GUI

La GUI di questo progetto è sviluppata quasi completamente utilizzando degli script jQuery e delle funzioni AJAX, si è scelto però di utilizzare un'autenticazione basata su utenti presenti in un database per poter accedere alla parte privata, sarà quindi necessario creare un piccolo database in cui inserire amministratori e relative credenziali.

1. accedere a phpMyAdmin
2. autenticarsi con le credenziali di MySQL
3. andare nella sezione “importa”
4. nella sezione “File Da Importare” premere sfoglia e selezionare il file raspeinWeb.sql
5. premere il tasto “Esegui” lasciando invariato il resto del contenuto

Configurare quindi il file “include.php” presente all'interno della cartella “website/inc”, i parametri necessari sono:

**dbUser:** utente con privilegi di lettura sul database

**dbPass:** password dell'utente

**dbServer:** indirizzo del server su cui è installato il database

**dbName:** nome del database da utilizzare, se si utilizza quello precedentemente creato non è necessario sostituire questa parte

**instId:** id relativo all'installazione presente sul Core, se è la prima effettuata lasciare il valore "1"

**apiKey:** chiave da utilizzare per autenticarsi al Core

**apiServer:** url a cui accedere per interagire con il Core

Al termine della configurazione sarà sufficiente copiare tutti i file presenti all'interno della cartella "website" nel server web che eseguirà questo modulo. Le credenziali da utilizzare per accedere alla pagina di configurazione sono:

**Username:** admin

**Password:** admin

## B.3 Configurazione

Terminata l'installazione sarà necessario accedere alla GUI ed iniziare a configurare il sistema, per fare ciò entrare nella sezione admin e configurare innanzitutto la parte "System", in questa sezione bisognerà indicare:

- percorso sul server di polling in cui vengono salvati i file dei dispositivi
- dati relativi alle chiavi dell'applicazione creata su Twitter
- contatto Twitter dell'amministratore

una volta terminata questa parte sarà possibile aggiungere le varie torri e tutti i dispositivi presenti al loro interno, il nome di ognuno di essi andrà poi indicato nel rispettivo file di configurazione così che i dati rilevati vengano correttamente associati.

# Sitografia

- [1] Gabriele D'Angelo, *Rapèin Project*, <http://pads.cs.unibo.it/doku.php?id=raspein:index>
- [2] Zoho Corp, *Site24x7 Logo*, <http://www.site24x7.com>
- [3] Nagios Enterprises *Nagios Logo*, <http://www.nagios.org>
- [4] Nagios Enterprises, *Nagios Quickstart Installation Guides*, <https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/quickstart.html>
- [5] Egalstad, *NRPE - Nagios Remote Plugin Executor*, <https://exchange.nagios.org/directory/Addons/Monitoring-Agents/NRPE-2D-Nagios-Remote-Plugin-Executor/details>
- [6] Javier Bello, *Top 15 best network monitoring tools for 2016*, <http://blog.pandorafms.org/network-monitoring-tools/>
- [7] Zabbix SIA, *Zabbix Logo*, <http://www.zabbix.com>
- [8] Zabbix SIA, *Zabbix documentation - quickstart*, <https://www.zabbix.com/documentation/3.0/manual/quickstart>
- [9] Stefano Falzaresi, *Raspèin Monitor on Twitter*, <https://twitter.com/raspeinmonitor>
- [10] Anonimo, *Bootstrap Homepage*, <http://getbootstrap.com>
- [11] W3Schools, *CSS3 Introduction*, [http://www.w3schools.com/css/css3\\_intro.asp](http://www.w3schools.com/css/css3_intro.asp)
- [12] jQuery Foundation, *jQuery API*, <http://api.jquery.com/>
- [13] Google Developers, *Google Charts Homepage*, <https://developers.google.com/chart/>
- [14] Stefano Falzaresi, *Raspèin homepage*, <http://raspein.stestaz.com/>
- [15] Stefano Falzaresi, *Raspèin admin page*, <http://raspein.stestaz.com/admin.php>

- 
- [16] Apache Software Foundation, *Apache Module mod\_rewrite*, [https://httpd.apache.org/docs/current/mod/mod\\_rewrite.html](https://httpd.apache.org/docs/current/mod/mod_rewrite.html)
- [17] HTML.it, *Riscrivere gli URL con il modulo mod\_rewrite*, <http://www.html.it/articoli/riscrivere-gli-url-con-il-modulo-modrewrite-di-apache-3/>
- [18] Anonimo, *Twitter4j Project*, <http://twitter4j.org/en/index.html>
- [19] Q-Success, *Historical trends in the usage of JavaScript libraries for websites*, [https://w3techs.com/technologies/history\\_overview/javascript\\_library/all](https://w3techs.com/technologies/history_overview/javascript_library/all)
- [20] Anonimo, *Bootstrap CSS overview*, <http://getbootstrap.com/css/>
- [21] Achecker, *Bootstrap Web accessibility checker*, <http://achecker.ca/checker/index.php>
- [22] W3C, *W3C website*, <https://www.w3.org/>
- [23] Apache Software Foundation, *Apache HTTP Server Project*, <https://httpd.apache.org/>
- [24] Nginx Inc., *Nginx*, <http://nginx.org/>
- [25] Oracle, *MySQL Homepage*, <http://mysql.com/>
- [26] PostgreSQL Global Development Group, *PostgreSQL*, <https://www.postgresql.org/>
- [27] Oracle, *MySQL Downloads and repositories*, <http://www.mysql.com/downloads/>
- [28] phpMyAdmin, *Bringing MySQL to the web*, <https://www.phpmyadmin.net/>
- [29] Stefano Falzaresi, *Monitorize Project Repository*, <https://github.com/stestaz/monitorize.git>
- [30] Twitter Inc, *Application Management*, <https://apps.twitter.com/>