Universitá del Piemonte Orientale

Dipartimento di Scienze e Innovazione Teconologica

Corso di Studi in Informatica

Relazione per la prova finale

MagicMirror(provvisorio)

Tutore interno:
Prof. Marco Guazzone

Candidato:
Riccardo Berto

Anno Accademico 2016/17

Indice

Introduzione

Nello stage che ho svolto presso Lab121 ho esteso il Magic Mirror (abbreviato in *MM*), un software di domotica sviluppato da Michael Teeuw, che gira sul calcolatore portatile RaspberryPi[3], sul quale ho svolto due principali attività:

- Lo sviluppo di applicazioni (detti anche *Moduli*) mirate al controllo del MagicMirror tramite l'impartizione di comandi vocali o input trasmessi dal movimento delle dita su una periferica touchpad.
  La scelta dello sviluppo di queste applicazioni stata presa con l'obbiettivo di dotare il MagicMirror di un'interfaccia uomo-macchina avanzata. Infatti, negli ultimi decenni, con lo sviluppo di nuove teconologie, sono stati sviluppati ed evoluti metodi non tradizionali per il controllo di dispositivi.
  Per esempio si possono osservare la maggior parte degli Smartphone moderni che permettono lo sblocco dello schermo con il riconoscimento del volto oppure l'interazione con esso tramite sintesi vocale, invece dell'utilizzo di tastiere analogie o digitali. Inoltre con il progresso tecnolgico di calcolatori e periferiche sempre più piccoli ed economici, ha permesso la nascita di una moltitudine di progetti casalinghi in ambito interazione uomo-macchina, contribuendo alla crescita dei software di questo tipo, oggi reperibili in quantitànche open source.

- Lo sviluppo di un'interfaccia grafica web, la quale, tramite un pannello di amministrazione, permette di modificare il file di configurazione del MM e delle applicazioni che si appoggiano su di esso, evitando cos di dover accedere a file potenzialmente critici ed eliminarne parti essenziali ad opera di utenti non esperti. Infatti, oltre alle teconologie, un altro aspetto che si è evoluto negli ultimi anni nel campo delle Applicazioni Web l'implementazione di interfacce grafiche di di amministrazione ad

alto livello sempre più intuitive e facili da comprendere. Capita spesso che, in assenza di queste interfacce pi evolute, un untente inesperto sia costretto ad utilizzare interfacce a basso livello (come strumenti di linea di comando o file di configurazione), con il rischio di causare potenziali danni.

Nei prossimi capitoli:

- Capitolo 1 (label) verranno trattati scopi e problemi affrontati nello stage
- Capitolo 2 (label) verranno elencate e spiegate le teconologie che sono state utilizzate dutante lo stage (verr spostato poi come ultimo capitolo)
- Capitolo 3 (label) verrà spiegata la struttura del Magic Mirror

Capitolo 1

Scopi e Problemi Affrontati

In questo capitolo verranno esposti gli scopi e i problemi incontrati durante l'attivit di stage divisi in due argomenti. In particolare:

- Nella sezione 1.1 si tratta lo sviluppo delle applicazioni per il MM
- Nella sezione 2.1 si tratta lo sviluppo dell'interfaccia grafica web

1.1 Applicazioni del MagicMirror

Il Magic Mirror è un software che espone API, le quali permettono l'integrazione di applicazioni sviluppate da terzi e forniscono funzionalit per la comunicazione, gestione ed organizzazione tra le applicazioni. Lo scopo nello sviluppo delle applicazioni, durante lo stage, è stato di creare programmi che permettessero l'interazione tra il software principale e l'essere umano. Nello specifico si è voluto implementare la possibilità, da parte di un utente, di controllare le applicazioni attive sul Magic Mirror tramite l'impartizione di comandi vocali o movimenti delle dita su una scheda Touch Board. In questa parte si è dovuto affrontare il problema di far comunicare i diversi dispositivi hardware con il calcolatore principale, e di programmare le applicazioni con linguaggi diversi, a seconda della compatibilità di un linguaggio e delle librerie a disposizione di una determinata periferica.

1.2 Interfaccia di controllo del MagicMirror

Il MM possiede all'interno delle sue cartelle un file di configurazione, il quale permette di configurare sia MM che le applicazioni caricate su quest'ultimo. Prima dello stage per modificare la configurazione, bisognava accedere direttamente al file di configurazione appena presentato e modificarlo. Questo approccio, oltre ad essere scomodo, pu anche causare errori, dal momento che necessaria una sintassi corretta all'interno del file. Nella seconda parte dello stage si è svolta la progettazione e lo sviluppo di un'interfaccia Web di configurazione remota il cui scopo è di permettere ad un utente autenticato di gestire le configurazioni del MM e dei moduli implementati su di esso da una pagina web, senza dover accedere alla macchina fisica su cui gira l'applicazione ed implementando una validazione per la sintassi del file di configurazione. L'interfaccia permette anche di disattivare (o attivare) le applicazioni presenti nel MM, la cui procedura, prima, richiedeva di aggiungere manualmente un pezzo di codice nella configurazione del software principale. Lo scambio dei messaggi tra il MM e la pagina web avviene in formato JSON, lo stesso formato usato per il file di configurazioni.
Un altro problema riscontrato in questa parte stata la difficolt dell'utilizzo di Javascript perch è un linguaggio orientato ad eventi, ovvero alcune funzioni vengono eseguite senza attendere il risulato della precedente. Il linguaggio, per poter sincronizzare la funzioni, mette a disposizione le callback: funzioni passate come parametro alla funzione principale che a loro volta avevano come paramentro il risultato, in modo da poter eseguire le operazioni solo dopo che la funzione chiamante avesse prodotto l'output. Senza buone pratiche di programmazione si rischia di incontrarsi in un fenomeno chiamato "Callback Hell", ovvero Callback che chiamano a loro volta altre Callback creando funzioni annidate più volte, come mostrato in figura 1.1. (spiegare la figura e modificarla nel caso)

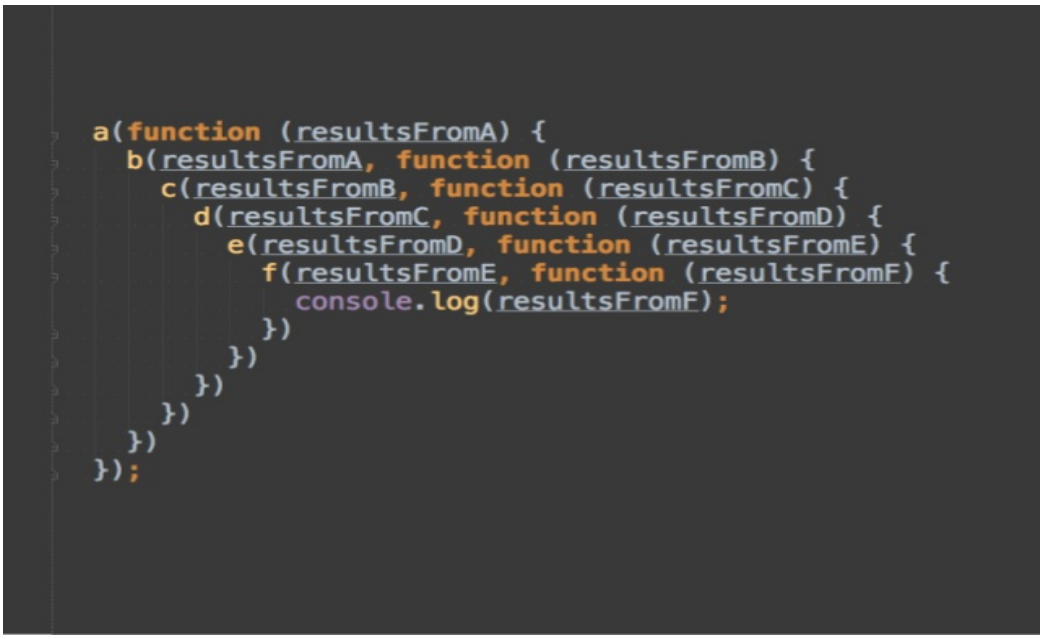

Figura 1.1: Esempio di Callback Hell

Capitolo 2

Tecnologie Implicate

In questo capitolo verranno discusse le teconologie che sono state studiate ed utilizzate durante le attivit di stage. In particolare il capitolo diviso tra teconologie hardware e tecnologie software.

2.1 Prefazione

Lo sviluppo del progetto è stato svolto nell'ambiente Raspbian[1] una distribuzione Debian[2] che gira sul dispositivo RasperryPi[3]. Inoltre sono state adottate diverse tecnologie nel campo del riconoscimento vocale e di immagini (OpenCV [4] e Google Speech API [5]), nello sviluppo di applicazioni web (Electron [6], NodeJS [14], Express [8], Mustache [10]), della gestione dati (MySQL [9]). Inoltre sono stati adottati diversi linguaggi di programmazione (JavaScript [11], Python [12]) e piattaforme per la gestione distribuita di progetti softwave (GitLab [13]).

2.2 Hardware

2.2.1 RasperryPi

RaspberryPi è un calcolatore elettronico, montato su una singola scheda elettronica, caratterizzato dal basso costo, dal consumo energetico ridotto e, per le sue dimensioni ridotte, dalla facile portabilità. Rilasciato per la prima volta nel 2012 è diventato un prodotto utilizzato per una moltitudine di progetti sia aziendali che casalinghi. Il modello usato durante lo stage è RaspberryPi 3 model B e monta:

- CPU con architettura ARM (Advanced RISC Machine)

- 1 porta HDMI
- 1 porta LAN
- 1 uscita Aux
- 4 porte USB
- 40 pin General Purpose Input/Output(GPIO)
- 1 scheda di rete wireless
- Alimentazione microUSB 5V
- un bus camera serial interface(CSI), ovvero una porta per telecamere con Flexible flat cable(FFC)
- ingresso per microSD

Il sistema operativo per Raspberry deve essere installato su una microSD opportunamente formattata e configurata con un Master Boot Record (MBR).

2.2.2 Periferiche

Nella creazione delle applicazioni per il Magic Mirror sono state usate diverse periferiche, tra cui un microfono USB, per catturare la voce in input e un componente Skydriver Touch Board (94mm x 122mm) di Piromoni collegabile tramite i 40 pin GPIO del calcolatore principale, per catturare input fisici tramite il movimento delle dita sulla Touch Board.

2.3 Software

2.3.1 Raspbian

Raspbian è una distribuzione del sistema operativo Linux derivata da Debian, completamente libera, ottimizzata per Raspberry. Fu sviluppata da Mike Thompson e Peter Green come progetto non affiliato alla compagnia RaspberryPi Fundation, per tenere in considerazione la limitata capacit di calcolo dei processori ARM. La prima versione venne rilasciata nel 2012.

2.3.2 Electron

Electron è un framework open source rilasciato per la prima volta nel 2013, ma la prima versione stabile è uscita durante giugno 2017. È disponibile per i sistemi operativi Windows, MacOS e Linux ed è scritto in C++ e Javascript. Il framework permette la creazione di applicazioni multipiattaforma utilizzando teconologie già esistenti per lo sviluppo del lato client e del lato server (Javascript, NodeJS, V8 [15]). All'avvio di Electron viene inizializzata una pagina con Chromium [17](un web browser installato insieme all'applicazione) nel quale viene mostrata una pagina web, e un server in NodeJS. Un'applicazione sviluppata con Electorn ha bisongo di 3 componenti principali:

- Il package.json, un file JSON, che deve contenere almeno il nome dell'applicazione, la versione dell'applicazione creata, la descrizione di quest'ultima e il nome del file principale dell'applicazione (necessaria per l'avvio), come mostrato nella seguente immagine:

```
1 {
2   "name": "magicmirror",
3   "version": "2.1.1",
4   "description": "The open source smart platform",
5   "main": "js/electron.js"
6 }
```

  Name il campo che assegna il nome dell'applicazione, version il cmapo che assegna la versione dell'applicazione, description una stringa che descrive il programma, main il cmapo che punta al file Javascript eseguibile

- Un file HTML che contiene il template della pagina generata dall'applicazione

- Un file JavaScript che contiene il codice di esecuzione dell'applicazione come ad esempio la creazione di una finestra o la visualizzazione di una pagina.

2.3.3 OpenCV

OpenCV (Open Source Computer Vision Library) è una libreria software sviluppata intorno al 2000 utilizzata nell'ambito della visione artificiale per l'acquisizione e il riconoscimento di immagini da parte di una macchina per mezzo di input digitali, ottenuti tramite telecamera o fotocamera.

La libreria è disponibile per i linguaggi C++(linguaggio in cui è scritta), C, Python e Java e per diversi sistemi operativi, compresi quelli specifici per i dispositivi mobili.
OpenCV prende in input un'immagine o uno stream (come un video o una serie di immagini) e, l'utilizzo di specifici algoritmi di Machine Learning per individuare e riconoscere oggetti specifici.

2.3.4 Google Speech API

Negli ultimi anni Google ha ampliato sempre di più il suo catalogo per quanto riguarda i servizi cloud e web API. Tra questi si può individuare anche Google Speech API, il quale è un servizio che, ricevendo in input un file o uno stream audio, ottenuto per mezzo di un'acquisizione da un dispositivo di audio input, traduce il parlato in testo scritto tramite algoritmi avanzati di riconoscimento della voce.
L'API supporta oltre 110 lingue e si possono usare su diverse piattaforme dato che le librerie sono disponibili nei linguaggi C#, GO, Java, Node.JS, PhP, Python e Ruby. Inoltre Google Speech To Text dispone di alcune varianti:

- Una con interfaccia REpresentational State Transfer(REST), che comunica per mezzo di URI
- Una con gRPC, un sistema di chiamata di procedura remota

2.3.5 Node.JS

Node.JS era gi implementato con Electron, ed stato usato per la creazione del server per la gestione dell'interfaccia web per il MM. Node.JS è una piattaforma open source che permette l'esecuzione del linguaggio Javascript anche per il lato server, ovvero la parte del sistema che contiene applicazioni e programmi scritti, in questo caso, in Javascript ed eseguiti sfruttando il motore JavaScript V8 sviluppato da Google. Il lato server una parte del sistema che l'utente, di solito, non interagisce direttamente e che ne permette l'effettivo funzionamento e, nel caso, la manipolazione e l'elaborazione dei dati.
Node.JS esegue delle operazioni al verificarsi di uno specifico evento, che può essere un accesso ad una porta del server o la richiesta di una pagina. Per gestire i pacchetti di questo framework viene utilizzato NPM[16], uno strumento che permette di scaricare ed installare librerie private o pubbliche salvate su un database.

2.3.6 Express

Express è un framework per Node.JS che permette di creare applicazioni Web e API in JavaScript, offrendo strumenti e pacchetti che implementano pi facilmente tutti i servizi (e anche di pi) offerti da Node.JS. Il software viene usato per creare e gestire il backend di un server, che è composto di 3 entità importanti:

- il Controller, che definisce le funzioni associate ad un determinato modello.
- il Routing, utilizzato per determinare come il server debba rispondere ad un determinato metodo di richiesta, ricevuta sottoforma di URI, inoltrando la richiesta alla funzione del controller del modello a cui fa rifermento.
- i Modelli, creati per ogni entità-oggetto che esiste all'interno del server, ad ognuno dei quali viene associato un controller. Inoltre, tramite i modelli si accede al databse per estrarre i dati e spedirli al controller per la manipolazione, ricevendoli successivamente modificati e salvandoli, se necessario.

2.3.7 Mustache

Mustache un motore di template disponibile per molti linguaggi tra cui Javascript. È stato usato come motore di template per l'interfaccia web di configurazione del MagicMirror. Mustache un sistema logic-less perch privo di istruzioni per il controllo del flusso (come ad esempio l'if e l'else), quindi tutti i controllli di questo tipo vengono fatti attraverso programmazione data-driven. Il nome di questo motore dovuto all'utilizzo delle parentesi graffe, usate spesso all'intero della sua sintassi, che, se viste da una certa angolatura, assomigliano a un paio di baffi.

2.3.8 MySQL

MySQL stato usato durante lo stage per l'archiviazione di utenti con i permessi accedere all'interfaccia web del MM. MySQL uno tra i pi famosi database open source sviluppato da Oracle, pi precisamente sistema per la gestione di basi di dati basato sul modello relazionale. MySQL composto

da una semplice riga di comando e un server web, ma sono implementati anche programmi per l'amministrazione del database (come ad esempio il famoso phpMyAdmin). La prima versione fu rilasciata nel maggio del 1995 Sviluppato da Oracle e di propriet di MySQL AB, distribuito sia con licenza commerciale sia con licenza libera.

2.3.9 JavaScript

Javascript è un linguaggio di programmazione ad alto livello orientato agli oggetti e ad eventi, che è supportato da tutti i browser per lo scripting delle pagine web, e che supporta la programmazione procedurale. Inizialmente usato per il lato client ha subito un'evoluzione che lo ha portato ad essere utilizzato per lo sviluppo di backend e web app.

ECMAJavascript(ES) è lo standard di Javascript che negli ultimi anni ha sviluppato ed evoluto il linguaggio in diverse versioni. In tutte il problema più trattato è il fatto che Javascript è orientato agli eventi, ed alcune funzioni necessitano di Callback, che possono degenerare se non vengono utilizzati buone pratiche di programmazione. Le callback sono necessarie perchè alcune chiamate di funzioni non vengono fatte direttamente, ma vengono fatte attraverso messaggi, i quali vengono salvati in una coda di messaggi e vengono spediti sequenzialmente ad uno stack di chiamata dove viene salvata la corrispondente funzione per l'esecuzione. Questo metodo rende il linguaggio asincorno perchè le funzioni e gli eventi vengono eseguiti in successione senza attendere il termine della precedente.

Le soluzioni adottate in ECMA Javascript 5 (ES5), la versione usata durante lo stage, sono l'utilizzo delle Callback gi discusse nella sezione 1.2 (label).

In ECMA Javascript 6 (ES6) sono state introdotte le Promises, ovvero al posto di far tornare una funzione Callback, ritorna una Promise(Promessa), la quale garantisce che una variabile/oggetto avrà un ritorno, mettendo cosi la funzione in attesa fino al ricevimento del valore o di un errore. Sono tutt'ora in via di sviluppo nuove versioni di Javascript.

2.3.10 Python

Questo linguaggio è stato particolarmente utile per l'interfacciamento con la telecamera e la Touch Board, essendo a disposizione librerie apposite per gestirle. Python è un linguaggio di programmazione open source, ad alto

livello con semantica dinamica, orientato ad oggetti, usato per sviluppo di applicazioni e scripting.

Come molti altri linguaggi supporta pacchetti anche sviluppati da terzi, salvati in una repository pubblica, attraverso un suo gestore di pacchetti Pip Installs Packages o Pip Installs Python (pip, acronimo ricorsivo).

Inoltre con Python è possibile creare ambienti isolati per l'utilizzo di pacchetti e moduli senza doverli installare all'interno del sistema. Per poterlo fare è disponibile virtualenv, uno strumento Python che permette, appunto, di creare ambienti virtuali (virtual enviroments).

2.3.11 GitLab

GitLab è una piattaforma web che implementa le funzionalità offerte dal software Git e altri servizi tra i quali la possibilità di creare wiki e un servizio di issue tracking, utile per tenere traccia di eventuali richieste o problemi. La comodit di questa piattaforma sta nel poter creare dei Commit, ovvero ad ogni modifica del codice si pu salvarne lo stato assegnadoli un'etichetta in locale, per poi spedire la versione al remoto del progetto. In ogni momento si pu tornare ad una versione vecchia tramite gli strumenti di reversering offerti dal servizio.

Capitolo 3

Struttura del Magic Mirror

Il Magic Mirror è un progetto ideato e sviluppato da Michael Teeuw, successivamente esteso nelle sue funzionalità da una moltitudine di utenti su GitHub. Una prima versione è stata scritta completamente in Python, mentre successivamente è stata creata una seconda versione nella quale si è preferito l'utilizzo di Electron, che ha comportato una variazione di linguaggio, a favore di Javascript. In questo modo è stato possibile implementare un'interfaccia esteticamente più gradevole e API più intuitive. L'idea dell'autore è nata rifacendosi allo specchio magico dell'omonima fiaba scritta dai fratelli Grimm, Biancaneve e i Sette Nani.
Il software viene mostrato attraverso un comune monitor, trasmettendo immagini poste su uno sfondo completamente nero. Applicando sopra una semplice pellicola a specchio (la quale da un lato permette di specchiarsi e dall'altro di vedere attraverso) si crea un effetto particolare per cui una persona riesce a specchiarsi e allo stesso tempo riesce a vedere le scritte o le immagini trasmesse dal monitor, come mostrato in figura 3.1.

Figura 3.1: Magic Mirror by Michael Teeuw

3.1 Il MM dall'alto livello

Il MM una applicazione con diverse API e strutture che permettono l'interazione e la comunicazione tra i moduli. Si possono individuare alcuni elementi principali rappresentati in figura 3.2:

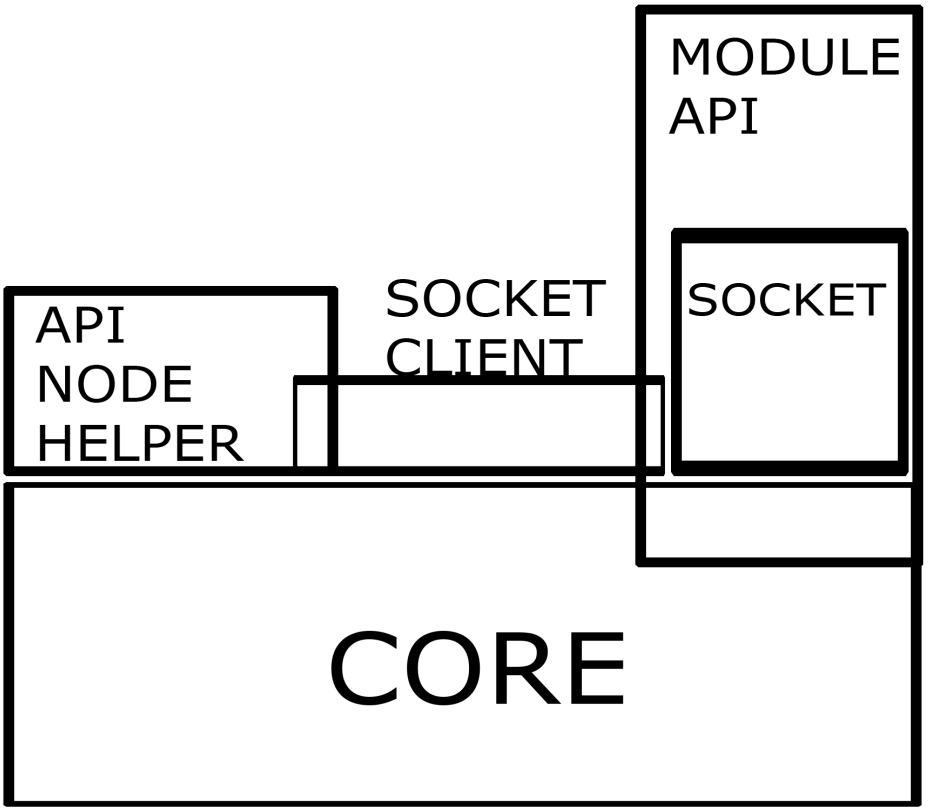

Figura 3.2: Struttura MagicMirror

- Il core, il cuore del MM dove gira il codice principale.
- Le API per i moduli, i metodi e funzionalit esposti dla MM per poter itegrare i moduli creati
- Le API per i Node Helper, i metodi e funzionalit esposti dla MM per poter itegrare i NodeHelper dei relativi moduli
- Il Socket, entit che si trova in mezzo alle API e ne fornisce un canale di comunicazione
- Il SocketClient, entit che si trova tra le API delle applicazioni e dei Node Helper, e fornisce il canale di comunicazione tra questi due

3.2 Avvio ed Escuzione

Il Magic Mirror viene avviato tramite linea di comando, la quale esegue il codice di un file Javascript, il cui percorso viene indicato nel documento *package.json*, descritto nella sezione 2.3.2. Il file contiene il codice di Electron, che si occupa della creazione di una nuova finestra browser, il cui compito di mostrare l'output dei moduli del MM, e l'avvio del codice che rappresenta il core, che carica tutte le strutture princiapli del MM. La pagina costruita e l'output dei moduli sono un insieme di Document Object Model (DOM), ovvero oggetti HTML che compongo una pagina web, che vengono mostrati solo dopo l'avvio di tutti i moduli attivi. Le strutture principali caricate dal MM sono le seguenti:

- gli end-point delle API per le applicazioni, che sono le interfacce usate per leggere e caricare le applicazioni inserite nel Magic Mirror.
- gli end-point delle API per i Node Helper, per la gestione dei questi ultimi. Sono strutture opzionali usate per collegamenti esterni al Magic Mirror (per esempio, con API di un servizio cloud). Ogni applicazione ha il proprio Node Helper con cui può comunicare tramite messaggi in modo simile a come comunicano le applicazioni tra di loro.
- un "Socket" e un "Socket-Client", entità principali che definiscono le funzioni per lo scambio dei messaggi tra le applicazioni e i rispettivi Node Helper.
- un Logger, implementato per tenere i log dell'applicazione e degli evenutali errori. Usato pricipalmente per il debugging.
- un file di configurazione, nel quale sono specificati il nome e le coordinate per la posizione dei vari moduli all'interno della pagina.

Inoltre viene inizializzato un server, il cui compito è quello di trasmettere la pagina resa, con gli output delle varie applicazioni, al browser precedentemente avviato.

3.2.1 Il file di Configurazione

Come discusso nella sezione **??**, il Magic Mirror carica un file di configurazione, che composto dai seguenti campi:

- la porta del server
- una whitelist, ovvero un IP oppure un range di IP che possono collegarsi allo specchio
- la lingua principale del sistema
- il formato dell'ora (12h o 24h)
- unità di misura usata (ad esempio, metrica), usata per gestire la distanza dei DOM
- una lista di applicazioni (in formato JSON) da caricare con la relativa posizione nella pagina. Per ogni applicazione deve necessariamente comparire il nome e la posizione; opzionalmente si possono inserire un campo *header* e un campo *config* specifico per l'applicazione, come mostrato nell'esempio in figura.3.2.1

```
1 {
2         "module": 'nome dell'applicazione',
3         "position": 'la posizione dell'applicazione
              all'intenro dello specchio',
4         "header": 'una stringa che viene stampata
             sopra all'applicazione',
5         "config": { opzioni varie in formato JSON }
6 }
```

La lingua, il formato dell'ora e l'unità di misura usata sono parametri messi a disposizione dal sistema per la creazione di un'applicazione (ad esempio, il display di un orologio).

3.3 Implementazione di un'applicazione

La modifica del file di configurazione del Magic Mirror appena descritta serve per "notificare" la presenza delle applicazioni a quest'ultimo, ma perchè possano funzionare è necessario che rispettino alcune specifiche regole. Per inserire il codice dell'applicazione all'interno dello specchio è necessario creare una cartella con un nome identificativo dell'applicazione nella directory *Modules*. Dentro la cartella appena creata devono essere inseriti:

- 1 file Javascript (JS), ovvero il documento principale con lo stesso nome della cartella appena creata. Contiene il codice dell'applicazione, il quale conterrà a sua volta il codice per la creazione dei DOM
- 1 file Cascading Style Sheets (CSS), per modificare l'estetica del DOM della relativa applicazione (opzionale)
- 1 file node_helper.js, che è il Node Helper associato alla specifica applicazione (opzionale)
- Altri file necessari all'applicazione (immagini, JSON, etc)

3.4 Messaggistica del MM

In precedenza è stato già accennato che il Magic Mirror implementa un meccanismo di messaggistica sfruttando un sistema di socket integrato, utile per l'organizzazione e la moderazione delle applicazioni tramite l'utilizzo di funzioni messe a disposizione dall'API. Le funzioni usate per ricevere i messaggi, come già detto precedentemente, sono: *socketNotificationReceived(notification, payload)*, per la ricezione dei messagi da parte del Node Helper, e *notificationReceived(notification, payload, sender)*, per la ricezione di messaggi dalle altre applicazioni. Per spedire messaggi vengono chiamate le funzioni *sendSocketNotification(notification, payload)* e *sendNotificationReceived(notification, payload, sender)*, i cui campi indicano:

- notification, l'indentificatore della notifica, una sorta di header al quale si può assegnare un qualunque oggetto
- payload, il corpo del messaggio (opzionale)
- sender, usato solo nei messaggi tra i moduli, dove viene riportato l'identificativo dell'applicazione che lo ha mandato

sendNotificationReceived a differenza del suo corrispettivo per il Node Helper, manda il messaggio in broadcast a tutte le applicazioni attive sulla macchina, e deve essere filtrato dalle singole applicazioni in base al campo *notification*.

Capitolo 4

Applicazioni per il MM

4.1 Modulo per comandi vocali

La prima applicazione implementata, come stato accennato nell'introduzione, stata il controllo del Magic Mirror tramite comandi vocali. Nello specifico l'applicazione doveva, tramite delle specifiche frasi, gestire le altre applicazioni presenti nel Magic Mirror.
Il compito di trasformare da parlato a testo scritto e l'elaborazione delle frasi stato lasciato a carico di Google Speech API. Prima di spiegare il codice ci sono alcune dipendenze da spiegare.

4.1.1 Sound eXchange (SoX)

Per permettere al microfono di catturare ed elaborare correttamente l'audio nell'ambiente Raspbian necessario installare Sound eXchange (SoX), un software per la manipolazione dell'audio, utile per effettuare uno streaming dell'audio. Per installarlo correttamente necessario avere il Sistema Operativo aggiornato, usando il seguente comando da Shell:

```
1   $ sudo apt-get update
```

Il comando *sudo* un controllo speciale che permette di eseguire i comandi in modalit Super User, un utente speciale che ha i privilegi di amministrazione all'interno del sistema. *apt-get* uno strumento da riga di comando che permette di usare l'APT (Advanced Packaging Tool), un gestore di pacchetti dei sistemi Debian. Il parametro *get* indica l'operazione da eseguire con *apt*, in questo caso scaricare un pacchetto. *update* indica l'obbiettivo di cercare aggiornamenti del S.O. Una volta completato si pu installare il software sempre tramite comando da Shell:

```
1   $ sudo apt-get install sox
```

Il parametro *install* serve ad indicare di installare il pacchetot oltre a scaricarlo, mentre *sox* il nome del pacchetto che si vuole cercare. Al termine del processo necessario collegare il driver audio di Raspbian ALSA (Advanced Linux Sound Architecture) a SoX e indicare a quest'ultimo a quale periferica interfacciarsi, tramite i comandi:

```
$ export AUDIODEV=hw:1,0
$ export AUDIODRIVER=alsa
```

export un comando Linux usato per assegnare valori a Variabili d'Ambiente. *AUDIODEV* la variabile d'ambiente che identifica il dispositivo audio di Default, mentre *hw:1,0* indica l'assegnamento della periferica 0 presente sulla scheda 1. il comando *export AUDIODRIVER=alsa* opzionale, perch assegna alla variabile d'ambiente *AUDIODRIVER*, la varibiale che fa riferiemnto al driver audio del sistema, ALSA, il quale dovrebbe essere gi quello predefinito.

4.1.2 Autenticazione Google API

Per poter usufruire dell'API di Google necessario fornire un'autenticazione a livello di sistema. Per poterlo fare necessario ottenere delle credenziali di sicurezza per un account Google, attivabili tramite Google Cloud Platform Console. Le credenziali consistono in un username, la mail dell'account google, e una chiave di sicurezza unica, entrambi sono contenuti in un file JSON che pu essere scaricato e salvato in locale. Per poter permettere al sistema di utilizzare l'API bisogna fare in modo che le credenziali siano raggiungibili, inserendole in una Variabile d'Ambiente con il comando:

```
$ export GOOGLE_APPLICATION_CREDENTIALS='Percorso del
    JSON'
```

4.1.3 Node Heloer dell'applicazione

Il node Helper dell'applicazione si occupa di gestire lo streaming con L'API e di mandare i risultati (o gli errori) all'applicazione tramite le funzioni messe a disposizione dal Magic Mirror mostrate precedentemente.

```
var NodeHelper = require("node_helper");
const record = require('node-record-lpcm16');
const Speech = require('@google-cloud/speech');
const speech = Speech();
const request = {
  config: {
    encoding: encoding /*(Es. LINEAR16)*/,
```

```
8         sampleRateHertz: sampleRateHertz /*(Es. 16000)*/,
9         languageCode: languageCode /*(Es. it-IT)*/
10      },
11      interimResults: false /*usato se si vuole conoscere
            l'output prima dell'elaborazione*/
12    };
13
14  module.exports = NodeHelper.create({
15    start: function () {
16          console.log("Starting Node Helper of module: "
              +this.name);
17      },
18
19      socketNotificationReceived: function(notification,
          payload) {
20          var self = this;
21          if (notification === "ready") {
22              listen();
23          }
24      },
25
26      listen: function(){
27        // Create a recognize stream
28        const recognizeStream = speech.streamingRecognize
            (request)
29          .on('error', sendSocketNotification("error"))
30          .on('data', (data) =>
31            if(Transcription: ${data.results[0].
                alternatives[0].transcript}\n)
32              sendSocketNotification('limit_reached')
33            else
34              sendSocketNotification('response', data.
                  results[0])
35
36        // Start recording and send the microphone input
            to the Speech API
37        record
38          .start({
39            sampleRateHertz: 1600,
40            threshold: 0,
41            verbose: false,
```

```
42              recordProgram: 'sox',
43              silence: '20.0'
44            })
45            .on('error', sendSocketNotification('error'))
46            .pipe(recognizeStream);
47      }
48  })
```

4.1.4 Il Main

Il main dell'applicazione gestisce le risposte ricevute dal NodeHelper ed esegue i comandi impartiti da esse. Inoltre si occupa di mostrare a schermo quando il programma "in ascolto".

```
1  Module.register("voicecontrol",{
2  // Default module config.
3  defaults: {
4    rosso: "modules/voicecontrol/icons/redpin.png",
5    verde: "modules/voicecontrol/icons/greenpin.png",
6    ready: false,
7    modules: [],
8    current: "",
9  }
10
11 start:function(){
12   console.log("module"+ this.name+"started");
13 }
14
15 getDom: function(){
16   var wrapper = document.createElement("div");
17   var containerled = document.createElement("div");
18   var containerlist=document.createElement("div");
19   var list=document.createElement("ul");
20   var led=document.createElement("span");
21   led.className="ledcontainer";
22   var imag = document.createElement("img");
23   if(!this.config.ready){
24     imag.src=this.config.rosso;
25     led.innerHTML="Not Listening...";
26   }
27   else
```

```
28        imag.src=this.config.verde;
29        led.innerHTML="On Air";
30      }
31      imag.style.height = '30px';
32      imag.style.width = '30px';
33      for(var i=0; i<this.config.modules.length; i++){
34        var option=document.createElement("li");
35        if(modules[i]==current)
36          option.style.color = 'red';
37        option.innerHTML=modules[i];
38        list.appendChild(option)
39      }
40      led.appendChild(imag);
41      containerled.appendChild(led);
42      containerlist.appendChild(list);
43      wrapper.appendChild(containerled);
44      wrapper.appendChild(containerlist);
45      return wrapper;
46  }
47
48  notificationReceived: function(notification, payload,
        sender) {
49                      if (notification === '
                            ALL_MODULES_STARTED') {
50                                MM.getModules().exceptModule(
                                      this).enumerate(function(
                                      module) {
51                                          this.config.modules.
                                              push(module.name);
52              module.hide(1000, function() {
53                                                  //Module hidden
                                                        .
54                                          });
55                                          });
56                          });
57                }
58      if (notification === 'DOM_OBJECTS_CREATED') {
59        this.sendSocketNotification('ready');
60        this.config.ready=true;
61        this.updateDom();
62      }
```

```
63 },
64
65 socketNotificationReceived: function(notification,
       payload) {
66     if (notification === 'limit_reached') {
67       this.config.ready=false;
68       this.updateDom();
69     }
70     else if(notification === 'error'){
71       this.config.ready=false;
72       this.updateDom();
73     }
74     else if(notification === 'response'){
75       for(var j=0; j<this.config.modules.length; j++){
76         if(payload===modules[j]){
77           this.config.current=true;
78         }
79       MM.getModules().exceptModule(this).enumerate(
           function(module) {
80                                     if(module.name==this.
                                          config.current)
81           module.show(1000, function() {
82                                           //Module show.
83                                     });
84                                     });
85                   });
86       this.updateDom();
87     }
88   }
89 }
```

Bibliografia

[1] Raspbian wikipedia, https://it.wikipedia.org/wiki/Raspbian

[2] Debian wikidia, https://it.wikipedia.org/wiki/Debian

[3] Raspberry official website, https://www.raspberrypi.org/

[4] OpenCV official website, http://opencv.org/

[5] Google Speech to Text API documentation, https://cloud.google.com/speech/

[6] Electron official website, https://electron.atom.io/

[7] MVC Wikipedia, https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[8] Express official website, http://expressjs.com/it/

[9] MySQL official website, https://www.mysql.com/it/

[10] Mustache Git Site, https://mustache.github.io/

[11] JavaScript, https://www.javascript.com/

[12] Python website, https://www.python.it/

[13] GitLab website, https://about.gitlab.com/

[14] Nodejs website, https://nodejs.org/it/

[15] V8 wikipedia, https://it.wikipedia.org/wiki/V8_(motore_JavaScript)

[16] npm docs, https://docs.npmjs.com/getting-started/what-is-npm

[17] Chromium wikipedia, https://it.wikipedia.org/wiki/Chromium