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 stato studiato il Magic Mirror, un software di domotica sviluppato da Michael Teeuw, che gira sul calcolatore portatile RaspberryPi[3], sul quale ho svolto due principali attivit:

- Lo sviluppo di applicazioni (detti anche moduli) che si interfacciano con periferiche esterne, o con api (italic), e il software principale.
- Lo sviluppo di un'interfaccia web che permette di modificarne la configurazione senza dover accedere a file potenzialmente critici ed eliminarne parti essenziali ad opera di utenti non esperti.

Per la prima parte dello stage sono state create applicazioni mirate al controllo del Magic Mirror tramite l'impartizione di comandi vocali o input trasmessi dal movimento delle dita su una periferica touchpad.
Questa scelta è stata presa prendendo che nel corso degli ultimi decenni, con il perfezionamento delle teconologie e dei software, ha fatto nascere una tendenza a controllare macchine o dispositivi con metodi sempre meno "diretti". 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. Inoltre, con lo sviluppo di calcolatori e periferiche sempre pi piccoli ed economici ha permesso la nascita di una moltitudine di progetti casalinghi con questo fine, contribuendo cos alla crescita dei software in ambito relazione uomo-machcina, che al giorno d'oggi se ne possono trovare una moltitudine open source.

Un altro aspetto che si è evoluto soprattuto negli ultimi anni nel campo delle applicazioni e delle Applicazioni Web (chiamate anche webb app) è l'implementazione di interfacce grafiche di gestione sempre più intuitive e facili da comprendere. Capita spesso che un'applicazione venga utilizzata da utenti

privi di consocenze informatiche, i quali, se mancasse l'interfaccia di gestione, dovrebbero accedere direttamente ai file di configurazione per modificarli, e potrebbero causare potenziali danni.
A tal fine, nella seconda parte dello stage, è stata sviluppata un'interfaccia grafica web, la quale mostra un pannello di configurazione il compito è di permettere la corretta modifica dei file di configurazione delle applicazioni e del Magic Mirror.

Nei caitoli successivi verr spiegata la struttura del Magic Mirror

Capitolo 1

Scopi e Problemi Affrontati

1.1 Applicazioni del MagicMirror

Il Magic Mirror è un software che implementa api, le quali permettono l'integrazione di applicazioni sviluppate da terzi e forniscono strumenti per la comunicazione, gestione ed organizzazione tra di esse. 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 da movimenti delle dita su una scheda touchpad. 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

Nella seconda parte dello stage si è svolto la progettazione di un'interfaccia Web di configurazione remota il cui scopo è di permettere ad un utente autenticato di gestire le configurazioni delle applicazioni implementate nel software da una pagina web, senza dover accedere fisicamente alla macchina. Quest'ultima permette anche di disattivare (o attivare) le applicazioni presenti nel Magic Mirror, la cui procedura, prima, richiedeva di aggiungere manualmente un pezzo di codice nella configurazione del software principale.

Nello sviluppo dell'interfaccia si è affrontato il problema di dover modificare il documento di configurazione del software principale senza eliminarne parti essenziali o modificarlo in un formato sbagliato. Lo scambio dei messaggi tra il Magic Mirror e la pagina web avviene in formato JSON, lo stesso formato usato per il suo file di configurazione e delle sue applicazioni.

Capitolo 2

Tecnologie Implicate

Lo sviluppo del progetto é stato svolto nell'ambiente Raspbian[1] una distribuzione Debian[2] che gira sul dispositivo RasperryPi[3]. Inoltre sono state adottate diverse tecnologie, recenti e non, nel campo della creazione di applicazioni(Electron [6], OpenCV [4], GoogleSpeechRecognition [5]), dei web server (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]).

2.1 Hardware

2.1.1 RasperryPi

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

- 1 porta HDMI
- 1 porta LAN
- 1 uscita Aux
- 4 porte USB
- 40 pin General Purpose Input/Output(GPIO)

- 1 scheda di rete wirless
- 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 miscro-SD opportunamente formattata e configurata con un Master Boot Record (MBR).

2.1.2 Periferiche

Nella creazione delle applicazioni per il Magic Mirror sono stati usati 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 scheda.

2.2 Software

2.2.1 Raspbian

Raspbian é un sistema operativo multi-architettura della distribuzione Debian, completamente libero, ottimizzato per Raspberry. Fú sviluppato da Mike Thompson e Peter Green come progetto non affiliato alla compagnia Raspberry Pi Fundation, pensato apposta per le basse prestazioni dei processori Advanced RISC Machine(ARM) montati sul dispostivo. La prima versione venne rilasciata nel 2012.

2.2.2 Electron

Electron é un Framework open source rilasciato per la prima volta nel 2013, ma la prima versione stabile é uscita di recente. É disponibile sui sistemi operativi Window, MacOS e Linux ed é scritto in C++ e Javascript. Il framework permette la creazione di interfacce grafiche (GUI) per applicazioni cross platform, utilizzando teconologie gié esistenti per lo sviluppo del backend e del frontend (Javascript, NodeJS, V8 [15]). All'avvio Electron inizializza una pagina in Chromium, che renderizza una pagina web, e

un server NodeJs che si occupa di trasmettere e interagire con il frontend. Un'applicazione 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)

```
{
  "name": "magicmirror",
  "version": "2.1.1",
  "description": "The open source smart platform",
  "main": "js/electron.js"
}
```

- Un file html che contiene il template della pagina mostrata dall'applicazione
- Un file JavaScript che contiene il codice di esecuzione dell'applicazione: creazione della finestra, rendering della pagina, ecc...

2.2.3 OpenCV

OpenCV (Open Source Computer Vision Library) é una libreria software sviluppata intorno al 2000 utilizzata nell'abito della visione in tempo reale da parte di una macchia per mezzo di input digitali, ottenuti tramite telecamera o fotocamera ad esempio.
La librearia é supportata per i linguaggi C++(linguaggio in cui é scritta e dunque di cui ha l'interfaccia primaria), C, Python e Java e per diversi sistemi operativi, anche mobile.
OpenCV prede in input un'immagine o uno stream (come un video o una serie di immagini) e, utilizzando algoritmi di visioning implementati al suo interno, riconosce oggetti o specifiche forme, inoltre se ne puó aumentare l'efficienza applicando algoritmi di Machine Learning per individuare e riconoscere oggetti specifici.

2.2.4 Google Speech Recognition

Negli ultimi anni Google ha ampliato sempre di piú il suo catalogo per quanto rigarda i servizi cloud e web API. Tra questi si puó individuare anche Google Speech To Text (o Google Speech Recognition), il quale é un servizio che, ricevendo in input un file o uno stream audio, ottenuto per mezzo di 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

2.2.5 NodeJS

NodeJS é una piattaforma open source, utilizzata per progettare il backend di un server web, sfruttando il motore JavaScript V8 sviluppato da Google. NodeJS esegue delle operazioni al verificarsi di uno specifico evento, che puó essere un accesso ad una porta del server, la richiesta di una pagina o l'invio di un form data. Per gestire i pacchetti di questo linguaggio viene utilizzato NPM[16], una linea di comando che permette di scaricare ed installare librerie private o pubbliche salvate su un database. Durante lo stage é stata utlizzato il pattern architetturale Model-View-Controller(MVC), usando Express come middleware, Mustache come Sistema di Template e MySQL come Database.

Express

Express é un framework per NodeJS che permette di creare applicazioni Web e API in JavaScript. Il software viene usato per progettare il middleware di un server, che é composto di 3 entitá importanti:

- il Routing, utilizzato per determinare come il server debba rispondere ad un determinato metodo di richiesta, ricevuta sottoforma di URI, inoltrando la richiesta al controller del modello a cui fa rifermento.
- i Modelli, creati per ogni entitá-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 di nuovo se necessario.

- il Controller, che riceve la richiesta inoltrata dal Routing, nel quale sono descritte le operazioni da eseguire (in base al modello cui fa riferimento) e la risposta da ritornare.

2.2.6 JavaScript

Javascript é un linguaggio di programmazione di alto livello per oggetti ed eventi, che é stato supportato da tutti i browser per lo scripting delle pagine web. Inzialmente usato per il client-side ha subito un'evoluzione che lo ha portato ad essere utilizzato per lo sviluppo di backend e web app, permettendo di usare il linguaggio sia per programmazione proceduale sia per programmazione orientata ad oggetti. Gli oggetti in Javascript vengono creati a livello di codice collegando metodi e proprietá ad altri oggetti, anche vuoti, a tempo di esecuzione. Java

ECMAJavascript(ES) é lo standard di Javascript che negli ultimi anni ha sviluppato ed evoluto il linguaggio in diverse verisoni. In tutte il problema piú trattato é il fatto che Javascript é asincrono, ovvero non attende il completamento di alcune operazioni prima di eseguire le funzioni successive. QUesto avviene perché le chiamate di funzioni non vengono fatte direttamente, ma vengono fatte via 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 perché le funzioni e gli eventi vengono eseguiti in successione senza attendere il termine dalla precedente.

Le soluzioni adottate in ECMA Javascrit 5 (ES5), sono l'utilizzo delle 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. Il problema di questo approccio é stato il fenomeno "Callback Hell", ovvero Callback che chiamano a loro volta altre Callback creando funzioni annidate piú volte.

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 avrá 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.

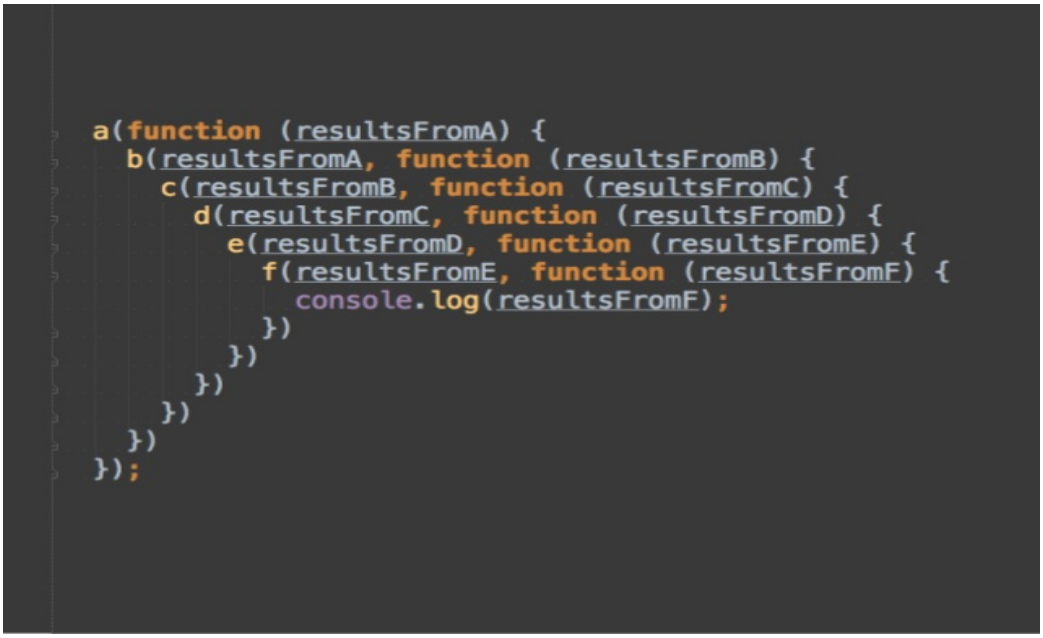

Figura 2.1: Esempio di Callback Hell

2.2.7 Python

Python é un linguaggio di programmazione open source, ad alto livello con semantica dinamica, orientato ad oggetti, usato per sviluppo di applicazioni, scripting e per collegare componenti tra di loro.

Come molti altri linguaggi supporta moduli e 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 Python dispone di virtualenv uno strumento per creare ambienti virtuali (virtual enviroments), che consistono in ambianti Python isolati per l'utilizzo di pacchetti e moduli senza doverli installare all'interno del sistema. Python é stato un linguaggio particolarmente utile per l'interfacciamento

con la telecamera, avendo una libreria apposita per gestirla.

2.2.8 GitLab

Per poter comprendere il funzionamento di GitLab occore prima conoscere come funziona Git. Quest'ultimo é stato creato da da Linus Torvalds nel 2005 per la gestione del versionamento del codice in Linux. Git é, in sostanza, un grafo diretto aciclico di oggetti, ognuno con un'identificativo univoco, ed é composto da 4 parti principali:

- i Blob, insieme di bytes che contengono i dati all'inteno dei files.
- i Trees, sono le cartelle che contengono all'intrno un lista di files con alcuni bit al quale fanno riferimento al rispettivo blob o ad un'altro Tree.
- i Tag, sono etichette con dati aggiuntivi per identificare e gestire i nodi.
- i Commit, collegano piú oggetti Tree ad uno storico nel quela sono salvati anche i nomi di tutti i commits parenti.

GitLab, dunque, é un web repository sostituivo al software Git, con implementati altri servizi tra i quali la possibilitá di creare wiki e un servizio di issue tracking, utile per dividere i lavori e controllarne lo stato di sviluppo in team composti da piú persone.

Capitolo 3

Struttra del Magic Mirror

Il Magic Mirror un progetto ideato e sviluppato da Michael Teeuw, successivamente esteso nelle sue funzionalit da una moltitudine di utenti su GitHub. Una prima versione stata scritta completamente in Python, in seguito stata creata una seconda versione nella quale si preferito l'utilizzo di Electron, che ha comportato una variazione di linguaggio, a favore di Javascript. In questo modo stato possibile implementare un'interfaccia esteticamente pi gradevole ed stato possibile implementare un sistema per far comunicare i moduli fra di loro.

3.1 Perch Magic Mirror?

L'idea dell'autore nata rifacendosi allo specchio magico dell'omonima fiaba scritta dai fratelli Grimm, La Bella Addormentata.
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.

Figura 3.1: Magic Mirror by Michael Teeuw

3.2 Avvio ed Escuzione

Il Magic Mirror viene avviato tramite riga di comando di una shell: npm start, che va a ricercare il file javascript principale indicato dal package.json. All'avvio viene eseguito il codice di Electron, che si occupa della creazione di una nuova finestra, che l'interfaccia contenente il browser usato per contenere i DOM e di stampare la relativa pagina HTML, e il codice dell'applicazione, ovvero il corpo principale dello specchio. Quest'ultima crea e avvia un server per il backend, il cui compito di caricare tutte le strtture dati dello specchio:

- i Moduli, che sono le entit che permettono di creare e configurare applicazioni da agganciare al software principale.

- i Node Helper, strutture "speciali", opzionali, che servono come supporto esterno ai moduli, per mezzo dei quali si possono interfacciare le api di servizi esterni al Magic Mirror. Ogni modulo ha il proprio Node Helper con cui pu comunicare tramite messaggi in modo simile a come comunicano i moduli tra di loro.
- un Socket, entit principale che permette lo scambio dei messaggi tra i moduli e i rispettivi Node Helper.
- un Logger, implementato per tenere i log dell'applicazione e degli evenutali errori. Usato pricipalemnte per il debugging.

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