Universita' del Piemonte Amedeo Avogado

Corso di Laurea in Informatica

MagicMirror(provvisorio)

Relatore
Dottor Marco Guazzone

Relazione della prova finale di:
Riccardo Berto
Matricola: 20003275

Anno Accademico 2016/2017

Indice

Introduzione **2**

1 Scopi e Problemi Affrontati **3**
1.1 Moduli del MagicMirror 3
1.2 Interfaccia di controllo del MagicMirror 3

2 Tecnologie Implicate **5**
2.1 Raspberry 5
2.2 Raspbian 6
2.3 Periferiche 6
2.4 Electron 6
2.5 OpenCV 7
2.6 Google Speech Recognition 7
2.7 Model-View-Design 8
2.8 NodeJS 9
2.8.1 Express 9
2.9 JavaScript 10
2.10 Python 12
2.11 GitLab 12

Introduzione

Nel corso degli ultimi decenni con il perfezionamento delle teconologie e l'evoluzione dei calcolatori, é risultato sempre piú facile, sia dal punto di vista economico che dello spazio fisico, ottenere dispositivi elettronici dall'alta portabilitá e dalla buona efficenza, di cui alcuni hanno permesso anche lo sviluppo di molti progetti fatti in casa. Questa evoluzione ha portato sempre di piú alla "fusione" con le macchine, poterle controllare senza dover necessariamente dover digitare dei caratteri o premere tasti, basti guardare buona parte degli Smartphone moderni che permettono lo sblocco dello schermo con il volto, oppure l'interazione con esso tramite sintesi vocale. Al giorno d'oggi molti software di questo ambito relazione tra uomo-macchina sono open source per permettere agli utenti di poterli evolvere e perfezionarli sempre di piú. Nella prima parte dello stage sono stati sviluppati dei moduli il Magic Mirror, un software di terze parti sviluppato in Javascript sviluppato con electron e NodeJS, in diversi linguaggi di programmazione, che permettessero l'interazione tra uomo-macchina per mezzo di periferiche esterne. Tra i moduli sviluppati vi sono il riconoscimento di presenza umana o di uno specifico volto (tramite l'utilizzo di una telecamera) e il riconoscimento vocale, ovvero l'impartizione di comandi specifici per mezzo di un microfono.

Un altro aspetto che si é evoluto soprattuto negli ultimi anni nel cmapo delle applicazioni e delle web app é l'implementazione di interfacce di gestione sempre piú intuitive e facili da comprendere. Capita spesso, infatti, che un'applicazione venga utilizzata da terzi, anche senza che questi abbiano conoscenze informatiche, se non vengono servite delle astrazioni a livello applicativo un utente, senza tali conoscenze, non puó gestire o configurare un'applicazione. Nella seconda parte dello stage, é stata sviluppata un'interfaccia web, la quale implementava un pannello di configurazione il compito é di permettere di modificare la configurazione dei moduli (anche quelli creati da terzi e inseriti successivamente) senza dover andare a modificare il file in locale rendendo piú facile aggiornare le features del dispositivo.

Capitolo 1

Scopi e Problemi Affrontati

1.1 Moduli del MagicMirror

Nello sviluppo dei moduli del MagicMirror si é dovuto affrontare il problema di far comunicare i diversi dispositivi hardware (tra cui microfono e telecamera) con il calcolatore principale, utilizzando tecnologie, software e linguaggi diversi tra loro, scelti in base alle loro caratteristiche e ai loro punti di forza. Inoltre era necessario che i moduli comunicassero tra di loro perch si potessero coordinare, o sempicemente perch un modulo, in base a degli input ricevuti, doveva controllarne un altro. Un'altro problema rilevante era la corretta scelta e gestione degli input, essendo che sia con il microfono sia con la telecamera c'erano pi soluzioni di acquisizione, che poteva essere in tempo reale (stream) o tramite una registrazione avvenuta precedentemente.
Lo scopo dei moduli é di permettere l'interazione tra il software principale e l'essere umano, e di poter controllare le funzioni della macchina senza dover utilizzare i tradizionali metodi di input (mouse e tastiera), e accedere ai diversi servizi che si possono implementare.

1.2 Interfaccia di controllo del MagicMirror

Nello sviluppo della pagina Web di controllo si é affrontato il problema di dover implementare un'altrazione che permettesse di modificare il documento di configurazione del software principale da un'interfaccia senza eliminarne parti essenziali o modificarlo in un formato sbagliato (gestito tramite validatore). Lo scambio dei messaggi tra il Backend ed il Frontend avviene in

formato JSON, lo stesso formato con cui é stipulato il file di configurazione del Magic Mirror e dei suoi moduli.
Lo scopo dell'interfaccia di permettere ad un qualsiasi utente (con i permessi) di gestire i diversi moduli implementati nel software modificandone il JSON, senza accedere fisicamente alla macchina. La pagina, inoltre, permette di configurare moduli creati anche da terzi: inserendoli semplicemente nella directory del dispositivo il Backend li cerca e li indicizza nell'interfaccia, da dove possono essere attivati.

Capitolo 2

Tecnologie Implicate

Lo sviluppo del progetto é stato svolto nell'ambiente Raspbian[1] una distribuzione Debian[2] che gira sul dispositivo Rasperry[3]. Inoltre sono state adottate diverse tecnologie, recenti e non, nel campo della creazione di un'applicazione e i relativi moduli(Electron [6], OpenCV [4], GoogleSpeechRecognition [5]) e di web server (NodeJS [14], Model-View-Controller Architecture [7], Express [8], MySQL [9], Mustache [10]), diversi linguaggi di programmazione (JavaScript [11], Python [12]) e tecnologie per visionare e condividere codici (GitLab [13]).

2.1 Raspberry

RaspberryPi é un calcolatore elettronico, montato su una singola scheda elettornica, a basso costo dal consumo ridotto e alta portabilitá. Rilasciato per la prima volta intorno al 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:

- una porta HDMI
- porta LAN
- uscita Aux
- 4 porte USB
- 40 General Purpose Input/Output(GPIO)
- Scheda di rete wirless
- Alimentazione microUSB 5V

- un bus camera serial interface(CSI)
- ingesso per microSD

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

2.2 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.

2.3 Periferiche

Nell'implementazione dei moduli del software sono stati usati diverse periferiche, tra cui una telecamera compatibile per il RaspberryPi (collegata tramite interfaccia Bus) per catturare immagini o uno stream, un microfono USB per catturare la voce e un componente Skydriver Touch Board di Piromoni collegabile tramite i 20pin del calcolatore principale, per catturare input fisici tramite il movimento delle dita sulla scheda.

2.4 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 que-

st'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...

2.5 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.

2.6 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

2.7 Model-View-Design

Il Model-View-Design (MVC) é un pattern architetturale che suddivide lo sviluppo di un'applicazione web in 3 parti:

- Model(Modello), sono oggetti che rappresentano lo stato dell'applicazione e operazioni logiche da eseguire sul primo. Di solito lo stato del modello viene estratto, manipolato per mezzo di operazioni e salvato da un database con cui comunicano, oppure passato al controller.
- Controller, é un'interfaccia che comunica tra il Model, la View e l'Utente. Il suo compito é di gestire le richieste dell'utente, il quale comunica tramite input ed interazioni, utilizzando il modello che rientra nel dominio dei dati inerente alla richiesta e selezionando una View per il rendering dell'interfaccia utente.
- View(Visualizzazione), ha il compito di far visualizzare all'utente i dati estratti tramite un'interfaccia grafica, che viene creata partendo da un modello HTML.

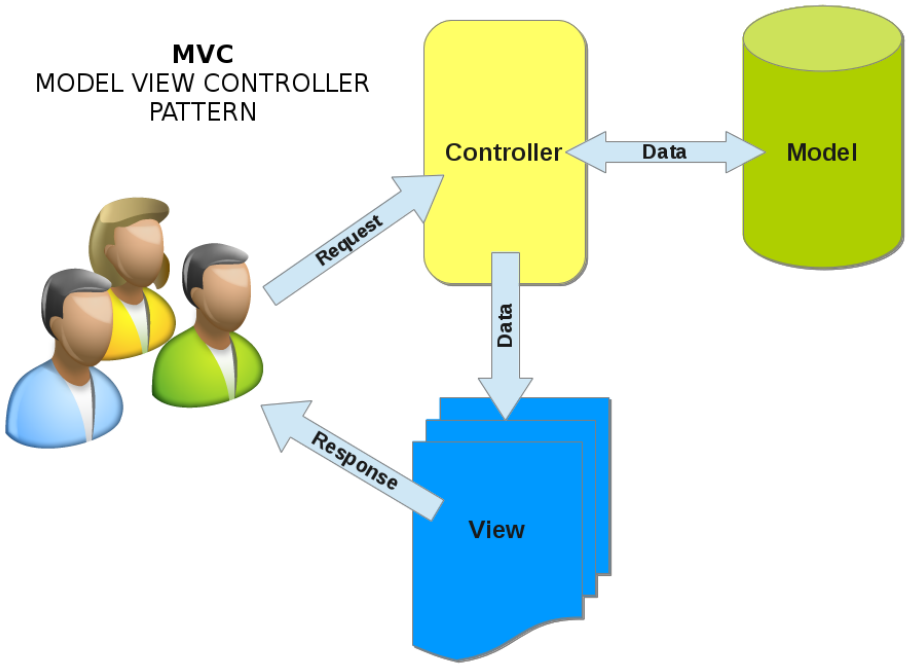

Figura 2.1: MVC Pattern

2.8 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[**?**], 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.

2.8.1 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.

2.9 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.

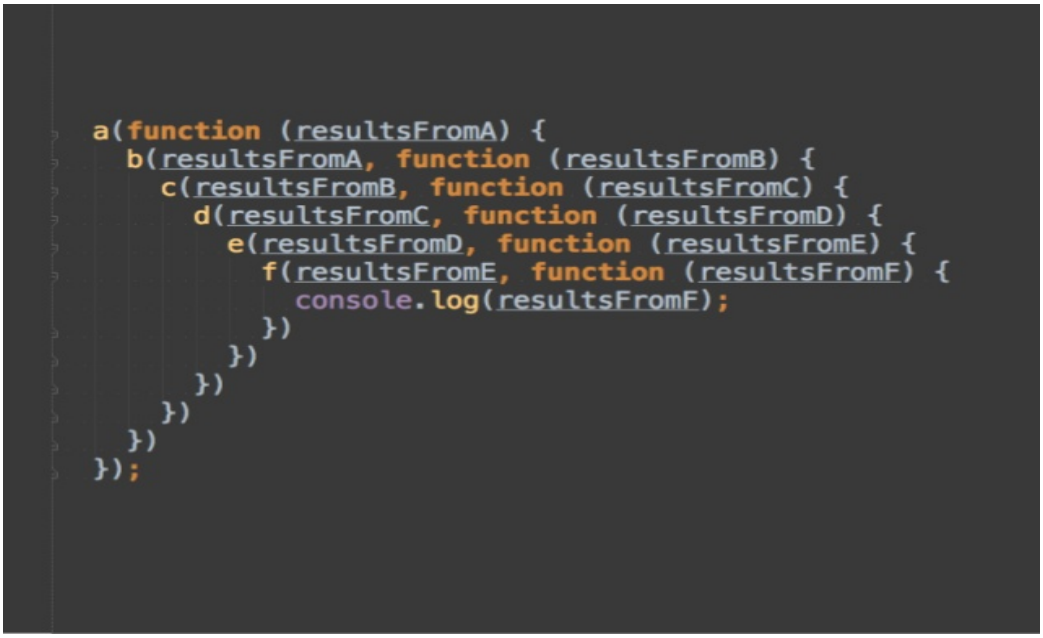

Figura 2.2: Esempio di Callback Hell

2.10 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.

2.11 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, in sostanza, 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, in sostanza, 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.

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)