Universita' del Piemonte Amedeo Avogado

Corso di Laurea in Informatica

MagicMirror

Relatore
Prof. Marco Guazzone

Relazione della prova finale di:
Riccardo Berto
Matricola: 20003275

Anno Accademico 2016/2017

Abstract

L'obbiettivo dello stage era lo sviluppo di moduli in diversi linguaggi che permettesse l'interazione tra uomo-macchina da parte di un dispositivo, e di un'interfaccia che permettesse di configurare quest'ultimo da una pagina web.

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 tramite voce per mezzo di un microfono.

Nello sviluppo di questi moduli si sono incontrate moderne tipologie di software come OpenCV e Google SpeechtoText che sono state implementate con diverse tipologie di linguaggi.

La pagina web sviluppata aveva il compito di modificare la configurazione dei moduli (anche quelli creati da terzi) senza dover andare a modificare il file in locale rendendo pi facile aggiornare le features del dispositivo.

Contents

Introduzione

prova

prova

prova

prova

prova

Chapter 1

Nozioni, linguaggio e strumenti utilizzati

Durante il percorso di stage presso Wellnet ho appreso conoscenze riguardanti le tecnologie pi recenti e promettenti nel campo dei sistemi e delle macchine virtuali (Docker [1]), utilizzando il nuovo linguaggio di programmazione Go (Golang [6]) sviluppato da Google e tecnologie di versioning del codice stabili e collaudate (Git [9]).
Nel corso dello sviluppo del tool abbiamo cercato di seguire una metodologia di lavoro chiamata Agile, che descrive una serie di principi che aiutano lo sviluppatore ed il team di sviluppo nel produrre software di qualit.

1.1 Docker

Docker [1] un software open source per la gestione dei container: aiuta ad amministrare in maniera pi semplice i Containers Linux (tramite prima i driver LXC, poi recentemente sostituiti da runC [2]). Lo scopo di Docker quello di fornire un layer di astrazione intermedio (tra la macchina e l'applicazione) che fornisce alle applicazioni un kernel linux condiviso. Insieme alla gestione del kernel condiviso Docker offre anche un sistema di distribuzione delle immagini Docker (modelli di containers) e numerosi tool per una migliore gestione dei container (dalla gestione dei cluster di containers con Docker Swarm a interfacce grafiche per l'hub di distribuzione).
Docker permette di gestire il meccanismo di isolamento di processi in un kernel condiviso. possibile isolare in modo semplice un singolo processo (quindi una singola applicazione) in un ambiente protetto e gestibile, sia dal punto di vista della sicurezza, che del consumo delle risorse.

1.1.1 Differenza tra luso di Docker e le Virtual Machines

Negli ultimi anni cresciuto l'interesse per l'utilizzo dei container linux come sostituti delle macchine virtuali. Mentre una virtual machine offre una virtualizzazione a livello hardware (full virtualization), i container linux offrono una virtualizzazione a livello del sistema operativo (OS virtualization).

Una virtualizzazione a livello hardware fornisce un isolamento maggiore (ad esempio la comunicazione tra virtual machines deve avvenire attraverso la rete) ma porta con s alcuni difetti, tra i quali grandi costo a livello hardware e una gestione complicata delle macchine[3].
Una virtualizzazione a livello di sistema operativo presenta numerosi punti di forza:

- il supporto da parte di qualsiasi kernel linux abbastanza recente (e quindi la possibilit di essere usato su dispositivi molto vari, da un server ad un sistema embedded)
- la velocit, sia in avvio che in esecuzione di applicazioni [4]
- la flessibilit e l'efficienza nell'uso delle risorse: in caso di necessit possibile replicare il container e scalare orizzontalmente l'applicazione
- un isolamento "diverso": non forte come quello delle virtual machines, ma permette ai container di comunicare anche condividendo files, socket, code FIFO... proprio perch hanno un kernel condiviso. [5]
- la distribuzione dei modelli di containers: una immagine di Docker di dimensione molto ridotta rispetto all'immagine di un sistema operativo per una virtual machine: pi facile distribuire una immagine.

Oltre a queste caratteristiche dei containers linux, Docker fornisce nativamente un protocollo per la distribuzione delle immagini (modelli di containers) pubblico e open-source.

1.2 Go

Go un linguaggio di programmazione open source sviluppato inizialmente in Google nel 2009 da Robert Griesemer (programmatore), Rob Pike (informatico, creatore insieme a Ken Thompson dello standard UTF-8) e Ken

Thompson (informatico, creatore del linguaggio B e vincitore del premio Turing nel 1983). Il linguaggio Go compilato, fortemente tipizzato, concorrente e ha un sistema di garbage collection per la gestione della memoria[6].

I programmi sono composti da packages i quali permettono una gestione efficiente delle dipendenze. L'implementazione attuale utilizza il tradizionale modello compilazione/link per generare i file eseguibili. La grammatica compatta e regolare e si focalizza sull'uso di poche keywords (25 in totale) dalla forte chiarezza semantica. [7] Il codice sorgente codificato nativamente secondo lo standard Unicode UTF-8.

Una delle caratteristiche che hanno permesso al linguaggio Go di acquisire un sempre maggior successo [8] l'uso di alcuni tool integrati nel programma di sviluppo che ne permettono una scrittura rapida e sicura. Ne sono un esempio gofmt e golint, che permettono di avere uno stile univoco nel codice e uno standard nell'uso di spazi e indentazioni, e godoc e go get per una generazione standard di documentazione e gestione delle librerie.

1.3 Git

Git[9] un software per la gestione del versionamento del codice creato da Linus Torvalds nel 2005. Semplificando, git non altro che un grafo diretto aciclico (DAG) di oggetti, ciascuno identificato da un hash univoco[10].
Gli oggetti presenti nel grafo sono principalmente tre: i blob, gli alberi e le etichette. I blob sono semplici insiemi di bytes che contengono i dati contenuti nei files, gli alberi rappresentano le directory mentre le etichette mostrano dati aggiuntivi per identificare e gestire i nodi.
Quando un nodo punta ad un altro nodo viene definita una relazione di dipendenza tra i due: il secondo non pu esistere senza il primo. Le relazioni tra i singoli nodi (blob o alberi) e le etichette tra i nodi definiscono la struttura del progetto e la sua storia (il suo stato prima e dopo ogni singolo cambiamento). Questa struttura permette di visualizzare e gestire integramente tutte le modifiche avvenute su un file.

Due caratteristiche che hanno reso git (o sue alternative per il versionamento del codice) uno strumento indispensabile per ogni sviluppatore: la semplicit di utilizzo e la gestione distribuita tramite repository online. Semplicit di utilizzo perch per iniziare ad usare git in maniera efficace, anche grazie ai tool grafici presenti in rete, non necessaria alcuna conoscenza pregressa. Un repository online (nel nostro caso abbiamo usato bitbucket per lo sviluppo

iniziale e poi github) permette di condividere il codice scritto tra i componenti del team e gestire lo stato di ogni file sorgente tra i partecipanti al repository.

1.4 Metodologie agili

Per metodologie agili si intendono una serie di principi (o modelli) per lo sviluppo software attraverso i quali il prodotto prende forma e si evolve attraverso la collaborazione tra il team di sviluppo ed il cliente. Il termine "agile" venne utilizzato per la prima volta nel 2001 nel "manifesto Agile", anche se lo sviluppo di tali metodologie iniziato gi nella met degli anni '90 in contrapposizione ai metodi a cascata tradizionali.

Manifesto agile [11]

Stiamo scoprendo modi migliori di creare software, sviluppandolo e aiutando gli altri a fare lo stesso. Grazie a questa attivit siamo arrivati a considerare importanti:

Gli individui e le interazioni pi che *i processi e gli strumenti*
Il software funzionante pi che *la documentazione esaustiva*
La collaborazione col cliente pi che *la negoziazione dei contratti*
Rispondere al cambiamento pi che *seguire un piano*

Ovvero, fermo restando il valore delle voci a *destra*,
consideriamo pi importanti le voci a **sinistra**.

Poich nel manifesto agile sono descritti solamente i principi sui quali fondata la metodologia agile, nel corso degli anni sono state date differenti definizioni di Metodologie o Metodi Agili, tra le quali una in particolare le descrive in modo sufficientemente completo:
Il metodo agile un modello di sviluppo software incentrato sulle persone, orientato alla comunicazione, flessibile e reattivo (in modo tale da adattarsi in ogni momento a cambiamenti voluti e non), veloce (incoraggiando uno sviluppo del prodotto rapido ed iterativo), snello (focalizzandosi su piccoli intervalli di tempo e sulla qualit) e proattivo (puntando sul miglioramento durante e dopo lo sviluppo)[12].
Su questo modello sono nate numerose metodologie specifiche e spesso si parla di "tecniche agili".
Le pi popolari sono: Scrum, Kanban, Adaptive Software Development, Extreme Programming, Lean Software Development, etc.
Durante lo sviluppo, vista la natura dinamica del progetto, abbiamo utilizzato principalmente un mix di queste "tecniche agili": Adaptive Software Development (ASD), Scrum e Kanban.

1.4.1 Adaptive Software Development (ASD)

Lo sviluppo Adaptive Software Development promuove il paradigma adattivo ed stato proposto per la prima volta da Jim Highsmith nel 1999 nel suo libro "Adaptive Software Development". Lo sviluppo secondo questo paradigma si focalizza sulla collaborazione tra gli individui e l'organizzazione autonoma del team.
A differenza di altre "tecniche agili" si basa su un ciclo diviso in tre fasi: una di speculazione, una di collaborazione ed una di apprendimento.
Questo ciclo permette uno sviluppo continuativo, dinamico e adattivo del progetto e favorisce una maggiore tolleranza ai cambiamenti. Il termine "speculazione" prende il posto di "pianificazione" di un normale ciclo di vita del software e si riferisce al paradosso della pianificazione dettagliata: nel caso di progetti basati su tecnologie molto recenti ed in continuo mutamento (ad esempio Docker: che durante i mesi di stage ha sviluppato una versione nativa per macchine non linux ed una gestione di pacchetti di containers sperimentale) necessario che alcune fasi della pianificazione siano continuamente messi in discussione ed aggiornate.
Per collaborazione ed apprendimento ci si riferisce allo sforzo di gestire la speculazione tra i membri del team ed alla risposta dello stesso per modificare e adattare lo sviluppo alle nuove sfide.

1.4.2 Scrum

Scrum un metodo di sviluppo agile pensato all'inizio degli anni '90 da Jeff Sutherland. Nel corso degli ultimi anni lo studio e lo sviluppo di questo metodo stato portato avanti da numerose personalit di spicco dell'ingegneria del software, tra cui Schwaber e Beedle.
Il metodo Scrum si basa sui princpi del manifesto agile e tali princpipi sono usati per guidare il ciclo di vita del prodotto, suddiviso in cinque principali attivit: requisiti, analisi, progettazione, evoluzione e consegna. Ogni attivit di lavoro segue un particolare modello temporale chiamato "sprint".

Scrum si focalizza su un insieme di metodologie dalla provata efficacia per progetti nei quali sono presenti particolari caratteristiche: tempistiche ristrette e requisiti dinamici. Come si pu vedere nell'immagine, il metodo definisce alcune strutture utili:
Backlog: una lista delle priorit nei requisiti del progetto o caratteristiche che potrebbero dare maggior valore al prodotto. Il project manager prepara il backlog ed aggiorna la lista.
Sprints: uno sprint consiste in una unit di base dello sviluppo secondo il metodo Scrum. di durata fissa (da una a quattro settimane) e durante lo sprint vengono create porzioni complete del prodotto. Durante lo sprint non possibile cambiare gli obiettivi ma necessario aspettare il termine dello stesso per eventuali modifiche.
Scrum incorpora inoltre una serie di caratteristiche che enfatizzano le priorit del progetto, la suddivisione del progetto in blocchi, la comunicazione tra il team e un feedback continuo con il cliente.
Nel corso dello stage abbiamo definito sprint di 15 giorni per tutta la durata dello sviluppo. Ad ogni termine di ogni sprint il backlog stato aggiornato con le caratteristiche da implementare e sono state discusse eventuali problematicit o scelte architetturali del tool.

1.4.3 Kanban

Kanban una metodologia per gestione della conoscenza ispirata ai principi dello sviluppo software agile, proposta da David J. Anderson nel 2007 ed ufficializzata nel libro "Kanban" dello stesso nel 2010.
Tale metodologia nasce per molto prima, negli anni '40, sulla base dei primi processi di ottimizzazione negli stabilimenti Toyota per gestire le scorte dei prodotti utilizzati nelle sue fabbriche. Per gestire al meglio gli inventari, in relazione ai materiali consumati dai vari gruppi di lavoro, venne istituita una tessera (un "kanban") che doveva essere scambiata tra gli utilizzatori delle risorse[15].

Quando un materiale finiva, un kanban veniva passato dalla fabbrica al magazzino con tutti i dati necessari (tipo di materiale, esatta quantit del materiale richiesto, etc). Il magazzino a sua volta riforniva la fabbrica e forniva un kanban al fornitore.
Mentre la tecnologia per rappresentare questo tipo di scambio si evoluta, questo processo produttivo rimasto pressoch identico.
Oggi i team che utilizzano la metodologia Agile possono utilizzare gli stessi principi sostituendo i materiali con le attivit di sviluppo ed il passaggio di consegne con il ciclo di vita del software. Questo permette al team una migliore e pi veloce pianificazione delle attivit ed una completa trasparenza nel ciclo di sviluppo.
In particolare, il metodo Kanban aiuta nella consapevolezza del team dello stato del progetto, nella visualizzazione del flusso di lavoro e nella trasparenza delle politiche di processo.
Nello sviluppo software di solito utilizzata anche una Kanban board (o lavagna) che raccoglie e suddivide le tessere Kanban in liste.

Durante tutta la durata dello stage abbiamo utilizzato una Kanban Board, con tessere Kanban virtuali, chiamata Trello.
I progetti sono rappresentati da una lavagna, che contiene differenti liste (stati del progetto), ciascuna contenenti le tessere Kanban (task del progetto). Ad ogni completamento di una attivit, la stessa viene spostata da una lista allaltra. Ogni tessera (attivit) contiene tutte le informazioni della stessa attivit (durata, descrizione, stato, se bloccante, etichette, etc.)
Le tessere progrediscono da una lista allaltra tramite un semplice drag-and-drop possibile assegnare particolari notifiche alle liste o alle tessere.

Bibliography

[1] Docker official website, `https://www.docker.com`, Ultimo accesso: 05/10/2016

[2] runC Open Container Project, `https://runc.io`, Ultimo accesso: 05/10/2016

[3] Jrme Petazzoni (Docker), `http://www.slideshare.net/jpetazzo/implementing-separation-of-concerns-with-docker-and-containers`, Ultimo accesso: 05/10/2016

[4] Wes Felter - Alexandre Ferreira - Ram Rajamony - Juan Rubio (IBM), An Updated Performance Comparison of Virtual Machines and Linux Containers, July 21 2014

[5] Aaron Grattafiori (NCC Group Whitepaper), Understanding and Hardening Linux Containers, 2016 April 20

[6] Golang website, `http://golang.org`, Ultimo accesso: 05/10/2016

[7] runC Open Container Project, `https://golang.org/ref/spec`, Ultimo accesso: 05/10/2016

[8] Tiobe Programming languages index , `http://www.tiobe.com/tiobe-index/`, Ultimo accesso: 05/10/2016

[9] GIT website, `https://git-scm.com`, Ultimo accesso: 05/10/2016

[10] Tommi Virtanen article (Computer Scientist), `http://eagain.net/articles/git-for-computer-scientists/`, Ultimo accesso: 05/10/2016

[11] Manifesto agile, `http://agilemanifesto.org`, Ultimo accesso: 05/10/2016

[12] Metodologie agili, http://www.umsl.edu/~sauterv/analysis/6840_f09_papers/Nat/Agile.html, Ultimo accesso: 05/10/2016

[13] Roger Pressman e Bruce Maxim, Software Engineering:A Practitioner's Approach, 7th edition

[14] Roger Pressman e Bruce Maxim, Software Engineering:A Practitioner's Approach, 7th

[15] Kanban, https://www.atlassian.com/agile/kanban, Ultimo accesso: 05/10/2016

[16] Trello, virtual kanban board, https://trello.com, Ultimo accesso: 05/10/2016

[17] Libreria libcompose, https://github.com/docker/libcompose, Ultimo accesso: 05/10/2016

[18] Libreria libcompose, http://yaml.org, Ultimo accesso: 05/10/2016

[19] Libreria spf13/Cobra, https://github.com/spf13/cobra, Ultimo accesso: 05/10/2016

[20] Alan A. A. Donovan Brian W. Kernighan, The Go Programming Language, 2016 October 26

[21] Go Testing Package, https://golang.org/pkg/testing, Ultimo accesso: 05/10/2016