SweeneyThreads

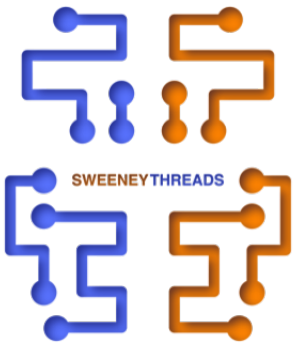
Actorbase

a NoSQL DB based on the Actor model

Glossario

Redattori:
Bonato Paolo
Bortolazzo Matteo
Maino Elia

Approvazione:
Padovan Tommaso
Verifica:
Biggeri Mattia
Tommasin Davide



Versione 1.0.0

21 gennaio 2016

Indice

Diario delle modifiche

Versione	Data	Autore	Descrizione
1.0.0	2016-01-21	*Analisti* Bonato Paolo Bortolazzo Matteo Maino Elia	Stesura scheletro documento e Introduzione

Tabella 1: Diario delle modifiche

Introduzione

1.1 Scopo del documento

In questo documento sono raccolti tutti i termini che possono risultare sconosciuti ad un lettore esterno o che possono generare ambiguità. Per ognuno di essi si riporta una breve definizione che aiuti a chiarirne il significato.

1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di un DataBase NoSQL key-value basato sul modello ad Attori$_G$ con l'obiettivo di fornire una tecnologia adatta allo sviluppo di moderne applicazioni che richiedono brevissimi tempi di risposta e che elaborano enormi quantità di dati. Lo sviluppo porterà al rilascio del software sotto licenza MIT.

1.3 Riferimenti

1.3.1 Informativi

- **Slide dell'insegnamento Ingegneria del software mod.A:**
  `http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E02.pdf`

1.3.2 Normativi

- **Norme di progetto:** *Norme di progetto v1.1.1*

A

1.1 Actor

Actor si riferisce agli attori all'interno del progetto Actorbase, non al ruolo dell'utente nell'interazione con il sistema. Un Actor può essere StoreKeeper, Ninja, StoreFinder, Manager.

1.2 Ambiente di lavoro

Quanto serve ai processi di produzione, è fatto persone, ruoli e procedure, infrastrutture.

1.3 Approccio sistematico

Sapere perché si fa una cosa, approcciare un problema conoscendo i passi da svolgere. Essere sistematici porta sia all'efficacia che all'efficienza.

1.4 Approccio disciplinato

Approccio basato su delle regole fissate in partenza da rispettare in ogni situazione al fine di rendere possibile l'organizzazione di un lavoro di gruppo e la pianificazione di tempi e costi del lavoro.

1.5 Approccio quantificabile

Ottenere un sistema in cui si riesca a dare un costo affidabile alle operazioni da svolgere. La disciplina e la sistematicità rendono possibile la quantificazione del lavoro.

1.6 Architettura SW

[...] che deve avere le seguenti qualità: * Sufficienza * Comprensibilità * Modularità * Robustezza * Flessibilità * Riusabilità * Efficienza * Affidabiltà * Disponibilità * Security * Safety * Semplicità * Incapsulazione * Coesione * Basso accoppiamento

1.7 Attore

ruolo dell'utente nell'interazione con il sitema, svolgono il caso d'uso per raggiungere l'obiettivo, sono un buon mezzo di individuazione dei casi d'uso, non includono dettagli implementativi, sui modi di interazione. Vanno identificati secondo un processo.

B

2.1 Baseline

E' l'insieme di Configuration Item ad una specifica Milestone. Rappresenta cioè un punto dello sviluppo verificato, approvato e garantito. Una volta fissata una baseline il progetto può solo avanzare, è una configurazione a cui si può tornare senza perdite in caso di fallimento.

2.2 Best practice

Prassi (modo di fare) che per esperienza e per studio abbia mostrato di garantire i migliori risultati in circostanze note e specifiche.

2.3 Brainstorming

Tecnica di analisi dei requisiti che prevede discussioni creative e paritetiche con o senza il committente. Solitamente necessita di un facilitatore, ovvero un partecipante neutrale che fa si che si rispettino le regole, ed un rapporteur, ovvero colui che tiene nota della discussione riportando solo le parti importanti producendo così le "miniature".

C

3.1 Casi d'uso

Un caso d'uso é uno specifico modo di utilizzare il sistema da parte di un attore per eseguire una certa funzionalitá del sistema stesso; Una sequenza di transizioni tramite la quale si ottiene un risultato di valore misurabile.

3.2 Configuration

E' un sistema in un suo istante; cambia a seconda di alcune svolte importanti (milestone).

3.3 Configurazione

Vedi Configuration.

3.4 Ciclo di vita del software

Gli stati che il prodotto assume dal concepimento al ritiro.

3.5 Coesione

Fa si che le attività di processo siano ben definite e correlate tra loro (e così anche i compiti al loro interno).

D

4.1 Design pattern

Soluzione progettuale ad un problema ricorrente. Definisce una funzionalità, ma lasciando dei gradi di libertà d'uso.

E

5.1 Efficacia

È determinata dal grado di conformità del prodotto rispetto alle norme vigenti e agli obiettivi prefissati. Un prodotto software finale è da considerarsi tanto più efficace quanto più si avvicina agli obiettivi fissati inizialmente. Perseguire l'efficacia equivale a garantire la qualità del prodotto.

5.2 Efficienza

È inversamente proporzionale alla quantità di risorse impiegate nell'esecuzione delle attività richieste. Perseguire l'efficienza equivale a contenere i costi e i tempi di produzione.

5.3 Elicitare

riferito a concetti o informazioni, ottenerli mediante domande o altri stimoli.

F

6.1 Fase

Durata temporale entro uno stato di ciclo di vita o una transazione tra essi.

6.2 Flipped-classroom

una flipped-classroom inverte i metodi tradizionali di insegnamento. Gli alunni guardano i concetti a casa, per conto loro, comunicando con altri alunni e con il docente. Durante la lezione disegnata sono loro a condurre l'insegnamento, esponendo concetti e informazioni elicitate a casa.

G

7.1 Gantt (diagramma di)

Diagramma per la dislocazione temporale delle attività. Per ogni attività indica durata temporale, sequenzialità/parallelismo con le altre. Permette di confrontare le stime con progressi reali.

I

8.1 Incremento

Procedere per incrementi significa aggiungere (o togliere) a un impianto base, la caratteristica fondamentale dell'incremento è l'utilizzare un solo tipo di operazione, non si torna sui propri passi come per l'iterazione. Ha caratteristiche preferibili perché permette di stabilire una data di termine.

8.2 Ingegneria (Engineering)

L'applicazione di principi scientifici e matematici a fini pratici (civili, sociali, prodotti di consumo).

8.3 Ingegneria del software

Lo studio e l'applicazione dell'ingegneria al design, allo sviluppo e al mantenimento del software. L'approccio utilizzato deve essere sistematico, disciplinato e quantificabile. Non è una branchia dell'informatica ma un interfacciarsi di diverse discipline dell'informatica, matematica, economia, ingegneria, psicologia e sociologia.

8.4 Ingegneria dei requisiti

È parte integrante dell'ingegneria di sistema e corrisponde all'insieme di attività necessarie per il trattamento sistematico dei requisiti. Tali attività si suddividono in due insiemi principali: 1 Analisi 2 Specifica di verifica e validazione

8.5 ISO 8601:2004

(Data elements and interchange formats - Information interchange - Representation of dates and times) E' lo standard di riferimento per la rappresentazione di date ed orari.

8.6 ISO/IEC 12207

ISO 12207 è uno standard dell'ISO per la gestione del Ciclo di vita del software. Si propone di diventare lo standard di riferimento definendo tutte le attività svolte nel processo di sviluppo e mantenimento del software. Lo standard ISO 12207 stabilisce un processo di ciclo di vita del software, compreso processi ed attività relative alle specifiche ed alla configurazione di un sistema. Ad ogni processo corrisponde un insieme di risultati (outcome).

8.7 Iterazione

Fare un passo indietro per fare un passo avanti, operare raffinamenti o rivisitazioni in maniera ciclica finché certe condizioni non vengono soddisfatte. L'iterazione ha caratteristiche molto pericolose perché è una operazione distruttiva (elimina quanto fatto e sovrascrive) e potenzialmente può durare all'infinito: non sa garantire un termine al processo di sviluppo, quindi è non quantificabile.

M

9.1 Manutenibilità/Manutenzione

Qualità fondamentale del sw che si presenta in diverse forme: * Correttiva: per correggere difetti eventualmente rilevati * Adattativa: per adattare il sistema alla variazione dei requisiti * Evolutiva: per aggiungere funzionalità al sistema Più risulta semplice eseguire queste operazioni più si considera il prodotto software mantenibile.

9.2 Milestone

Letteralmente pietre miliari, vengono tipicamente utilizzate nella pianificazione e gestione di progetti complessi al fine di importanti traguardi intermedi nello svolgimento del progetto stesso. Molto spesso sono rappresentate da eventi, cioè da attività con durata zero o di un giorno, e vengono evidenziate in maniera diversa dalle altre attività nell'ambito dei documenti di progetto.

9.3 Modello di ciclo di vita

Descrive come i processi si relazionano tra loro nel tempo rispetto agli stati di ciclo di vita, è la base concettuale intorno alla quale pianificare, organizzare, eseguire e controllare lo svolgimento delle attività necessarie. Esistono diversi possibili cicli di vita, non diversi per numero e significato degli stati ma diversi per le transizioni tra essi e le relative regole di attivazione.

9.4 Modularità

Fa si che i processi siano tra loro relazionati in modo chiaro e distinto.

O

10.1 Opportunità

Un'offerta relativa a un prodotto software non implica necessariamente un'opportunità (sw già esistente, progetto irrealizzabile, ...) dunque è compito del software engineer valutare la presenza di opportunità al momento di accettare o meno la commissione.

10.2 Organizzazione di processo

10.3 Pianificare (plan)

Definire attività, scadenze, responsabilità, risorse utili a raggiungere specifici obiettivi di miglioramento.

10.4 Eseguire (do)

Eseguire le attività secondo i piani.

10.5 Valutare (check)

Verificare l'esito delle azioni di miglioramento rispetto alle attese.

10.6 Agire (act)

Applicare soluzioni correttive alle carenze rilevate.

P

11.1 PERT

Acronimo di Program Evaluation and Review Technique. È una tecnica di project management (un grafico) nato per ridurre i tempi ed i costi per la progettazione. Con questa tecnica si tengono sotto controllo le attività di un progetto utilizzando una rappresentazione reticolare che aiuta ad individuare il cammino critico e a ragionare sulle scadenze di un progetto.

11.2 Problematiche essenziali

Problematiche del software relative al suo sviluppo concettuale e dunque non eliminabili dal progresso tecnologico.

11.3 Problematiche accidentali

Problematiche dello sviluppo software dovute all'inadeguatezza temporanea delle tecnologie di supporto, più legate agli strumenti che ai concetti, tali problemi possono essere risolti/eliminati grazie alle evoluzioni tecnologiche.

11.4 Prodotto software

The set of computer programs, procedures, and possibly associated documentation and data (ISO/IEC 12207) Il software vero e proprio associato eventualmente alla relativa documentazione.

11.5 Processo di ciclo di vita

Un processo di ciclo di vita specifica le attività che vanno svolte per causare transizioni nel ciclo di vita di un prodotto sw.

11.6 Processo software (Way of working)

Il quadro metodologico, normativo e strategico delle attività di progetto per organizzare al meglio le attività necessarie all'interno di vincoli dati di tempo, di risorse e di obiettivi. Un processo è un insieme di attività correlate e coese che trasformano ingressi in uscite secondo regole fissate, consumando risorse nel farlo (Glossario ISO 9000).

11.7 Progetto software

L'insieme delle attività di produzione di un software: * Pianificazione * Analisi dei requisiti * Progettazione * Realizzazione * Verifica e validazione * Manutenzione

11.8 Progetto didattico

Sequenza di revisioni di avanzamento come principale modalità di interazione basata sulla logica di relazione cliente-fornitore. Ciascuna revisione di avanzamento richiede diversi adempimenti formali e valuta il raggiungimento di uno specifico stato di avanzamento.

11.9 Programmatore

Figura professionale che svolge un'attività creativa fortemente personalizzata, scrive programmi per se stesso, da solo, sotto la propria esclusiva responsabilità tecnica.

11.10 Prototipo

Serve per provare e scegliere soluzioni a lavoro in corso, possono essere di due tipi: "usa e getta" (iterazione) o "baseline" (fornire stati di incremento). L'utilizzo di prototipi comporta in ogni caso un costo che se supera il beneficio ottenibile ne rende improduttivo l'uso.

R

12.1 Requisito

Una delle primissime attività da svolgere per quanto riguarda lo sviluppo è la definizione dei requisiti del prodotto, un sistema software sarà considerato efficace se e solo se si dimostrerà in grado di soddisfare i requisiti. Inoltre la definizione dei requisiti fornisce subito un'idea sulla realizzabilità del prodotto (l'opportunità è sostenibile?) e sui tempi di sviluppo. 1 Condizione (capability) necessaria a un utente per risolvere un problema o raggiungere un obiettivo [lato bisogno] 2 Condizione (capability) che deve essere soddisfatta o posseduta da un sistema per adempiere a un obbligo (contratto, standard, specifica, documento formale) [lato soluzione] Spesso un requisito lato utente si trasforma in molti requisiti lato soluzione.

12.2 Riuso

Molte volte nel swe risulta decisamente produttiva la tecnica del riuso a patto che lo si faccia in maniera sistematica e consapevole. Adattare componenti preesistenti alle proprie esigenze se fatto con criterio può far risparmiare al team molte risorse, aumentando l'efficienza.

12.3 Roulo

Funzione aziendale assegnata a progetto.

S

13.1 Servizio

Mezzo per fornire valore all'utente, agevolando il raggiungimento dei suoi obiettivi, sollevandolo da costi e rischi.

13.2 Software engineer

Realizza parte di un sistema complesso con la consapevolezza che potrà essere usato, completato e modificato da altri. Deve guardare e comprendere il quadro generale nel quale il sistema cui contribuisce si colloca, deve operare compromessi intelligenti e lungimiranti tra visioni e spinte contrapposte.

13.3 Standard di processo

Sono fondamentalmente di due tipi: Standard come modello di azione e Standard come modello di valutazione. I primi definiscono e impongono/propongono delle procedure da seguire, i secondi cercano di identificare una "best practice" da usare come linea di valutazione dell'operato.

13.4 Standard ISO/IEC 12207

Modello più noto e riferito, è ad alto livello e divide i processi in tre macro-aree: * Processi primari * Processi di supporto * Processi organizzativi

13.5 Stakeholder (People)

L'insieme di persone a vario titolo coinvolte nel ciclo di vita del SW con influenza sul prodotto. * Business management * Project management * Team di sviluppo * Clienti * Utenti Finali

T

14.1 Task

Compito organizzativo ottenuto dalla frantumazione di un processo. Generalmente è delle dimensioni adeguate per essere svolto da una solo persona ed è parallelizzabile.

U

15.1 UTF-8

(Unicode Transformation Format, 8 bit) è una codifica dei caratteri Unicode in sequenze di lunghezza variabile di byte, creata da Rob Pike e Ken Thompson. UTF-8 usa gruppi di byte per rappresentare i caratteri Unicode.

V

16.1 Validazione

Serve ad accertare che il prodotto realizzato corrisponda alle attese (Did I built the right system?) ed è rivolta ai prodotti finali.

16.2 Verifica

Serve ad accertare che l'esecuzione delle attività di processo non abbia introdotto errori. Si confronta ciò che si sta facendo con le regole che si devono rispettare. (Am I building the system right?)

W

17.1 WBS

Vedi Work Breakdown Structrure.

17.2 Work Breakdown Structure

Identifica l'elenco di tutte le attività di un progetto. Le WBS sono usate dal Project manager come supporto per le attività di cui è responsabile.