Norme di Progetto

SWEefty - 4 Dicembre 2017

Informazioni sul documento

Versione	0.0
Redazione	Alberto Galinaro Davide Zago Elia Montecchio Francesco Parolini Giuseppe Merlino Lisa Parma Paolo Eccher
Verifica	XXX
Approvazione	YYY
Uso	Interno/esterno
Distribuzione	ZZZ

Descrizione

Questo documento descrive le regole, gli strumenti e le convenzioni adottate dal gruppo SWEefty durante la realizzazione del progetto etctectetc.

Diario delle modifiche

Modifica	Autore	Ruolo	Data	Versione
Prima stesura dello scheletro del documento	Francesco Parolini	Project manager	2017-12-04	0.01
Scrittura della sezione "Fornitura" dei processi primari	Alberto Gallinaro	Chief Artist	2017-12-07	0.02
Scrittura della sezione "Codifica" dei processi primari	Alberto Gallinaro	Chief Artist	2017-12-07	0.03

Indice

1 Introduzione

1.1 Scopo del documento

Il seguente documento ha l'obiettivo di esplicitare le norme, le convenzioni e la strumentazione che sarà adottata dal gruppo SWEefty durante l'intero svolgimento del progetto. Con questa prospettiva deve essere visionato da tutti i membri del gruppo, i quali sono tenuti ad osservare quanto scritto per mantenere consistenza ed omogeneità in ogni aspetto del ciclo di vita del software che sarà prodotto durante il progetto.

1.2 Scopo del prodotto

Lo scopo del *prodotto*$_G$ è sviluppare dei plugin Kibana in attraverso la libreria Javascript D3.

1.3 Ambiguità

Per scongiurare ogni malinteso ed ogni ambiguità nella terminologia impiegata viene fornito il *Glossario*, il quale contiene le definizioni dei termini in corsivo con una G a pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Verbale di incontro interno:**
- **Verbale di incontro esterno:**

1.4.2 Informatici

eventuali guide latex, git, però devono essere serie....

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Lo scopo di questo *processo*$_G$ è di trattare i termini e le norme, dalle piu' triviali alle piu' importanti, che tutti i componenti del gruppo SWEefty sono tenuti a rispettare per diventare fornitori dell'azienda IKS s.r.l e dei committenti Prof. Tullio Vardanega e Prod. Riccardo Cardin

2.1.2 Aspettative

Nel corso dell'intero progetto il gruppo intende instaurare con IKS in particolare nelle figure dei referente Stefano Bertolin e Stefano Lazzaro un rapporto di costante collaborazione al fine di:

- Stimare i costi del prodotto finale
- Concordare la qualifica del prodotto
- Determinare vincoli su processi e requisiti
- Determinare aspetti fondamentali al fine di soddisfare la necessità del proponente

2.1.3 Descrizione

Il gruppo intende mantenere un constante dialogo con il proponente in modo da poter avere un riscontro sull'efficacia del lavoro svolto e sull'applicazione delle tecnologie conivolte

2.1.4 Attività

2.1.4.1 Studio di fattibilità

2.1.4.2 Piano di Progetto

2.1.4.3 Piano di Qualifica

2.1.5 Descrizione

Intendiamo mantenere un costante rapporto con il proponente in modo tale da rivecere un $feedback_G$ costante sul lavoro svolto.

2.1.6 Attività

2.1.6.1 Studio di fattibilità

Sono state organizzate diverse riunioni al fine di permettere ai membri del team di scambiarsi opinioni sui capitolati proposti. Abbiamo valutato il capitolato secondo diversi criteri:

- **Dominio tecnologico:** E' stata presa in considerazione la conoscenza attuale delle tecnologie richieste per portare a termine i vari capitolati, in base anche a esperienze passate avute con problemi simili;
- **Individuazione rischi:** sono state analizzate le varie difficoltà che potrebbero essere incontrare durante la realizzazione dei vari capitolati considerando in modo particolare la mancanza di conoscenze adeguate relative alle tecnologie necessarie per realizzare i capitolati.

2.1.6.2 Piano di Progetto

Il *Progect Manager* affiancato dagli *Amministratori* dovrà stilare un piano sa seguire per la realizzazione del progetto. Il documento dovrà rispettare e seguire i seguenti punti:

- **Analisi dei rischi:** si analizzano nel dettaglio i rischi che si potrebbero presentare durante lo svolgimento del progetto, capendo con quale probabilità potrebbero accadere e qual è il livello di gravità di ogni rischio;
- **Pianificazione:** si pianificano le attività da svolgere nel corso del progetto fornendo delle scadenze temporali precise;
- **Preventivo:** si stima, secondo la pianificazione, la quantità di lavoro necessaria per portare a termine ogni fase, in modo tale da riuscire a proporre un preventivo finale per il costo totale del progetto.

2.1.6.3 Piano di Qualifica

Ai *Verificatori* è assegnato il compito di scegliere un metodo per la $verifica_G$ e $validazione_G$ del materiale realizzato dal team. Il documento deve contenere:

- **Metodo di verifica:** vengono stabilite le procedure di controllo sulla qualità di processo e di prodotto.
- **Metriche:** è necessario stabilire delle metriche oggettive per i documenti e i processi software;
- **Gestione della revisione:** si devono stabilire delle modalità per comunicare le anomalie e le procedure per controllare la qualità di processo;

- **Pianificazione collaudo:** è necessario definire dettagliatamente le motodologie di collaudo del progetto;
- **Resoconto attività verifica:** alla fine di ogni attività svolta si devono riportare le metriche relative e un resoconto sulla verifica effettuata.

2.2 Sviluppo

2.2.1 Scopo

Questa sezione affronta le attività e i compiti svolti dal gruppo al fine di produrre il software richiesto dall'azienda IKS Srl.

2.2.2 Aspettative

Al fine di implementare correttamente questa attività vi sono le seguenti aspettative:

- Realizzare un prodotto software conforme alle richieste del proponente
- Realizzare un prodotto software che soddisfi i test di verifica
- Realizzare un prodotto software che soddisfi i test di validazione
- Fissare degli obiettivi di $sviluppo_G$
- Fissare eventuali vincoli tecnologici
- Fissare i vincoli di design

2.2.3 Descrizione

Il processo di sviluppo si svolge in accordo con lo standard ISO/IEC 12207.Pertanto si compone delle seguenti attività:

- Analisi dei requisiti
- Progettazione
- Codifica

2.2.4 Attività

2.2.4.1 Analisi dei requisiti

2.2.4.1.1 Scopo Lo scopo di questa analisi è quella di elencare nel modo più preciso possibile, tutti i requisiti del progetto. I requisiti sono estrapolati da diverse fonti:

- Specifica del capitolato
- Incontri tenuti direttamente con il proponente
- Casi d'uso

Dopo aver svolto questo processo viene stilato in modo formale un documento di analisi chiamato *Analisi dei Requisiti v1.0.0*. Quest'ultimo è steso dagli *Analisti* e contiene una lista dei requisiti e dei casi d'uso. In questo modo sarà possibile effettuare e definire dei test di superamento del software.

2.2.4.1.2 Aspettative L'obiettivo di questa attività è quello di definire in modo formale e dettagliato tutti i requisiti richiesti dal proponente.

2.2.4.1.3 Descrizione Nel documento inerente a quest'analisi sono specificati i requisiti analizzati. Il metodo utilizzato per l'analisi dei requisiti è quella dei casi d'uso.

2.2.4.1.4 Casi d'uso

2.2.4.1.5 Codice identificativo

2.2.4.1.6 Requisiti

2.2.4.1.7 Codice identificativo

2.2.4.1.8 UML I diagrammi UML devono essere realizzati usando la versione del linguaggio *v2.0*

2.2.4.2 Progettazione

2.2.4.2.1 Scopo

2.2.4.2.2 Aspettative

2.2.4.2.3 Descrizione

2.2.4.2.4 Specifica Tecnica

2.2.4.2.5 Definizione di Prodotto

2.2.4.3 Codifica

2.2.4.3.1 Scopo Questa attività ha come scopo l'effettiva implementazione del prodotto software richiesto. In questa fase dunque si concretizzano attraverso la codifica le funzionalità previste dai requisiti concordati.

2.2.4.3.2 Aspettative Obiettivo dell'attività è la creazione di un prodotto software confome alle richieste del proponente

2.2.4.3.3 Descrizione L'attività deve rispettare quando imposto dai documenti *Definizione di prodotto* e *Piano di Qualifica*

2.2.4.3.4 Stile di codifica Al fine di garantire uniformità all'intera $codebase_G$, ciascun membro del gruppo è obbligato a rispettare le seguenti norme:

- Formattazione: è richiesto l'uso di uno spazio (" ") ove possibile per rendere il codice di facile comprensione. Di seguito alcuni esempi di buona e cattiva formattazione.

```
int a=3; // BAD
int a = 3; // GOOD
int a = 3,b = 5; // BAD
int a = 3, b = 4; // GOOD

getFoo(a,b,c,d) // BAD
getFoo(a, b, c, d) // GOOD

if(a==5){ // BAD
if (a == 5) { // GOOD
```

- Indentazione: sono richiesti 4 spazi (si consiglia di impostare adeguatamente il proprio IDE_G), inoltre non sono permessi spazi bianchi o tabulazioni a fine riga.
- Lunghezza linee di codice: una riga di codice non deve superare i 100 caratteri, in caso contrario spezzare la riga di codice andando a capo
- Nomi: i nomi delle funzioni e delle variabili devono essere significativi e devono cominciare con la lettera minuscola e seguire la notazione $camelCase_G$. I nomi delle classi devono avere la prima lettera maiuscola mentre i nomi delle costanti devono essere scritti interamente in maiuscolo.

```
var foo; // BAD
var profiledata // BAD
var profileData // GOOD
```

- Lingua: i nomi delle variabili i metodi le funzioni e i commenti vanno scritti in inglese.
- Lunghezza metodi e funzioni: i corpo dei metodi e delle funzioni non deve superare le 40 linee e i due gradi di indentazione. Superare tali limiti è chiaro segno che la funzione / metodo necessita di essere spezzata. A volte però può risultare preferibile mantenere tutta la logica su un solo metodo per facilitarne la comprensibilità.
- Strutture di controllo (if, switch, for, etc...): le parentesi graffe per i costrutti che le prevedono devono essere in linea con il costrutto stesso. E' prevista l'omissione delle graffe soltanto quando il corpo della struttura è di una sola riga.

```
if (...) { // GOOD
    // CODICE
} else {
    // CODICE
}


if (...) // BAD
{
    // CODE
}
else
{
    // CODE
}
```

- Commenti: i commenti devono essere esaustivi e non prolissi in modo da aiutare il lettore a comprendere al meglio ciò che si sta leggendo.

E' inoltre consentito l'utilizzo di particolari tipi di commento con lo scopo di aiutare la stesura del codice. Vi sono:

- TODO: soluzione temporanee, memento di ogni tipo o zone che necessitano di un'implementazione.

```
// TODO: this solution stinks
// TODO: Write that query
```

- HOW: per segnalare che non si ha capito l'implementazione o il funzionamento di una porzione di codice e che si necessita di uno studio più approfondito

```
// HOW: need to study this API call, I m puzzled
```

- FIX: quando una particolare implementazione necessita di essere riparata o sistemata.

```
// FIX: this implementation doesn t work with IE
```

Questi commenti vanno scritti tutti in maiuscolo e seguiti da due punti (:), questa sintassi favorisce la loro individuazione da particolari $tool_G$ che ne permetto una consultazione agevolata.

2.2.4.3.5 Intestazione Ogni $file_G$ contenente codice sorgente deve avere la seguente intestazione:

```
/*

* File : nome file
* Version : versione file
* Type : tipo file
* Date : data di creazione
* Author : nome autore / i
* E - mail : email autore / i
*
* License : tipo licenza
*
* Avvertenze : lista avvertenze e limitazioni
*
* Descrizione: breve descrizione del contenuto del file
*
* Registro modifiche :
* Autore || Data || breve descrizione modifiche
*
*/
```

2.2.4.3.6 Versionamento Il versionamento dei file segue la filosofia "Change Significance". La versione di un file è espressa secondo la notazione X.Y.Z, ogni volta che si modifica un file va modificata anche la versione corrente seguendo questa logica:

- X: major release, l'incremento di questo numero significa che sono state aggiunte funzionalità importanti
- Y: minor improvement, bugfix, va incrementato in caso di implementazione di funzionalità più piccole
- Z: incrementato in caso di $typos_G$, errori nei commenti o piccoli bug

Quando un file raggiunge la versione 1.0.0 significa che sono state implementate le funzionalità obbligatorie e quindi si può cominciare a testare il codice.

2.2.4.3.7 Ricorsione E' caldamente sconsigliato l'utilizzo di ricorsione, esistono però dei casi in cui la soluzione ricorsiva rende il problema triviale, in tal caso è richiesta la verifica di correttezza e di terminazione della ricorsione.

2.2.5 Strumenti

2.2.5.1 Trender

2.2.5.2 Astah

2.2.5.3 IntelliJ IDEA

3 Processi Organizzativi

3.1 Gestione

3.1.1 Scopo

3.1.2 Aspettative

3.1.3 Descrizione

3.1.4 Ruoli di progetto

3.1.4.1 Amministratore di Progetto

3.1.4.2 Responsabile di Progetto

3.1.4.3 Analista

3.1.4.4 Progettista

3.1.4.5 Verificatore

3.1.4.6 Programmatore

3.1.5 Procedure

3.1.5.1 Gestione delle comunicazioni

3.1.5.1.1 Comunicazioni interne

3.1.5.1.2 Comunicazioni esterne

3.1.5.2 Gestione degli incontri

3.1.5.2.1 Incontri interni

3.1.5.2.2 Incontri esterni

3.1.5.3 Gestione degli strumenti di coordinamento

3.1.5.3.1 Ticketing

3.1.5.4 Gestione degli strumenti di versionamento

3.1.5.4.1 Repository

3.1.5.4.2 Struttura del repository

3.1.5.4.3 Tipi di file e .gitignore

3.1.5.4.4 Norme sui commit

3.1.5.5 Gestione dei rischi

3.1.6 Strumenti

3.1.6.1 Sistema Operativo

3.1.6.2 Slack

3.1.6.3 Telegram

3.1.6.4 Wrike

3.1.6.5 Git

3.1.6.6 Github

4 Processi di Supporto

4.1 Documentazione

4.1.1 Scopo

4.1.2 Aspettative

4.1.3 Descrizione

4.1.4 Procedure

4.1.4.1 Approvazione dei documenti

4.1.5 Template

4.1.6 Struttura dei documenti

4.1.6.1 Prima pagina

4.1.6.2 Registro delle modifiche

4.1.6.3 Indice

4.1.6.4 Contenuto principale

4.1.6.5 Note a piè di pagina

4.1.7 Versionamento

4.1.8 Norme tipografiche

4.1.8.1 Stile del testo

4.1.8.2 Elenchi puntati

4.1.8.3 Formati comuni

4.1.8.4 Sigle

4.1.9 Elementi grafici

4.1.9.1 Tabelle

4.1.9.2 Immagini

4.1.10 Classificazione dei documenti

4.1.10.1 Documenti informali

4.1.10.2 Documenti formali

4.1.10.3 Verbali

4.1.11 Strumenti

4.1.11.1 LaTeX

4.1.11.2 TexStudio

4.1.11.3 Lucidchart

4.2 Verifica

4.2.1 Scopo

4.2.2 Aspettative

4.2.3 Descrizione

4.2.4 Attività

4.2.4.1 Analisi

4.2.4.1.1 Analisi Statica

4.2.4.1.2 Analisi dinamica

4.2.4.2 Test

4.2.4.2.1 Test di unità

4.2.4.2.2 Test di integrazione

4.2.4.2.3 Test di sistema

4.2.4.2.4 Test di regressione

4.2.4.2.5 Test di accettazione

4.2.5 Strumenti

4.2.5.1 Verifica ortografica

4.2.5.2 Validazione W3C

4.2.5.3 Analisi statica

4.2.5.4 Analisi dinamica

4.2.5.5 Metriche