Agents of S.W.E.

A SOFTWARE COMPANY

Agents of S.W.E - Progetto "Plugin Grafana"

Norme di Progetto

Versione	0.0.4
Approvazione	?
Redazione	Luca Violato Bogdan Stanciu Marco Favaro Marco Chilese
Verifica	? ?
Stato	Work in Progress
Uso	Interno
Destinato a	Agents of S.W.E. Prof. Tullio Vardanega Prof. Riccardo Cardin

agentsofswe@gmail.com

Indice

1 Introduzione

1.1 Scopo del Documento

1.2 Ambiguità e Glossario

1.3 Riferimenti

2 Processi Primari

2.1 Fornitura

In questa sezione del documento vengono trattate le norme che il team `Agents of S.W.E.` decide e si impegna a rispettare, con lo scopo di proporsi e divenire fornitori nei confronti dell'azienda proponente, *Zucchetti S.p.A.*, e dei committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin nell'ambito della progettazione, sviluppo e consegna del prodotto "*Plugin per Grafana*".

2.2 Studio di fattibilità

2.3 Sviluppo

2.3.1 Analisi dei requisiti

2.3.1.1 Classificazione dei requisiti

2.3.1.2 Classificazione dei casi d'uso

2.3.2 Progettazione

2.3.3 Codifica

Di seguito vengono definite delle norme che devono essere adottate dai Programmatori per garantire una buona leggibilità e manutenibilità del codice. Le prime norme che seguiranno sono le più generali, da adottarsi per ogni linguaggio di programmazione adottato all'interno del progetto, in seguito quelle più specifiche per i linguaggi JavaScript$_G$, HTML$_G$ e CSS$_G$.
Ogni norma è caratterizzata da un paragrafo di appartenenza, da un titolo, una breve descrizione, e se il caso lo richiede, un esempio.
Il rispetto delle seguenti norme è fondamentale per garantire uno stile di codifica uniforme all'interno del progetto, oltre che per massimizzare la leggibilità e agevolare la manutenzione, la verifica$_G$ e la validazione$_G$.

2.3.3.1 Convenzioni per i nomi:

- I Programmatori devono adottare come notazione per la definizione di cartelle, file, metodi, funzioni e variabili il CamelCase$_G$.
  Di seguito un esempio di corretta nomenclatura:

```
1 INSERIRE ESEMPIO
```

- Tutti i nomi devono essere **unici** ed **autoesplicativi**, ciò per evitare ambiguità e limitare la complessità .

2.3.3.2 Convenzioni per la documentazione:

- Tutti i nomi ed i commenti al codice vanno scritti in **inglese**;
- Nel codice è possibile utilizzare un commento con denominazione **TODO** in cui si vanno ad indicare compiti da svolgere;
- L'intestazione di ogni file deve essere la seguente:

```
1  /**
2  * File: nomeFile
3  * Type: fileType
4  * Creation date: gg/mm/yyyy
5  * Author: Name Surname
6  * Author e-mail: email@example.com
7  * Version: versionNumber
8  *
9  * Changelog:
10 * #entry || Author || Date || Description
11 */
```

- La versione del file nell'intestazione, deve rispettare la seguente formulazione: $Y.K$, dove Y rappresenta la versione principale, K la versione parziale della relativa versione principale.
  I numeri di versione del tipo $Y.0$, dalla 1.0, vengono considerate versioni stabili, e quindi versioni da testare per saggiarne la qualità .

2.3.3.3 ECMAScript 6: Seguendo le indicazioni presenti nella documentazione[1] dell'azienda fornitrice di *Grafana*, la piattaforma per cui si intende sviluppare il plugin, il team ha deciso di adottare come linguaggio di programmazione principale ECMAScript 6[2].
ECMAScript 6 viene stardardizzato da **ECMA**$_{\text{G}}$ [3] nel giugno 2015 con la sigla **ECMA-262**[4].
Come stile di codifica si adottano le line guida proposte da **Airbnb JavaScript**

[1] `http://docs.grafana.org/plugins/developing/development/`
[2] Linguaggio divenuto standard ISO: ISO/IEC 16262:2011, e relativo aggiornamento ISO/IEC 22275:2018.
[3] `http://www.ecma-international.org/`
[4] `https://www.ecma-international.org/ecma-262/6.0/`

Style Guide[5]. Per la verifica dell'adesione a tali norme, i Programmatori devono utilizzare, come suggerito dalla documentazione proposta da *Grafana*, **ESLint**$_G$ [6]. In particolare i Programmatori devono rispettare 5 linee guida proposte dalla documentazione ufficiale di *Grafana*:

1. Se una variabile non viene riutilizzata, deve essere dichiarata come `const`;
2. Utilizzare preferibilmente, per la definizione di variabili, la keyword `let`, anziché `var`;
3. Utilizzare il marcatore freccia (`=>`), in quanto non oscura il `this`:

```
testDatasource() {
  return this.getServerStatus()
  .then(status => {
    return this.doSomething(status);
  })
}
```

Invece che:

```
testDatasource() {
  var self = this;
  return this.getServerStatus()
    .then(function(status) {
  return self.doSomething(status);
  })
}
```

4. Utilizzare l'oggetto *Promise*:

```
metricFindQuery(query) {
  if (!query) {
    return Promise.resolve([]);
  }
}
```

Invece che:

```
metricFindQuery(query) {
  if (!query) {
    return this.$q.when([]);
  }
}
```

[5] https://github.com/airbnb/javascript
[6] https://eslint.org/

5. Se si utilizza *Lodash* è meglio essere conseguenti, e preferire in ogni caso le funzioni per gli array native di ES6.

Verranno esaminate di seguito le norme in merito allo stile di codifica che i Programmatori dovranno adottare.

Identazione

Norma 1 L'identazione è da eseguirsi con tabulazione la cui larghezza sia impostata a due (2) spazi per ogni livello.
Di seguito un esempio da ritenersi corretto:

```
1 function() {
2 ..let x = 2;
3 ..if (x > 0)
4 ....return true;
5 ..else
6 ....return false;
7 }
```

Qualsiasi altro tipo di indentazione è da ritenersi scorretta.

Norma 2 Dopo la graffa principale va inserito uno (1) spazio. Nel seguente modo:

```
1 function() { ... }
```

Norma 3 Dopo la keyword di un dato statement (`if, while`, etc.) va inserito uno (1) spazio. Per un esempio corretto si veda la norma successiva.
Norma 4 Prima dell'apertura della graffa negli statement di controllo va inserito uno (1) spazio. Nel seguente modo:

```
1 function() {
2   if (condition) {
3     ...
4   }
5   while (condition) {
6     ...
7   }
8 }
```

Norma 5 Negli statement di controllo (`if, while`, etc) le condizioni concatenate, o annidate, mediante operatori logici, che diventano eccessivamente lunghe NON vanno espresse in un'unica linea, bensì spezzate in più righe. Nel seguente modo:

```
function() {
  if (condition && condition) {
    ...
  }

  if (
   veryLongCondition
   && longCondition
   && condition
    ) {
    doSomething();
  }
}
```

Norma 6 Dopo blocchi, o prima di un nuovo statement va lasciata una riga vuota. Nel seguente modo:

```
function1() {
  if (condition) {
    doSomething():
  }

  return toReturn;
}

function2(){
  ...
}
```

Norma 7 I blocchi di codice multi-riga devono essere contenuti all'interno di graffe. Blocchi costituiti da una singola riga non è necessario che sia contenuti tra graffe: nel caso non vengano utilizzate, la definizione deve essere *inline*, sulla stessa riga. Nel seguente modo:

```
if (condition) return true;

if (condtion) {
  return true;
}
```

Commenti al codice

Il codice va commentato nel seguente modo:

- "//" se il commento occupa una sola riga;
- "/** ... */ " se il commento occupa più righe.

Nel seguente modo:

```
1 // single line comment
2 if (condition) return true;
3
4 /**
5 * multi line comment, line 1
6 * multi line comment, line 2
7 */
8 if (condtion) {
9   return true;
10 }
```

Variabili

Norma 1 Fare riferimento alle norme 1 e 2, delle 5 linee guida enunciate sopra.

Norma 2 Non utilizzare dichiarazioni multiple di variabili, dichiarare una variabile per riga.
Nel seguente modo:

```
1 // OK
2 var x = 1;
3 var y = 0;
4
5 // NO
6 var x = 1, y = 0;
```

Nomi

Norma 1 Oltre a quanto enunciato nel secondo punto del paragrafo §2.2.3.1, tutti i nomi di funzioni o variabili composti da una singola lettera, o che indichino temporaneità della variabile sono *vietati*: ogni nome deve essere significativo.

Norma 2

1. I nomi delle variabili, funzioni ed istanze devono utilizzare il `CamelCase`;
2. I nomi delle classi deve lo stile `capWords`.

Nel seguente modo:

```
1  // OK
2  var thisIsAVariable;
3
4  function thisIsAFunction() { ... }
5
6  class ThisIsAClass() {
7    ...
8  }
9
10 // NO
11 var Variable;
12
13 function Function() { ... }
14
15 class myClass() {
16   ...
17 }
```

2.3.3.4 HTML

3 Processi di supporto

3.1 Versionamento

3.1.1 Controllo di versione

3.1.1.1 Struttura del repository

3.1.1.2 Processo di implementazione

3.1.1.3 Ciclo di vita dei branch

3.1.1.4 Rilascio di versione

3.1.2 Configurazione versionamento

3.1.2.1 Remoto

3.1.2.2 Locale

3.2 Gestione di progetto

3.2.1 Configurazione strumenti di organizzazione

3.2.1.1 Inizializzazione

3.2.1.2 Aggiunta tasks

3.2.1.3 Aggiunta milestones

3.2.2 Ciclo di vita delle tasks

3.2.2.1 Apertura

3.2.2.2 Completamento

3.2.2.3 Richiesta di revisione

3.2.2.4 Completamento

3.3 Verifica

3.4 Validazione

4 Processi Organizzativi

4.1 Processi di Coordinamento

4.1.1 Gestione Comunicazioni

4.1.1.1 Comunicazioni Interne

4.1.1.2 Comunicazioni Esterne

4.1.2 Gestione Riunioni

4.1.2.1 Riunioni Interne

4.1.2.2 Riunioni Esterne

4.1.2.3 Verbale di Riunione

4.2 Ruoli di Progetto

Nell'ottica di un lavoro ben organizzato e collaborativo tra i membri del gruppo, ad ogni componente, in ogni momento, è attribuito un ruolo per un periodo di tempo limitato.
Questi ruoli, che corrispondono ad una figura aziendale ben precisa, sono:

- **Responsabile di Progetto**;
- **Amministratore di Progetto**;
- **Analista**;
- **Progettista**;
- **Programmatore**;
- **Verificatore**.

4.2.1 Responsabile di Progetto

Detto anche *"Project Manager"*, è il rappresentate del progetto, agli occhi sia del committente che del fornitore. Egli risulta dunque essere, in primo luogo, il responsabile ultimo dei risultati del proprio gruppo. Figura di grande responsabilità, partecipa al progetto per tutta la sua durata, ha il compito di prendere decisioni e

approvare scelte collettive.
Nello specifico egli ha la responsabilità di:

- Coordinare le attività del gruppo, attraverso la gestione delle risorse umane;
- Approvare i documenti redatti, e verificati, dai membri del gruppo;
- Elaborare piani e scadenze, monitorando i progressi nell'avanzamento del progetto;
- Redigere l'organigramma del gruppo e il *Piano di Progetto*.

4.2.2 Amministratore di Progetto

L'amministratore è la figura chiave per quanto concerne la produttività. Egli ha infatti come primaria responsabilità il garantire l'efficienza del gruppo, fornendo strumenti utili e occupandosi dell'operatività delle risorse. Ha dunque il compito di gestire l'ambiente lavorativo.
Tra le sue responsbilità specifiche figurano:

- Redigere documenti che normano l'attività lavorativa, e la loro verifica$_G$;
- Redigere le *Norme di Progetto*;
- Scegliere ed amministrare gli strumenti di versionamento$_G$;
- Ricercare strumenti che possano agevolare il lavoro del gruppo;
- Attuare piani e procedure di gestione della qualità$_G$.

4.2.3 Analista

L'analista deve essere dotato di un'ottima conoscenza riguardo al dominio del problema. Egli ha infatti il compito di analizzare tale dominio e comprenderlo appieno, affinchè possa avvenire una corretta progettazione$_G$.
Ha il compito di:

- Comprendere al meglio il problema, per poi poterlo esporre in modo chiaro attraverso specifici requisiti$_G$;
- Redarre lo *Studio di Fattibilità* e l'*Analisi dei Requisiti*$_G$.

4.2.4 Progettista

Il progettista è responsabile delle attività di progettazione$_G$ attraverso la gestione di aspetti tecnici del progetto.
Più nello specifico si occupa di:

- Definire l'Architettura$_G$ del prodotto, applicando quanto più possibile norme di *best practice*$_G$, prestando attenzione alla manutenibilità del prodotto;
- Suddividere il problema, e di conseguenza il sistema, in parti di complessità trattabile.

4.2.5 Programmatore

Il programmatore si occupa delle attività di codifica, le quali portano alla realizzazione effettiva del prodotto.
Egli ha dunque il compito di:

- Implementare l'architettura definita dal *Progettista*, prestando attenzione a scrivere codice coerente con ciò che è stato stabilito nelle norme di qualifica;
- Produrre codice documentato e manutenibile;
- Realizzare le componenti necessarie per la verifica$_G$ e la validazione$_G$ del codice;
- Redarre il *Manuale Utente*.

4.2.6 Verificatore

Il verificatore, figura presente per l'intera durata del progetto, è responsabile delle attività di verifica$_G$.
Nello specifico egli:

- Verifica l'applicazione ed il rispetto delle *Norme di Progetto*;
- Segnala al *Responsabile di Progetto* l'emergere di eventuali discordanze tra quanto presentato nel *Piano di Progetto* e quanto effettivamente realizzato;
- Ha il compito di redarre il *Piano di Qualifica*.

4.2.7 Rotazione dei Ruoli

Come da istruzioni ogni membro del gruppo dovrà ricoprire, per un periodo di tempo limitato, ciascun ruolo, nel rispetto delle seguenti regole:

- Ciascun membro dovrà svolgere **esclusivamente** le attività proprie del ruolo a lui assegnato;
- Al fine di evitare conflitti di interesse nessun membro potrà ricoprire un ruolo che preveda la verifica$_G$ di quanto da lui svolto, nell'immediato passato;
- Vista l'ampia differenza di compiti e mansioni tra i vari ruoli, e al fine di valorizzare l'attività collaborativa all'interno del gruppo, ogni componente che abbia ricoperto in precedenza un ruolo ora destinato a qualcun altro dovrà fornire supporto al compagno in caso di necessità, fornendogli consigli e, se possibile, affiancandolo in situazioni critiche.

4.3 Procedure

4.3.1 Gestione degli Strumenti di Versionamento

4.3.1.1 Struttura Repository

4.3.1.2 Tipi di files

4.3.1.3 Norme delle Commit

4.3.2 Gestione degli Strumenti di Coordinamento

4.3.2.1 Tasks

4.3.2.2 Tickets

4.4 Strumenti

4.4.1 Sistema Operativo

4.4.2 Versionamento e Issue Tracking

4.4.2.1 Git

4.4.2.2 Github

4.4.3 Comunicazione

4.4.3.1 Telegram

4.4.3.2 Slack

4.4.4 Diagrammi di Gantt

4.4.5 Diagrammi UML

4.5 Formazione del Gruppo

La formazione dei componenti del gruppo `Agents of S.W.E.` è da considerarsi individuale. Ogni membro del team è infatti tenuto a documentarsi autonomamente riguardo le tecnologie coinvolte nello sviluppo del progetto. Tuttavia, nell'ottica di un ambiente di lavoro sano e collaborativo, nel caso in cui fosse necessario, è compito degli *Amministratori* mettere a disposizione di chi ne avesse bisogno risorse basilari, al fine di agevolare la formazione dei restanti componenti del gruppo.

5 Changelog

Versione	Data	Autore	Ruolo	Descrizione
0.0.1	2018-11-23	Luca Violato	Amministratore	Strutturazione del Documento
0.0.2	2018-11-23	Marco Chilese	Verificatore	Prima stesura §2
0.0.3	2018-12-01	Luca Violato	Amministratore	Strutturazione §4, stesura §4.2 e §4.5
0.0.4	2018-12-02	Marco Chilese	Verificatore	Ampliamento §2

Tabella 1: Changelog del documento