QUIZZIPEDIA

team404swe@gmail.com

Norme di Progetto 3.0

Informazioni sul documento	
Nome Documento	Norme di Progetto
Versione	3.0
Uso	Interno
Data Creazione	21 dicembre 2015
Data Ultima Modifica	14 giugno 2016
Redazione	Martin Vadice Mbouenda
Verifica	Andrea Multineddu
Approvazione	Davide Bortot
Committente	Zucchetti SPA
Lista di distribuzione	Prof. Vardanega Tullio
	TEAM404

Registro delle modifiche

Versione	Autore	Data	Descrizione
2.	N.COGNOME (RUOLO)	GG/MM/2016	DESCRIZIONE
2.	N.COGNOME (RUOLO)	GG/MM/2016	DESCRIZIONE
2.	N.COGNOME (RUOLO)	GG/MM/2016	DESCRIZIONE
2.0.3	N.COGNOME (RUOLO)	28/05/2016	Ristrutturazione secondo ISO 12207: processo di gestione, processo di allestimento. inizio miglioramento attività di progettazione
2.0.2	M. Mbouenda (RUOLO)	27/05/2016	Ristrutturazione secondo ISO 12207: Processo di sviluppo, sistemazione attività di analisi dei requisiti
2.0.1	M. Mbouenda (RUOLO)	26/05/2016	Ristrutturazione secondo ISO 12207: Processi primari
2.0	A. Beccaro (Verificatore)	11/05/2016	Approvazione documento.
1.1	A. Multineddu (Amministratore)	10/05/2016	Verifica del documento.
1.0.4	M. Mbouenda (Amministratore)	09/05/2016	Aggiunta sottosezione §2.4.3.
1.0.3	M. Mbouenda (Amministratore)	07/05/2016	Modifica sottosezione Versionamento e Git Modo d'uso.
1.0.2	M. Mbouenda (Amministratore)	05/05/2016	Modifica sottosezione Analisi dei requisiti.
1.0.1	M. Mbouenda (Amministratore)	04/05/2016	Modifica sottosezione Ticketing.
1.0	Davide Bortot (Responsabile)	16/03/2016	Approvazione del documento.
0.2	A. Multineddu (Verificatore)	15/03/2016	Verifica completa del documento.
0.1.7	D. Bortot (Amministratore)	11/03/2016	Modifica sottosezione "Componenti grafiche".
0.1.6	M. Mbouenda (Amministratore)	11/03/2016	Modifica sottosezione "Test".
0.1.5	M. Mbouenda (Amministratore)	10/03/2016	Modifica sezione "Processo di Sviluppo".
0.1.4	M. Mbouenda (Amministratore)	18/01/2016	Stesura sottosezione "Produzione dei Documenti".
0.1.3	M. Mbouenda (Amministratore)	14/01/2016	Stesura sottosezione "Strumenti di Progetto".
0.1.2	M. Mbouenda (Amministratore)	12/01/2016	Modifica della sottosezione "Processo di verifica".

0.1.1	M. Mbouenda (Amministratore)	10/01/2016	Modifica sottosezione "Comunicazione".
0.1.0	A. Multineddu (Verificatore)	10/01/2016	Prima verifica. Segnalati errori nella sottosezione "Comunicazione".
0.0.5	M. Mbouenda (Amministratore)	08/01/2016	Inizio stesura della sezione "Processo Organizzativo".
0.0.4	M. Mbouenda (Amministratore)	05/01/2016	Inizio stesura della sezione "Processo di Supporto"
0.0.3	M. Mbouenda (Amministratore)	04/01/2016	Stesura sottosezione "Progettazione" e "Codifica".
0.0.2	M. Mbouenda (Amministratore)	27/12/2015	Inizio stesura della sezione "Processo di Sviluppo"
0.0.1	M. Mbouenda (Amministratore)	21/12/2015	Prima stesura del documento. Redazione della sezione "Introduzione" e strutturazione delle sezioni "Processo di Sviluppo", "Processo di supporto", "Processo organizzativo".

Tabella 1: Versionamento del documento

Indice

Elenco delle tabelle

Elenco delle figure

Sommario

In questo documento sono illustrate le "Norme di Progetto" del gruppo **Team404** relativo al capitolato **Quizzipedia**, commissionato da **Zucchetti S.p.A.**
Lo scopo del documento è di descrivere le regole e procedure adottate dal gruppo per la realizzazione del capitolato.

1 Introduzione

1.1 Scopo del documento

Questo documento riporta le regole e convenzioni per il coordinamento dei rapporti interni ed esterni del gruppo. Definisce anche gli strumenti da utilizzare e le modalità del lavoro al quale ogni membro del gruppo si dovrà sottoporre per garantire una migliore collaborazione e coerenza nello svolgimento del lavoro.
Ogni modifica del presente documento verrà tempestivamente notificata ad ogni componente del team per via dei canali adottati.

1.2 Scopo del Prodotto

Il progetto **Quizzipedia** ha come obiettivo lo sviluppo di un sistema software basato su tecnologie Web (Javascript$_G$, NodeJS$_G$, HTML5$_G$, CSS3$_G$) che permetta la creazione, gestione e fruizione di questionari. Il sistema dovrà quindi poter archiviare i questionari suddivisi per argomento, le cui domande dovranno essere raccolte attraverso uno specifico linguaggio di markup che verrà chiamato "Quiz Markup Language" e d'ora in poi denominato QML$_G$. In un caso d'uso a titolo esemplificativo, un "esaminatore" dovrà poter costruire il proprio questionario scegliendo tra le domande archiviate, ed il questionario così composto sarà presentato e fruibile all' "esaminando", traducendo l'oggetto QML$_G$ in una pagina HTML$_G$, tramite un'apposita interfaccia web. Il sistema presentato dovrà inoltre poter proporre questionari preconfezionati e valutare le risposte fornite dall'utente finale.

1.3 Glossario

Viene allegato un glossario nel file "*glossario_3.0.pdf*" nel quale viene data una definizione a tutti i termini che in questo documento appaiono con il simbolo '$_G$' a pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Capitolato d'appalto Quizzipedia:**
  http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C5.pdf
- **Analisi dei Requisiti:** "*analisi_dei_requisiti_3.0.pdf*"
- **Piano di Progetto:** "*piano_di_progetto_3.0.pdf*"
- **Piano di Qualifica:** "*piano_di_qualifica_3.0.pdf*"
- **Studio di Fattibilità:** "*studio_di_fattibilità_3.0.pdf*"
- **Introduzione all'uso di git:**
  http://git-scm.com/book/it

1.4.2 Informativi

- Corso di Ingegneria del Software anno 2015/2016:
  http://www.math.unipd.it/~tullio/IS-1/2015/

- Regole del progetto didattico:
  http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/PD01.pdf http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/
  http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/PD01b.html

2 Processi Primari

2.1 Processo di fornitura

2.1.1 Studio di fattibilità

DA COMPLETARE

2.2 Processo di sviluppo

Questo processo ha come scopo la produzione di un elemento di un sistema implementato come prodotto software. Trasforma specifici comportamenti, interfacce, e vincoli di implementazione in azioni che creano un sistema implementato come prodotto software. Il processo produce un software che soddisfa i requisiti architetturali e li verifica attraverso la verifica e la validazione. Il processo consiste delle seguente attività:

1. Analisi dei requisiti
2. Progettazione
3. Codifica
4. Verifica e validazione

2.2.1 Analisi dei requisiti

Sarà compito degli analisti estrarre i requisiti del progetto dal capitolato e/o da eventuali incontri col proponente. Questa attività ha per obiettivo la stesura di un documento affidabile e consistente che descrive le richieste ed esigenze del proponente.

2.2.1.1 Casi d'uso

Ogni caso d'uso dovrà essere descritto nel seguente modo

UC[codice identificativo][gerarchia]

presentare i seguenti campi obbligatori:

- **Codice identificativo:** codice univoco che rappresenterà il caso d'uso.
- **Titolo:** sarà il nome del caso d'uso, scelto a secondo del caso.
- **Attori primari:** identifica i protagonisti coinvolti direttamente.
- **Descrizione:** minimale e precisa che descrive il caso d'uso.
- **Precondizione:** condizioni verificate prima del caso d'uso.
- **Postcondizione:** condizioni verificate alla fine del caso d'uso.
- **Scenario:** descrizione della situazione e del contesto di attuazione del caso d'uso.
- **Diagramma UML :** sarà una rappresentazione grafica del caso.

E a secondo del caso, i seguenti campi saranno specificati dove presenti:

- **Attori secondari:** identifica i protagonisti coinvolti indirettamente.

- **Scenari alternativi:** descrizione delle situazioni che non appartengono al flusso principale di esecuzione.
- **Estensioni:** per segnare le estensione del caso.
- **Inclusioni** per segnare le inclusioni del caso.

2.2.1.2 Requisiti

Ogni requisito dovrà avere i seguenti campi:

- **Codice identificativo:** Ogni requisito sarà rappresentato da una lettera maiuscola iniziale seguita da une serie di numeri separati da un punto che identificherà la relazione gerarchica tra i requisiti dello stesso tipo. Un esempio di questa rappresentazione è la seguente:

  X 1.2.*.n

  dove la lettera iniziale indicherà il tipo di requisito individuato e i successivi numeri rappresenteranno una gerarchia.
- **Descrizione:** dettaglio della necessità del requisito.
- **Priorità:** che potrà essere obbligatoria o facoltativa.

Nel documento di Analisi dei requisiti 3.0 I requisiti dovranno essere dettagliati e suddivisi come segue:

- **Requisiti Funzionali**
- **Requisiti di qualità**
- **Requisiti QML**
- **Requisiti di sistema**

2.2.1.3 Strumenti per il tracciamento dei requisiti

Per il tracciamento dei requisiti verrà usato lo strumento Astah$_G$ nella sua versione gratuita per studenti e per l'utilizzo del quale ogni membro si dovrà registrare usando la mail rilasciata dall'Università degli studi di Padova per ottenere una licenza.

2.2.2 Progettazione

2.2.2.1 Scopo del processo

Tra gli obiettivi di questo processo abbiamo la stesura di documenti, quali la Specifica Tecnica e la Definizione di Prodotto, e soprattutto una vasta produzione di diagrammi UML$_G$ che modellino i requisiti individuati in fase d'analisi. Di seguito vengono elencate le norme a carico dei progettisti.

2.2.2.2 Descrizione

La progettazione deve in modo dimostrabile rispettare tutti i requisiti che il gruppo ha concordato con il committente. In particolare i componenti progettati devono essere tracciabili rispetto al requisito che soddisfano.

2.2.2.3 Diagrammi UML

UML2.0 rimane il linguaggio da usare per i seguenti diagrammi:

- **Diagrammi dei package:** dovranno essere presenti sia per l'architettura generale che di dettaglio, sarà fondamentale per definire i moduli all'interno del framework NodeJS$_G$ richiesto dal capitolato;
- **Diagrammi delle classi:** qualora il progetto utilizzasse delle classi, i diagrammi delle classi dovranno essere presenti sia per l'architettura generale che di dettaglio. Nell'ambiente NodeJS$_G$ a prima vista sembra che siano poco utilizzate, a favore dei package.
- **Diagrammi di sequenza:** utilizzati per definire le interazione tra l'utente e il sistema;
- **Diagrammi di attività:** per evitare di compiere errori verrà utilizzato per facilitare la verifica dei passi dell'algoritmo da implementare ;

Per la realizzazione dei diagrammi verrà utilizzato StarUML$_G$, che è un software di modelazzione open source compatibile con la norme UML2.0. Per maggiori dettagli sul prodotto si farà riferimento alla guida ufficiale disponibile all'indirizzo http://docs.staruml.io/en/latest/.

2.2.2.4 Stile di progettazione

- La progettazione dovrà usare quanto più possibile design pattern$_G$ globalmente affermati, la loro scelta dovrà essere giustificata;
- Suddividere il progetto in moduli, in accordo con lo stile di progettazione dell'ambiente NodeJS$_G$;
- Non utilizzare codice sincrono per operazioni di input e di output.

2.2.3 Specifica tecnica – ST

In fase di progettazione architetturale i progettisti, con il supporto degli analisti, hanno il compito di definire nel documento **ST** l'architettura ad alto livello dell'intero sistema ma anche dei singoli componenti. Questo viene fatto principalmente utilizzando i diagrammi adottati ed elencati in §2.2.2.3. Viene inoltre fornito un tracciamento per le componenti che associa ad ognuno di loro almeno un requisito.

2.2.4 Definizione di progetto – DP

In fase di progettazione di dettaglio, i progettisti hanno il compito di redarre il documento **DP** che amplia quanto riportato in **ST** definendo nel dettaglio le singole componenti del sistema sempre usando i diagrammi elencati in §2.2.2.3. Oltre al tracciamento delle componenti, vengono progettati in questa fase anche i test di unità da descrivere nel documento **PQ**. A seguito della stesura di questo documento, potrà iniziare la fase di codifica.

2.2.5 Codifica

2.2.5.1 Scopo del processo

Il processo di codifica ha come obiettivo la trasposizione in codice sorgente del software progettato in Specifica Tecnica e Definizione di Prodotto. Inoltre è necessaria la stesura di documentazione sul codice per assicurarne un buon grado di leggibilità e manutenibilità.

2.2.5.2 Descrizione

Questo processo porta a un software stabile, affidabile, funzionale e aderente con le richieste del proponente. La codifica verrà eseguita utilizzando gli strumenti in §4.3.1.

2.2.5.3 Intestazione dei file

A seconda del linguaggio di programmazione usato, ogni file di codice dovrà avere, in forma di commento, la seguente intestazione:

```
FileName:  nome_del_file
FilePath:  /uno/due/tre/
Version :  X.Y.Z

History:
-------------------------------------------------
  Autor                  Date             Version
=================================================
 Nome Cognome          GG/MM/AAAA          1.1.3
-------------------------------------------------
   Descrizione 1.1.3
        ...
-------------------------------------------------
 Nome Cognome          GG/MM/AAAA          1.1.2
-------------------------------------------------
   Descrizione 1.1.2
        ...
-------------------------------------------------
 Nome Cognome          GG/MM/AAAA          1.1.1
-------------------------------------------------
   Description 1.1.1
        ...
-------------------------------------------------
```

Figura 1: Esempio di intestazione di file

- **FileName:** nome completo del file ('nome.ext');
- **FilePath:** percorso del file con radice la cartella del progetto;
- **Version:** versione corrente del file;
- **History:** registro delle modifiche del file, composto come segue
  - **Author:** autore della modifica con il ruolo tra parentesi;
  - **Date:** data della modifica;
  - **Version:** versione della modifica;
  - **Description:** descrizione della modifica.

2.2.5.4 Formattazione

La formattazione del codice sorgente deve essere definita in modo rigoroso e consistente, così che tutto il codice sia formattato in maniera chiara e attendibile. A questo scopo si è scelto di usare come riferimento per i file Javascript$_G$ la linea guida della **Jquery Foundation** reperibile all'indirizzo http://contribute.jquery.org/style-guide/js/.

2.2.6 @@ Tecnologia

Per l'implementazione del prodotto abbiamo scelto il framework$_G$ MeteorJS$_G$ - che integra NodeJS$_G$ (come richiesto dal committente) insieme al il database MongoDB$_G$ - che si occuperà della gestione delle comunicazioni tra client e server, mentre il frontend$_G$ verrà gestito per con il framework$_G$ AngularJS$_G$ che ci permetterà di legare il model con la view.

2.2.7 Verifica e validazione

Per la **verifica** di un documento viene scelto dal responsabile di progetto uno tra i membri del gruppo nel ruolo di verificatore, senza suscitare nessun conflitto di interesse, ossia senza che colui designato a verificare un determinato documento abbia partecipato alla sua stesura. Le verifiche automatizzate sui documenti, dove previste, difficilmente sono esaustive e possono tralasciare delle anomalie. Per eseguire la verifica è necessario controllare che il documento rispetti tutte le norme descritte nelle Norme di Progetto.
La **validazione** del documento consiste nel controllare che il documento abbia il giusto contenuto, è un compito che va oltre la semplice verifica delle norme. Richiede una conoscenza a priori del contenuto e degli scopi del documento.

2.2.7.1 Strumenti di verifica

Per l'analisi statica del codice JavaScript$_G$ prodotto, si intende usare lo strumento software **Complexity-report$_G$** , fornito dalla piattaforma GitHub$_G$. Esso fornisce una valutazione delle metriche sull'intero progetto analizzando ogni singolo file e fornendo un report in formato testuale. Per una descrizione dettagliata delle metriche utilizzate per l'analisi statica del codice si rimanda al documento *piano_di_qualifica_3.0.pdf*.

3 Processi di supporto

3.1 Processo di documentazione

3.1.1 Scopo del processo

Questo processo ha lo scopo di definire degli strumenti per rendere coerente e uniforme l'insieme della documentazione prodotta nel ciclo di vita del software.

3.1.2 Procedure

Di seguito verranno presentate le regole e procedure che ogni componente di ***Team404*** dovrà seguire nella redazione e la manutenzione della documentazione di progetto. Il team ha scelto il linguaggio di markup LaTeX per la stesura dei suoi documenti per vari motivi:

- Permette una facile gestione degli indici e dei glossari.
- Dispone di un sistema di "impaginazione" di alta resa tipografica.
- Permette di personalizzare comandi da usare in seguito.
- Dispone di numerose librerie con nuovi comandi.

3.1.3 Template

Sarà creato un file *template.tex* che servirà di struttura iniziale a tutti i documenti in modo da garantire uniformità tra di loro.

3.1.4 Struttura dei documenti

3.1.4.1 Comandi personalizzati

All'interno del file *'template.tex'* verranno creati alcuni comandi generici da usare in modo da mantenere la stessa formattazione, quali:

- **\addglos:** per indicare che la parola è nel glossario aggiungendo una 'G' a pedice alla parola.
- **\beginglos:** per iniziare una sezione di termini del glossario.
- **\beginregistro:** per inserire una riga nel registro delle modifiche.
- **\fineglos:** per chiudere una sezione di termini del glossario.
- **\fineregistro:** per chiudere la tabella del registro delle modifiche.
- **\introtab:** per inserire la tabella di informazioni sul documento.
- **\itemglos:** per inserire un termine e la sua descrizione in una sezione del glossario.
- **\rigaregistro:** per inserire una riga nel registro delle modifiche. Ammette 4 parametri elencati in sezione.
- **\subsubsubsection:** per rendere disponibile la possibilità di usare una sotto sezione di terza livello.

3.1.4.2 Frontespizio

Nella prima pagina verrà inserito il frontespizio che sarà costruito nel secondo ordine:

1. **Nome del progetto:** centrato e con una dimensione pari a 32 pt;
2. **Logo del Gruppo:** Il logo del gruppo è nel file "team_not_found.jpg" che si trova nella cartella principale dei documenti.
3. **Indirizzo email:** centrato e con una dimensione pari a 16 pt;
4. **Nome del documento:** centrato, in maiuscoletto, corredato dalla versione e con una dimensione pari a 24 pt;
5. **Una tabella con informazioni generali del documento:**
   - Nome del documento;
   - Versione del documento;
   - Data di redazione: deve essere indicata secondo il formato [ISO 8601];
   - Redazione: elenco in ordine alfabetico dei redattori del documento;
   - Verifica: elenco in ordine alfabetico dei Verificatori del documento;
   - Approvazione: viene indicato il soggetto responsabile di aver approvato il documento;
   - Uso: interno o esterno;
   - Distribuzione: elenco in ordine alfabetico dei soggetti a cui verrà distribuito il documento in oggetto.
6. **Sommario:** brevissimo riassunto indicante lo scopo del documento.

3.1.4.3 Registro delle modifiche

Nella pagina successiva al frontespizio va inserito il registro delle modifiche per tener traccia di tutte le modifiche che verranno introdotte nel documento. Questo verrà fatto con una tabella formata come segue:

- **Versione:** la versione del documento dopo la modifica;
- **Autore:** l'autore della modifica con il ruolo tra parentesi;
- **Data:** la data della modifica;
- **Descrizione:** una breve indicazione della modifica effettuata.

3.1.4.4 Indice

Per agevolare la consultazione e permettere una lettura ipertestuale e non necessariamente sequenziale, verrà creato una tabella dei contenuti in modo da facilitare la navigazione all'interno del documento.

3.1.4.5 Intestazione

Ogni pagina del documento ad eccezione della prima avrà un intestazione formata nel modo seguente:

- Logo del gruppo a sinistra;
- Nome del progetto affianco al logo;
- Numero e titolo della sezione corrente a destra.

3.1.4.6 Piè di pagina

Come per l'intestazione ogni pagina ad eccezione della prima avrà un piè di pagina formato nel modo seguente:

- L'e-mail del gruppo a sinistra;
- Numero progressivo di pagina a destra.

3.1.5 Versionamento

Ogni documento prodotto deve essere versionato in modo che chiunque lo utilizzi possa avere una cronologia delle sue modifiche. Ad ogni versione corrisponde una riga nel registro delle modifiche. Per numerare le versioni verrà adottata la forma:

v.X.Y.Z

dove **Z**:

- Parte da 0 e non è limitato superiormente;
- Viene incrementata ad ogni modifica del documento.

dove **Y**:

- Parte da 0 e non è limitato superiormente;
- Viene incrementato dal verificatore ad ogni verifica;
- Quando viene incrementato, **Z** viene azzerato.

dove **X**:

- Parte da 1 ed è limitato superiormente dal numero di revisioni;
- Viene incrementato dal Responsabile del gruppo quando approva il documento;
- Quanto viene incrementato, **Y** e **Z** vengono azzerati.

Le righe di registro dovranno apparire in ordine decrescente dall'ultima modifica fatta.

3.1.6 Norme tipografiche

3.1.6.1 Formattazione del testo

- **Grassetto:** da usare per i titoli, per gli elementi di un elenco. Verrà usato anche per le parole significative;
- **Glossario:** viene aggiunta una lettera '**G**' a pedice a tutte le parole che hanno una corrispondenza nel glossario;

3.1.6.2 Punteggiatura

- **Maiuscolo:** la lettera iniziale maiuscola va utilizzata solo per i nomi propri o dopo i segni di punteggiatura punto, punto interrogativo e punto esclamativo;
- **Numeri:** per rappresentare i numeri verrà usato lo standard del sistema internazionale SI/ISO 31 che usa lo spazio come separatore delle migliaia e la virgola come separatore decimale (ad esempio 1 234 567,89);

3.1.6.3 Composizione del testo

- **Elenco puntato:**
  - La prima parola va sempre con la lettera maiuscola;
  - La prima parola va in grassetto se è seguita da una descrizione della parola stessa;
  - Ogni punto dell'elenco deve terminare con un punto e virgola, tranne l'ultimo che deve terminare con un punto.

3.1.6.4 Formati

- **Orario**: sarà usato lo standard ISO 8601 per rappresentare l'orario nel formato HH:MM dove:
  - HH: indica le ore, da scrivere sempre con 2 cifre;
  - MM: indica le minuti, da scrivere sempre con 2 cifre.
- **Data:** per agevolare la leggibilità, per la data verrà usato il formatto GG/MM/AAAA inverso allo standard dove:

- GG: indica il giorno;
- MM: indica il mese;
- AAAA: indica l'anno.

- **Link:** i link ipertestuali dovranno essere di colore blu.

3.1.7 Sigle

Sono stare adottate le seguente sigle:

- **AR** -> Analisi dei Requisiti;
- **PP** -> Piano di progetto;
- **NP** -> Norme di progetto;
- **SF** -> Studio di fattibilità;
- **PQ** -> Piano di qualifica;
- **ST** -> Specifica tecnica;
- **MU** -> Manuale utente;
- **DP** -> Definizione di prodotto;
- **RR** -> Revisione dei requisiti;
- **RP** -> Revisione di progettazione;
- **RQ** -> Revisione di qualifica;
- **RA** -> Revisione di accettazione.

3.1.8 Componenti grafiche

3.1.8.1 Tabelle

Ogni tabella nei documenti dovrà avere sotto di essa un numero identificativo e una didascalia. Inoltre dopo l'indice di ogni documento dovrà apparire un elenco delle tabelle inserite.

3.1.8.2 Immagini

Le immagini da inserire in un documento dovranno essere raggruppate in una sotto cartella della cartella contenente il documento stesso; devono essere ben distanziate dai paragrafi che la precedono e da quelli che la seguono ed essere centrate orizzontalmente

Inoltre dopo l'indice di ogni documento dovrà figurare un elenco delle figure e delle immagini presenti. Gli strumenti scelti per la produzione dei documenti sono elencati in §4.3.4

3.1.9 Elenco dei documenti

Per la realizzazione del progetto Quizzipedia viene prodotta una serie di documenti tali:

- **Studio di Fattibilità:** riporta lo studio e le motivazioni che hanno portato alla scelta del capitolato Quizzipedia$_G$
- **Norme di Progetto:** descrivere le regole e procedure adottate dal gruppo per la realizzazione del capitolato.
- **Piano di Progetto:** definisce le scelte progettuali significative adottate, le problematiche d'interesse e un adeguata offerta tecnico-economica relativa al progetto da presentare al committente.
- **Piano di Qualifica:** definisce le strategie di verifica qualitative dei processi che coinvolgono l'attuazione di modelli standard e l'utilizzo di metriche e misure necessarie alla valutazione oggettiva del processo o prodotto in analisi.
- **Analisi dei requisiti:** dallo studio del capitolato d'appalto, individua e definisce chiaramente i requisiti che faranno cardine alla software Quizzipedia$_G$.
- **Specifica Tecnica:** descrive l'architettura software sulla quale verrà sviluppata Quizzipedia$_G$.
- **Definizione di Progetto:** descrive nel dettaglio la struttura del sistema Quizzipedia.
- **Glossario:** contiene la descrizione estesa dei termini utilizzati nei vari documenti.
- **Manuale Utente:** destinata all'utente finale, fornisce una guida dettagliata delle funzionalità di Quizzipedia$_G$
- **Verbali:** contiene un resoconto delle incontri e riunioni del gruppo o del gruppo con il committente.

3.2 Processo di verifica

3.2.1 Scopo del processo

Questo processo ha per scopo di confermare che ogni componente software sviluppato sia conforme ai requisiti.

3.2.2 Risultati osservabili

Il processo di verifica produce i seguenti risultati:

- l'individuazione e l'applicazione di una strategia di verifica;
- i criteri per la verifica di quanto prodotto dal software sono identificati;
- le attività di verifica sono state svolte;
- i difetti sono stati individuati e catalogati;
- i risultati della verifica sono resi disponibili ai proponenti.

3.2.3 Tecniche di analisi

3.2.3.1 Analisi statica

Tale tecnica è attuabile sia alla documentazione sia al codice, consiste nell'individuazione di errori ed anomalie ad esempio effettuando una lettura critica del testo a largo spettro oppure più mirata. Il Verificatore controllerà i documenti e il codice utilizzando le seguenti tecniche:

- **Walkthrough$_G$:** Consiste in una lettura del documento/codice cercando errori ed anomalie a largo spettro senza un'idea precisa di quali tipi errori sarà possibile trovare. Ogni difetto rilevato sarà discusso con gli autori allo scopo di evitare incomprensioni e concordare sulle modifiche necessarie. L'utilizzo è previsto principalmente durante lo sviluppo iniziale, quando non si possiede ancora una chiara visione sui possibili errori. Un uso ripetuto di questa tecnica rende possibile la stesura di una lista di controlla.
- **Inspection$_G$:** si basa sulla lettura mirata dei documenti/codice. Durante tale lettura si cercano gli errori segnalati nella lista di controllo. Progressivamente, con l'acquisizione di esperienza e grazie al precedente uso della tecnica di walkthrough$_G$, la lista di controllo viene estesa e specializzata, rendendo l'inspection$_G$ sempre più efficace;

3.2.3.2 Analisi dinamica

L'analisi dinamica si applica solamente al prodotto software e viene svolta durante l'esecuzione del codice mediante l'uso di appositi test per verificarne il funzionamento e rilevare possibili difetti d'implementazione. Affinché i test siano utili, devono essere ripetibili e devono trovare errori: infatti ogni test ha un costo, e un test che non aiuti a trovare dei difetti non ha motivo di esistere. I test sono ripetibili se, dati gli stessi dati in input, lo stesso ordine di esecuzione, lo stesso hardware e lo stesso software, ottengo in uscita gli stessi risultati;

3.2.4 Test

3.2.4.1 Test di unità

Il test di unità verifica che ogni singola unità di prodotto software funzioni correttamente. Attraverso questo test si verifica la correttezza di tutti i moduli base che compongono i software andando a limitare gli errori di implementazione.

3.2.4.2 Test di integrazione

Il test di integrazione verifica che due o più moduli precedentemente verificati (con test di unità), una volta assemblati, funzionino ed interagiscano come previsto. Aiuta inoltre a rilevare eventuali difetti residui dei moduli, non individuati nella precedente fase di test. Questo test verifica inoltre l'interazione dei moduli prodotti con componenti software esterne come framework o librerie. È inoltre possibile che siano utilizzate delle componenti fittizie (driver e stub), impiegate al posto di determinati moduli (già pronti o ancora in costruzione) che abbiano un comportamento "sempre corretto" rispetto al contratto dei moduli che sostituiscono: questo permette di effettuare test sul corretto funzionamento dell'intero sottosistema e delle singole parti sviluppate.

3.2.4.3 Test di sistema

I test di sistema consistono nella validazione dei prodotti software. Questo test viene eseguito quando si ritiene il prodotto giunto ad una versione definitiva. Viene quindi verificata la completa copertura dei requisiti da parte del prodotto.

3.2.4.4 Test di regressione

Questo test viene eseguito subito dopo che una componente viene modificata. Consiste nel rieseguire tutti i test per verificare che dopo le modifiche il resto dei moduli continuino a funzionare in modo corretto. Il tracciamento aiuta a capire quali sono i test da ripetere (di ogni tipo) poiché potenzialmente a rischio in caso di modifica.

3.2.4.5 Test di accettazione

Coincide con il collaudo del software in presenza del proponente. Se tale test ha esito positivo, il prodotto sarà considerato sufficientemente maturo da permetterne il rilascio.

4 Processi Organizzativi

4.1 Scopo del processo

Questo processo a come scopo di strutturare e facilitare l'organizzazione interna del gruppo stabilendo norme e modalità di comunicazione e di interazione.

4.2 Processo di gestione

il processo di gestione di uno o più processi ed include le attività ed i tasks che vanno dalla definizione alla implementazione di un processo, dalla pianificazione alla chiusura

4.2.1 Ruoli di progetto

4.2.1.1 Definizione dei ruoli

Durante lo sviluppo di questo progetto vi sono diversi ruoli che definiscono le responsabilità e i compiti da effettuare. Ogni membro del gruppo è tenuto a ricoprire tutti i ruoli almeno una volta. Vediamo nel dettaglio i vari ruoli:

Responsabile:

È il responsabile ultimo, per conto del suo gruppo, dei risultati del progetto. Elabora ed emana piani e scadenze, ed approva l'emissione di documenti. Coordina le attività del gruppo, relazionandosi con il controllo di qualità interno al progetto. Redige Organigramma e Piano di Progetto, ed approva inoltre l'Offerta ed i relativi allegati.

Amministratore:

È responsabile dell'efficienza e dell'operatività dell'ambiente di sviluppo, della redazione e attuazione di piani e procedure di gestione per la qualità. Controlla versioni e configurazioni del prodotto e gestisce l'archivio della documentazione di progetto. Collabora alla redazione del Piano di Progetto e nel contempo redige le Norme di Progetto per conto del Responsabile.

Analista:

È responsabile delle attività di analisi. Redige lo Studio di Fattibilità (documento interno al gruppo) e l'Analisi dei Requisiti.

Progettista:

È responsabile delle attività di progettazione. Redige Specifica Tecnica, Definizione di Prodotto e la parte programmatica del Piano di Qualifica.

Programmatore:

È responsabile delle attività di codifica miranti alla realizzazione del prodotto e delle componenti di ausilio necessarie per l'esecuzione delle prove di verifica e validazione.

Verificatore:

È responsabile delle attività di verifica. Redige la parte retrospettiva del Piano di Qualifica che illustra l'esito e la completezza delle verifiche e delle prove effettuate secondo il piano.

4.2.2 Comunicazione

4.2.2.1 Interna

Per le comunicazioni ad uso interno del gruppo è stato creato un gruppo sull'App$_G$ mobile Whatsapp$_G$ per una rapida interazione tra i membri. Inoltre verrà usato il web tool$_G$ Trello$_G$, il cui uso è spiegato nella sezione §4.2.4.

4.2.2.2 Esterna

Il Responsabile di progetto sarà l'unico che potrà mantenere i contatti con individui esterni al gruppo, per farlo è stata creata una casella di posta elettronica: team404swe@gmail.com. Tutte le comunicazioni con il committente dovranno essere effettuate con questo indirizzo e-mail. Sarà suo dovere informare successivamente, gli altri componenti del gruppo di tutte le informazioni scambiate con il committente.

4.2.3 Incontri

4.2.3.1 Interni

In caso di necessità, uno o più componenti del gruppo possono fare richiesta di una riunione al responsabile del progetto. Sarà compito di quest'ultimo autorizzarla o meno.

4.2.3.2 Esterni

Sarà compito del Responsabile di progetto fissare incontri con il proponente o committente e informare tutti i componenti del gruppo. Ogni riunione comprenderà la stesura di un verbale ufficiale contenete le seguenti informazioni:

- Data e ora;
- Luogo;
- Partecipanti esterni;
- Partecipanti interni;
- Domande e risposte.

4.2.4 Ticketing

Per la gestione dei task$_G$ relativi al progetto, il gruppo ha scelto il tool$_G$ di gestione di progetti Trello$_G$ con il quale è possibile creare, assegnare, seguire e commentare delle liste composte da una o più etichette personalizzabili a secondo delle necessità e suddividendole anche in categoria. Così abbiamo creato una categoria Opens Tickets$_G$ per i tickets aperti e Closed tickets$_G$ per quelli chiusi.Ogni tickets sarà rappresentato da un'etichetta che sarà parametrizzata nel modo seguente:

1. **Titolo:** riassume in modo conciso il task$_G$;

2. **Membri:** sono i membri ai quali viene assegnato il task;
3. **Label:** identifica il tipo di task$_G$ e gli assegna un colore;
4. **Data di scadenza:** indica la data entro la quale il task deve essere completato;
5. **Descrizione:** è una descrizione più ampia che può essere inserita per spiegare il compito da svolgere;
6. **Allegato:** permette di allegare un file al task;
7. **Stato:** rappresenta lo stato del task che può essere successivamente 'Accettato', 'Eseguito' e poi 'Verificato' ed include anche una barra di progressione da 0 a 100%;
8. **Commenti:** è una funzionalità che permette ai membri di scambiare parere e informazioni riguardanti il task.

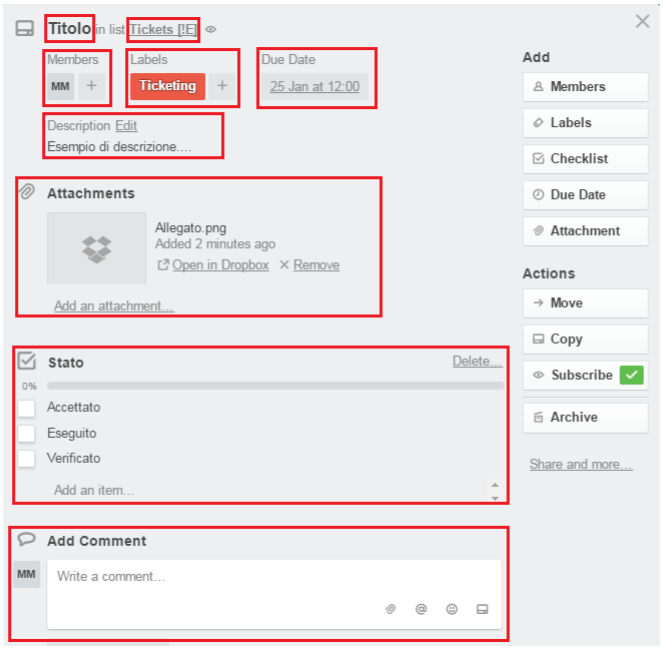

Figura 2: Esempio di configurazione di un task su Trello

Inoltre, ogni componente del team è tenuto a installare la versione mobile di Trello$_G$ per ricevere in tempo reale notifiche relative al progetto in modo da poter agir di conseguenza.

4.3 Processo di Allestimento

Il processo di allestimento e tenuta in esercizio di tutte le infrastrutture e servizi necessari allo sviluppo di uno o più processi

4.3.1 Strumenti di progetto

4.3.1.1 Sistema operativo

La scelta del sistema operativo da utilizzare sarà a discrezione di ogni componente del gruppo, dato che non influirà sul funzionamento dell'applicazione.

4.3.1.2 Codifica dei caratteri

Per assicurarsi la corretta visualizzazione dei caratteri accentati tutti i file testuali presenti nel repository$_G$ (sia codice sorgente che documentazione) devono essere memorizzati con la codifica **UTF-8$_G$.**

4.3.1.3 Versionamento

Per il versionamento dei documenti e del codice sorgente è stato creato un account per il gruppo sul servizio di hosting GitHub$_G$ con nome utente **team404swe** con tutti i membri del gruppo come contributori. Nell'account sono stati creati le repository$_G$ **"team404swe/Documenti"** per i documenti e **"team404swe/Quizzipedia"** per il codice sorgente.Quest'ultimi verranno aggiornati in tutta sicurezza attraverso il software **GitHub Desktop$_G$** dai sistemi compatibili (Windows$_G$ e Mac OSX$_G$)per la gestione delle repository$_G$ GitHub$_G$.

4.3.1.4 Git – Modo d'uso

Qualche comandi utili per prendere in mano Git$_G$.

1. Per creare una copia di un repository locale

   ```
   $ git clone /percorso/del/repository
   ```

2. Per creare una copia di un repository locale usando invece un server remoto

   ```
   $ git clone nomeutente@host:/percorso/del/repository
   ```

3. Per proporre modifiche al file <nomedelfile>

   ```
   $ git add <nomedelfile>
   ```

4. Per proporre modifiche a tutti i file

   ```
   $ git add *
   ```

5. Per validare queste modifiche proposte

   ```
   $ git commit -m "Messaggio per la commit"
   ```

6. Per inviare queste modifiche al repository remoto

   ```
   $ git push origin master
   ```

7. Per aggiornare il tuo repository locale alla commit più recente

   ```
   $ git pull
   ```

4.3.1.5 GitHub – Modo d'uso

GitHub Desktop essendo molto intuitivo, si farà riferimento direttamente alla guida ufficiale disponibile all'indirizzo: https://guides.github.com/introduction/getting-your-project-on-github/ o direttamente tramite riga di comando come in §4.3.1.4

4.3.2 @ Openshift online

Per ospitare il nostro software online abbiamo scelte la piattaforma **OpenShift$_G$ Online** che mette a disposizione, grazie all'account creato con la mail del gruppo, una vasta gamma di linguaggi e servizi distribuiti in applicazioni che possono anche essere framework$_G$ tra i quali MeteorJS$_G$ che useremmo per il nostro progetto.
Questa piattaforma ci permette inoltre di lavorare direttamente in locale, salvando di volta in volta il lavoro in una apposita repository$_G$ direttamente da riga di comando attraverso l'applicazione di supporto di RedHat$_G$ rhc$_G$ che da windows richiede l'installazione dei software Ruby$_G$ versione 1.9.3 o 2.0.0, e Git$_G$ (ultima versione disponibile) per funzionare correttamente.
Per la procedura ufficiale e dettagliata per windows, seguire le istruzioni al seguente link https://developers.openshift.com/getting-started/windows.html

4.3.2.1 @ MeteorJS

La piattaforma NodeJS$_G$ Questo framework$_G$ web basato su NodeJS$_G$ che produce codice multi piattaforma. Si integra con MongoDB$_G$ e utilizza lo stile architetturale Publish/Subscribe$_G$. E' compatibile con con qualsiasi libreria di interfaccia utente e per il nostro progetto viene utilizzato il framework$_G$ AngularJS$_G$.

4.3.2.2 @ AngularJS

Il framework$_G$ AngularJS$_G$ viene usato per esaltare e potenziare l'approccio dichiarativo dell'HTML$_G$ nella definizione dell'interfaccia grafica sfruttando la Single Page Application$_G$ in modo da velocizzare il caricamento delle nostre pagine. Per il nostro progetto, AngularJS$_G$ è stato installato insieme a MeteorJS$_G$ sul nostro server presso OpenShift$_G$. Una documentazione ricca di esempi e dettagli viene fornita dagli sviluppatori del progetto ed è disponibile a seguente link:

4.3.2.3 @@ MaterializeCss – da valutare

Con l'obbiettivo di migliorare l'esperienza dell'utente finale del nostro software, viene utilizzato il framework MaterializeCss$_G$ creato da Google e basato sul Material Design$_G$ e compatibile con i maggiori browser. MaterializeCss$_G$ è un semplice framework$_G$ di interfaccia utente frontend$_G$ per la costruzione per la realizzazione di progetti web responsive$_G$ e user-friendly$_G$. Per usarlo, basterà inserire all'interno delle pagine HTML$_G$ principali del seguente codice:

@@input: file con codice di inserimentomaterialize.js

Per altri riferimenti specifici o per un ampia documentazione, si farà riferimento alla guida ufficiale disponibile all'indirizzo: http://materializecss.com/

4.3.3 @@@ Installazione Meteor–Angular–Materialize

Per installare MeteorJS$_G$ sul proprio computer, ogni membro dovrà seguire le seguente istruzioni:

```
//Installazione MeteorJS AngulaJS MaterializeCss
// da riga di commando


//installa meteorJS
$ curl https://install.meteor.com/ | sh

//crea l'applicazione quizzipedia
$ meteor create quizzipedia

//entra nella cartella dell'applicazione
$ cd quizzipedia

//instala nodejs
$ meteor npm install

//esegue l'applicazione
$ meteor

//a questo punto ci si collega dal browser al link
        http://localhost:3000

//rimuovere i templates Blaze di default su meteor
$ meteor remove blaze-html-templates

//aggiungere  i templates di angularJS
$ meteor add angular-templates

//aggiungere templates angular per nodeJS
$ meteor npm install --save angular angular-meteor


// per windows basta scaricare e installare il programma
// dal seguente link
        https://install.meteor.com/windows

        //FINE
```

Figura 3: Istruzioni Meteor-Angular-Materialize

4.3.4 Produzione dei documenti

4.3.4.1 Dropbox

Per la condivisione di file e documenti non soggetti a controllo di versione tra i membri del gruppo. A questo scopo è stata creata una cartella che contiene anche i vari manuali degli strumenti di interesse per il progetto.

4.3.4.2 Editor

Come editor per i documenti viene scelto TexMaker$_G$, un programma open source per scrivere documenti in LaTeXche offre multiple funzionalità quali il supporto dell'unicode, la colorazione sintattica e la correzione autografica nonché la possibilità di esportare il documento in HTML$_G$ o in formato OpenDocument$_G$.

4.3.4.3 Correttore ortografico

Come correttore ortografico dei documenti prodotti verrà utilizzato il software **MySpell**$_G$ integrato nell'editor TexMaker.