CONFIGURATION MANAGEMENT

Versione: 0.0.0 22/11/2018

Organizzazione:

Il team GL^3 è composto da tre membri: Baradel Luca, Di Bendetto Gianluca, Pellizzari Luca.

Il team si riunisce due volte a settimana:

1) una subito dopo l'incontro con il cliente per fare un'attività di brainstorming quindi per cercare di chiarire gli eventuali dubbi e per individuare gli obiettivi comuni del lavoro;
2) una precedente al prossimo incontro con il cliente per implementare il lavoro pianificato nell'incontro precedente.

Nella prima delle due sedute inoltre si pianifica il lavoro individuale che i membri del team dovranno svolgere entro la data prevista per il secondo incontro in modo da poter discutere (ed eventualmente modificare) i risultati ottenuti prima di mostrarli al cliente.

Il conduttore scelto per il gruppo è Gianluca Di Benedetto.

Gli altri ruoli dei membri del team sono i seguenti:

- Baradel Luca: documentazione, gestione versioni, database;
- Di Benedetto Gianluca: responsabile del CM, wiki;
- Pellizzari Luca: codice, change request, software (tool) di supporto.

Lo strumento scelto dal team per gestire i file relativi al progetto è GitHub; inoltre è stato creato un gruppo Whatsapp per accordarsi sulle date degli incontri e per ulteriori comunicazioni.

Identificazione oggetti (item):

Gli item identificati in questa prima fase del progetto sono i seguenti:

1) Documentazione:
    a. Documento dei requisiti;
    b. Analisi dei requisiti;
    c. Manuale utente;
    d. Verbale di interazione con il cliente;
    e. Offerta in relazione al mandato;
2) Change request;
3) Codice;
4) Gestione versioni;
5) Database;
6) Wiki;
7) Software (tool) di supporto.

Di seguito vengono elencate le caratteristiche principali degli item e il modo in cui questi item verranno gestiti dal team:

1) Documentazione:
    a. Documento dei requisiti:
        i. I requisiti vanno classificati in base allo stakeholder (cliente, sviluppatori, utenti finali, eccetera);

ii. Ogni requisito viene scritto in linguaggio naturale strutturato e deve essere inserito nel documento dei requisiti insieme ad un'eventuale motivazione;
iii. Un requisito può essere associato ad una priorità che indica il suo valore di business.

b. Analisi dei requisiti:
   i. Il documento di analisi dei requisiti deve contenere, per ogni requisito presente nel documento dei requisiti, una descrizione che ci permette di trasformare il requisito del cliente in un requisito del sistema. In altre parole, ogni azione descritta dal cliente deve essere mappata in un'azione che deve compiere il sistema;
   ii. Ad ogni documento di analisi dei requisiti deve essere associata una data e un codice che identifica la versione del documento (come descritto nella gestione delle versioni).

c. Manuale utente:
   i. Tutte le possibili operazioni che l'utente può svolgere tramite il sistema software che verrà sviluppato devono essere descritte in un manuale;
   ii. Ogni aggiornamento del sistema che modifica le funzionalità messe a disposizione dallo stesso o che cambia il modo di effettuare alcune operazioni deve essere descritto nel manuale: fra questo e il sistema non devono esserci inconsistenze;
   iii. Ogni versione (non variante) del sistema deve essere associato ad una versione del manuale.

d. Verbale relazione con il cliente:
   i. Deve contenere: data incontro, luogo incontro, argomenti trattati e per ogni argomento la descrizione dei contenuti trattati;
   ii. Deve essere mostrato al cliente per presa visione entro l'incontro successivo a quello descritto nel verbale.

e. Offerta in relazione al mandato:
   i. L'offerta viene formulata in linguaggio naturale strutturato senza la presenza di termini tecnici legati all'implementazione;
   ii. Deve contenere la data dell'offerta, codice di versione ed eventualmente un glossario con i riferimenti a regolamenti o fonti esterne;
   iii. Contiene una descrizione generale della soluzione proposta che descrive le funzionalità del sistema dal punto di vista dell'utilizzatore. Queste funzionalità dipendono dai requisiti forniti dal cliente e dalla loro successiva analisi svolta dal team di sviluppo.

2) Change request:
   a. Contengono: un codice univoco (numero della richiesta), autore della richiesta, data della richiesta, priorità, descrizione, moduli coinvolti (opzionale se esterna), tempo di lavoro stimato e successivamente data approvazione/rifiuto, data inizio modifiche;
   b. Possono essere interne (esigenze degli sviluppatori) o esterne (esigenze del cliente);
   c. Se sono esterne si pianifica la data di inizio delle modifiche (in base al tempo di lavoro stimato e alla priorità della modifica), se interne prima devono essere approvate (votazione a maggioranza fra gli sviluppatori);
   d. Possono essere presentate in formato cartaceo durante gli incontri (fra gli sviluppatori o con il cliente) oppure in formato digitale (ad esempio pdf) ma in entrambi i casi devono contenere le informazioni elencate al punto a.

3) Codice:

a. Il codice sorgente legato al progetto viene gestito tramite GitHub, ogni membro del team può leggerlo e modificarlo a condizione che gli altri membri siano d'accordo (se necessario va effettuata una change request per poter modificare il codice);
b. Ogni volta che uno degli sviluppatori effettua una modifica o aggiunge nuovo codice deve registrare i cambiamenti tramite la procedura per la gestione delle versioni.

4) Gestione versioni:
    a. È stato scelto di gestire le versioni delle componenti del sistema nel seguente modo:
        i. Ogni versione è identificata da un codice numerico a tre livelli (solo i primi due quando presentata al cliente);
        ii. Il terzo livello contiene il numero della revisione interna su un certo item, partendo da zero;
        iii. Il secondo livello contiene il numero della consegna al cliente (può essere zero solo in fase di sviluppo, quindi la prima consegna di un certo oggetto avrà 1 come codice di secondo livello);
        iv. Il codice di primo livello identifica un oggetto che mantiene obiettivi e caratteristiche fondamentali per tutte le sue consegne, questo codice è zero fino a che non si ottiene una versione completa, funzionante e utilizzabile dal cliente.
        v. Esempio:
            1. 0.2 è la seconda consegna di un prototipo con alcune funzionalità minori ma che non può ancora essere considerato un sistema;
            2. 0.2.9 è la nona revisione interna sul prototipo sopra citato;
            3. 1.0.3 è la terza revisione interna della versione completa e funzionante del sistema non ancora rilasciata;
            4. 1.1 è la prima versione completa e funzionante del sistema consegnata al cliente;
            5. 3.1.15 è la quindicesima revisione interna della prima consegna della terza versione completa e funzionante del sistema.