TheFellowshipOfTheCode

Piano di Qualifica

Informazioni sul documento

Nome Documento	PianoDiQualifica_v1_0_0.pdf
Versione	1
Data di Creazione	12 Dicembre 2015
Data ultima modifica	20 Gennaio 2016
Stato	Approvato
Redazione	Marco Prelaz
	Mattia Varotto
Verifica	Simone Magagna
Approvazione	Matteo Granzotto
Uso	Esterno
Distribuzione	TheFellowshipOfTheCode
Destinato a	Prof. Tullio Vardanega,
	Prof. Riccardo Cardin,
	Zucchetti S.P.A.
Email di riferimento	thefellowshipofthecode@gmail.com

Sommario

Documento contenente le strategie adottate dal gruppo TheFellowshipOfTheCode per raggiungere gli obiettivi qualitativi richiesti per il prodotto QuizziPedia.

Registro delle modifiche

Versione	Descrizione	Autore e Ruolo	Data
1.0.7	Creazione sezione Tracciamento dei test	Mattia Varotto Analista	2016-03-03
1.0.6	Stesura sezione Specifica dei test	Mattia Varotto Analista	2016-03-03
1.0.5	Rimozione sezione Gestione amministrativa della revisione	Mattia Varotto Analista	2016-03-02
1.0.4	Rimozione appendici PDCA, SPICE e Standard ISO/IEC 9126	Marco Prelaz Analista	2016-03-02
1.0.3	Stesura sezione Qualità di prodotto	Mattia Varotto Analista	2016-03-01
1.0.2	Stesura sezione Qualità di processo	Marco Prelaz Analista	2016-03-01
1.0.1	Sostituzione della sezione Strategie di Verifica con le sezioni Qualità di processo e Qualità di prodotto	Mattia Varotto Analista	2016-02-29
1.0.0	Approvazione del documento	Matteo Granzotto Responsabile	2016-01-20
0.3.0	Verifica del documento	Simone Magagna Verificatore	2016-01-19
0.2.1	Aggiunti valori delle verifiche automatizzate	Marco Prelaz Verificatore	2016-01-19
0.2.0	Verifica del documento	Simone Magagna Verificatore	2015-12-29
0.1.1	Modifica paragrafi Scopo del prodotto e Glossario	Mattia Varotto Analista	2015-12-28
0.1.0	Verifica del documento	Simone Magagna Verificatore	2015-12-28
0.0.4	Stesura sezione Gestione amministrativa della revisione e appendici A,B,C,D	Marco Prelaz Analista	2015-12-14
0.0.3	Stesura sezione Strategie di verifica	Mattia Varotto Analista	2015-12-13
0.0.2	Stesura sezione Introduzione	Mattia Varotto Analista	2015-12-12
0.0.1	Creato template	Matteo Granzotto Responsabile	2015-12-12

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del Documento

Lo scopo del presente documento è quello di illustrare le strategie di verifica e validazione che il gruppo TheFellowshipOfTheCode ha deciso di adottare per raggiungere gli obiettivi qualitativi prefissati sul prodotto da sviluppare. Per raggiungere tali obiettivi è necessaria una verifica continua sulle attività svolte. In questo modo sarà possibile individuare rapidamente eventuali anomalie che si potrebbero riscontrare e quindi risolverle in modo tempistico e senza spreco di risorse, garantendo la correttezza del prodotto e il soddisfacimento del cliente.

1.2 Scopo del prodotto

Lo scopo del prodotto è di permettere la creazione e gestione di questionari in grado di identificare le lacune dei candidati prima, durante e al termine di un corso di formazione.
Il sistema dovrà offrire le seguenti funzionalità:

- Archiviare questionari in un server suddivisi per argomento;
- Somministrare all'utente, tramite un'interfaccia, questionari specifici per argomento scelto;
- Verificare e valutare i questionari scelti dagli utenti in base alle risposte date.

La parte destinata ai creatori di questionari dovrà essere fruibile attraverso un *browser*$_G$ desktop, abilitato all'utilizzo delle tecnologie *HTML5*$_G$, *CSS3*$_G$ e *JavaScript*$_G$. La parte destinata agli esaminandi sarà utilizzabile su qualunque dispositivo: dal personal computer ai tablet e smartphone.

1.3 Glossario

Al fine di evitare ogni ambiguità i termini tecnici del dominio del progetto, gli acronimi e le parole che necessitano di ulteriori spiegazioni saranno nei vari documenti marcate con il pedice $_G$ e quindi presenti nel documento *Glossario*.

1.4 Riferimenti

1.4.1 Normativi

- *NormeDiProgetto_v_1_0_0*;
- **Capitolato**: `http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C4.pdf`.

1.4.2 Informativi

- **Piano di Progetto**: *Piano di Progetto*;
- **PDCA (Plan-Do-Check-Act)**: `http://it.wikipedia.org/wiki/PDCA`;
- **Standard ISO/IEC 12207:2008 - IEEE Std 12207-2008**: `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4475822`;
- **Standard ISO/IEC 15504**: `https://en.wikipedia.org/wiki/ISO/IEC_15504`;
- **Standard ISO/IEC 9126**: `http://it.wikipedia.org/wiki/ISO/IEC_9126`;
- **Indice di Gulpease**: `http://it.wikipedia.org/wiki/Indice_Gulpease`;

2 Qualità di processo

Per garantire la qualità del prodotto è necessario perseguire la qualità dei processi che lo definiscono. Per fare ciò si è deciso di adottare lo standard *ISO/IEC 15504*$_G$ denominato *SPICE*$_G$ che definisce un modello di riferimento per la valutazione del livello di maturità dei processi.
Per l'esattezza sono previsti sei livelli di maturità, per i quali vengono definiti degli attributi che permettano di misurarla:

- **Level 0 - Incomplete process**: il processo non è implementato o non riesce a raggiungere i suoi obiettivi;
- **Level 1 - Performed process**: il processo viene messo in atto e raggiunge i suoi scopi. Viene misurato tramite:
  - **Process performance**: capacità di raggiungere i propri obiettivi e di ottenere risultati identificabili.
- **Level 2 - Managed process**: il processo viene eseguito sulla base di obiettivi ben definiti. Viene misurato tramite:
  - **Performance management**: capacità di elaborare un prodotto coerente con gli obiettivi attesi;
  - **Work product management**: capacità di elaborare un prodotto appropriatamente documentato, controllato e verificato.
- **Level 3 - Established process**: il processo viene eseguito in base ai principi dell'ingegneria del software. Viene misurato tramite:
  - **Process definition**: capacità di raggiungere i propri obiettivi aderendo agli standard;
  - **Process resource**: capacità di sfruttare risorse adeguate che gli permettano di essere attuato efficacemente.
- **Level 4 - Predictable process**: Il processo è attuato all'interno di limiti ben definiti. Viene misurato tramite:
  - **Process measurement**: capacità di utilizzare i risultati raggiunti e le misure ricavate durante l'esecuzione per garantire il raggiungimento dei traguardi definiti;
  - **Process control**: capacità di correggere o migliorare, se necessario, le sue modalità di esecuzione, in seguito a controlli basati sulle misurazioni rilevate.
- **Level 5 - Optimizing process**: Il processo è predicibile ed in grado di adattarsi per raggiungere obiettivi specifici e rilevanti. Viene misurato tramite:
  - **Process change**: capacità di tenere sotto controllo tutti i cambiamenti strutturali e di esecuzione;
  - **Continuous improvement**: capacità di identificare e implementare le modifiche effettuate, per garantire un miglioramento continuo nella realizzazione degli obiettivi fissati.

Ogni attributo definito precedentemente è misurabile e lo standard stabilisce 4 differenti livelli:

- **N**: non posseduto (0% - 15%);
- **P**: parzialmente posseduto (16% - 50%);
- **L**: largamente posseduto (51% - 85%);

- **F**: completamente posseduto (86% - 100%).

Le misurazioni ottenute vengono utilizzate all'interno della strategia di miglioramento continuo della qualità, realizzata attraverso il ciclo *PDCA*$_G$, che definisce una metodologia di controllo dei processi durante il loro ciclo di vita. Tale approccio è suddiviso in 4 fasi:

- **Plan**: fase di pianificazione dove si individuano gli obiettivi e i processi necessari per il raggiungimento dei risultati attesi;
- **Do**: fase di attuazione del piano individuato al passo precedente e raccolta di dati sulla qualità ottenuta;
- **Check**: fase di verifica dove si confrontano i risultati ottenuti (fase di Do) ed i risultati attesi (fase di Plan);
- **Act**: fase in cui si determinano le cause delle differenze fra risultati ottenuti e risultati attesi, per decidere dove attuare eventuali azioni correttive per avere un effettivo miglioramento della qualità.

Per garantire una buona qualità di processo, il *team*$_G$ ha individuato dallo standard *ISO/IEC 12207:2008*$_G$ i processi che ritiene più importanti nell'arco del ciclo di vita del prodotto e li ha istanziati individuando obiettivi e metriche coerenti con i livelli di qualità perseguiti.

2.1 Infrastructure Management Process (6.2.2)

Il processo si pone come obiettivo quello di fornire, mantenere ed aggiornare l'infrastruttura ed i servizi necessari allo svolgimento del progetto nell'arco di tutto il suo ciclo di vita. Con il termine infrastruttura si intendono elementi hardware, software, metodi, strumenti, tecniche e standard impiegati nello sviluppo del prodotto.

2.1.1 Obiettivi di qualità

Per tutta la durata del progetto, l'infrastruttura impiegata nello sviluppo dovrà raggiungere determinati obiettivi; in particolare:

- tutte le procedure riguardanti le attività svolte più frequentemente durante lo sviluppo del progetto saranno descritte esaustivamente nel documento *Norme di Progetto*;
- tutti i riferimenti normativi e informativi saranno completi di informazioni utili al loro reperimento;
- la piattaforma DocumentsDB sarà disponibile all'uso ogniqualvolta un componente del *team*$_G$ avesse bisogno di accedere ai dati in essa contenuti;
- i dati ottenuti da DocumentsDB saranno sempre coerenti e aggiornati;
- nell'arco dello sviluppo del progetto, DocumentsDB si estenderà fornendo nuove funzionalità in relazione alle nuove attività intraprese; esse risulteranno implementate e funzionanti alla prima necessità di utilizzo;

2.1.2 Strategie

L'infrastruttura necessaria allo svolgimento del progetto dovrà essere mantenuta costantemente aggiornata; in particolare l'utilizzo delle metriche sotto indicate permetterà l'individuazione di eventuali errori all'interno degli strumenti utilizzati, la cui correzione permetterà di ripristinare l'erogazione di dati corretti e coerenti.

2.1.3 Metriche

2.1.3.1 Disponibilità DocumentsDB

Indica la percentuale di disponibilità di utilizzo della piattaforma DocumentsDB rispetto alle richieste di accesso.

- **Misurazione**: $D = \frac{A}{R} \cdot 100$, dove A corrisponde al numero di accessi avvenuti correttamente alla pagina di login della piattaforma e R il numero totale di richieste di accesso alla pagina di login inoltrate alla piattaforma;
- **Range-ottimale**: $90 - 100$;
- **Range-accettazione**: $80 - 100$.

2.1.3.2 Tempo di correzione incoerenze in DocumentsDB

Indica il periodo medio intercorso fra l'individuazione di un'incoerenza nella piattaforma DocumentsDB da parte di un verificatore ed il suo aggiornamento da parte di un altro componente del $team_G$.

- **Misurazione**: $T = \frac{\sum_{i=1}^{n} C_i}{n}$ (con T espresso in giorni) dove C_i è il tempo intercorso fra il momento di individuazione dell'incoerenza i in DocumentsDB e l'istante in cui tale dato viene corretto;
- **Range-ottimale**: $0 - 1$;
- **Range-accettazione**: $0 - 3$.

2.2 Project Planning, Assessment & Control Process (6.3.1 - 6.3.2)

Il macro-processo (derivante dall'unione dei processi Project Planning Process e Project Assessment & Control Process) ha lo scopo di produrre dei piani di sviluppo per il progetto, comprendenti la scelta del modello di ciclo di vita del prodotto, descrizioni delle attività e dei compiti da svolgere, pianificazione temporale del lavoro e dei costi da sostenere, allocazione di compiti e responsabilità, e misurazioni per rilevare lo stato del progetto rispetto alle pianificazioni prodotte.

2.2.1 Obiettivi di qualità

L'intero sviluppo del progetto dovrà seguire la pianificazione prodotta, in particolare:

- ogni attività verrà svolta da parte di colui al quale è stata assegnata, rispettando le tempistiche fissate e svolgendo tutti i compiti nei quali è stata suddivisa;
- il costo necessario allo svolgimento di una fase di progetto non dovrà eccedere quanto preventivato per tale fase.

2.2.2 Strategie

La pianificazione effettuata dovrà essere aggiornata costantemente durante tutta l'attività di progetto per essere sempre coerente con la situazione corrente. Qualsiasi eventuale valore negativo a livello di Schedule Variance o Budget Variance rilevato in una fase di lavoro dovrà essere assolutamente compensato entro la fine dell'attività di progetto, in quanto non è assolutamente ammesso eccedere le ore di lavoro finali e il preventivo dei costi finale indicato nella pianificazione.

2.2.3 Metriche

2.2.3.1 Schedule Variance

Indica se si è in linea, in anticipo o in ritardo rispetto la pianificazione temporale delle attività nella $baseline_G$.

- **Misurazione**: $SV = BCWP - BCWS$, dove $BCWP$ sono le attività completate ad un certo momento e $BCWS$ le attività che, secondo la pianificazione, dovrebbero essere state completate a quel momento;
- **Range-ottimale**: ≥ 0;
- **Range-accettazione**: ≥ 0.

2.2.3.2 Budget Variance

Indica se alla data corrente si è speso di più o di meno rispetto a quanto pianificato.

- **Misurazione**: $BV = BCWS - ACWP$, dove $BCWS$ è il costo pianificato per realizzare le attività di progetto alla data corrente e $ACWP$ è il costo effettivamente sostenuto alla data corrente;
- **Range-ottimale**: ≥ 0;
- **Valore di accettazione**: ≥ 0.

2.3 Risk Management Process (6.3.4)

L'obiettivo del processo è quello di identificare, analizzare, trattare e monitorare continuamente i rischi che possono insorgere durante l'intera attività di progetto.

2.3.1 Obiettivi di qualità

Il $team_G$ dovrà gestire correttamente i rischi, in particolare:

- all'inizio dell'attività di progetto, verranno individuati i principali fattori di rischio riguardanti l'organizzazione delle attività;
- all'inizio di ogni fase, l'analisi dei rischi porterà all'individuazione di nuovi rischi specifici per tale fase;
- i rischi analizzati che si paleseranno saranno trattati secondo le strategie individuate in fase di individuazione e il loro impatto sarà controllato.

2.3.2 Strategie

Il livello di probabilità dei rischi analizzati dovrà sempre essere tenuto sotto controllo. Anche se a basso livello di pericolosità, in caso il rischio si manifestasse, il $team_G$ dovrà attuare le contromisure previste al fine di mitigare i suoi effetti ed evitare che la sua pericolosità aumenti.

2.3.3 Metriche

2.3.3.1 Rischi non preventivati

Indicatore che evidenzia i rischi non preventivati.

- **Misurazione**: indice numerico che viene incrementato nel momento in cui si manifesta un rischio non individuato nell'attività di analisi dei rischi;

- **Range-ottimale**: 0;
- **Range-accettazione**: $0 - 5$.

2.3.3.2 Efficienza di gestione dei rischi

Misura il tempo medio trascorso fra l'individuazione di un rischio e il momento in cui manifesta in modo problematico i suoi effetti.

- **Misurazione**: $E = \frac{\sum_{i=1}^{n}(M_i \cdot P_i)}{\sum_{i=1}^{n} P_i}$, (con E espresso in giorni) dove M_i è il tempo intercorso fra l'individuazione del rischio i e l'istante in cui manifesta in modo problematico i suoi effetti, espresso in giorni, e P_i corrisponde al grado di pericolosità del rischio i, valutato in scala crescente [1-5];
- **Range-ottimale**: ≥ 60;
- **Range-accettazione**: ≥ 20.

2.4 System/Software Requirements Analysis Process (6.4.2 - 7.1.2)

Il processo punta a trasformare i requisiti individuati dalle fonti in un set di requisiti tecnici che fungerà da linea guida nella progettazione del sistema.

2.4.1 Obiettivi di qualità

I requisiti identificati dal $team_G$ dovranno essere gestiti in maniera tale da raggiungere i seguenti traguardi:

- per ogni requisito verrà tenuta traccia della fonte da cui è stato ricavato;
- per ogni requisito dovrà essere possibile indicare dei test, da effettuare per verificarne il soddisfacimento da parte del prodotto;
- per ogni requisito sarà possibile ricostruire i cambiamenti principali effettuati nella sua formulazione, durante tutto il ciclo di sviluppo del prodotto;
- nessun requisito dovrà risultare superfluo o ambiguo;
- tutti i requisiti che il prodotto andrà a soddisfare saranno stati precedentemente approvati dai committenti.

2.4.2 Strategie

Tutti i requisiti individuati dovranno essere correttamente inseriti nella piattaforma DocumentsDB, la quale si occuperà di mantenere traccia delle fonti dalle quali derivano, delle modifiche effettuate e della loro implementazione nel prodotto.

2.4.3 Metriche

2.4.3.1 Requisiti obbligatori soddisfatti

Indica la percentuale dei requisiti obbligatori soddisfatti dal prodotto.

- **Misurazione**: $S = \frac{N_S}{N_O} \cdot 100$, dove N_S è il numero dei requisiti obbligatori soddisfatti dal sistema e N_O è il numero dei requisiti obbligatori identificati;
- **Range-ottimale**: 100;
- **Range-accettazione**: 100.

2.5 System/Software Architectural Design Process (6.4.3 - 7.1.3)

Il processo si pone come obiettivo quello di identificare una corrispondenza fra requisiti di sistema ed elementi del sistema.

2.5.1 Obiettivi di qualità

Durante lo svolgimento delle attività previste da questo processo, il $team_G$ punterà a definire un'architettura adatta agli scopi del progetto:

- ogni componente progettato come parte del sistema risulterà essere necessario per il funzionamento del prodotto e, quindi, costantemente tracciabile ai requisiti che soddisfa;
- il sistema dovrà presentare basso accoppiamento ed alta coesione
- ogni componente dovrà essere progettato puntando su incapsulamento, modularizzazione e riuso di codice.

2.5.2 Strategie

Nel corso dell'attività di progettazione, sia ad alto livello che di dettaglio, le componenti verranno inserite nella piattaforma DocumentsDB, la quale si occuperà di mantenere aggiornati i tracciamenti fra esse ed i requisiti che soddisfano, oltre alle relazioni presenti fra le varie componenti.

2.5.3 Metriche

2.5.3.1 Structural Fan-In

In riferimento ad un modulo del software, misura quanti altri moduli lo utilizzano durante la loro esecuzione; tale indicazione permette di stabilire il livello di riuso implementato.

- **Misurazione**: indice numerico che incrementa nel momento in cui viene individuato un modulo che, durante la sua esecuzione, chiama il modulo in oggetto;
- **Range-ottimale**: ≥ 2;
- **Range-accettazione**: ≥ 0.

2.5.3.2 Structural Fan-Out

In riferimento ad un modulo del software, misura quanti moduli vengono utilizzati durante la sua esecuzione; tale indicazione permette di stabilire il livello di accoppiamento implementato.

- **Misurazione**: indice numerico che incrementa nel momento in cui viene individuato un modulo utilizzato dal modulo in oggetto durante la sua esecuzione;
- **Range-ottimale**: $0 - 1$;
- **Range-accettazione**: $0 - 5$.

2.6 Software Detailed Design Process (7.1.4)

Lo scopo del processo è fornire una progettazione di dettaglio del prodotto che andrà ad implementare i requisiti individuati.

2.6.1 Obiettivi di qualità

Le attività svolte dovranno raggiungere i seguenti obiettivi:

- il livello di dettaglio della progettazione dovrà essere tale da guidare codifica e testing senza bisogno di informazioni aggiuntive, indicando metodi con i relativi parametri e campi dati forniti da ciascuna classe;
- la struttura a basso livello dell'architettura e le relazioni fra le varie unità software concepite saranno esposte chiaramente nel documento di *Definizione di Prodotto*, che definirà dettagliatamente cosa implementare;
- oltre alle unità software individuate, le attività permetteranno di definire dettagliatamente le interfacce fra esse costituite.

2.6.2 Strategie

Sarà necessario effettuare un'analisi dettagliata delle componenti individuate in progettazione architetturale, suddividendole in unità che siano facilmente codificabili e testabili per le attività successive.

2.6.3 Metriche

2.6.3.1 Numero di metodi per classe

Indica il numero di metodi definiti in una classe; un valore molto alto potrebbe indicare una cattiva decomposizione delle funzionalità a livello di progettazione.

- **Misurazione**: indice numerico che indica il numero di metodi definiti in una classe;
- **Range-ottimale**: $1 - 7$;
- **Range-accettazione**: $1 - 10$.

2.6.3.2 Numero di parametri per metodo

Indica il numero di parametri passati ad un metodo; un valore molto alto potrebbe indicare un metodo troppo complesso e non efficacemente suddiviso in sotto-metodi.

- **Misurazione**: indice numerico che indica il numero di parametri passato ad un metodo;
- **Range-ottimale**: $0 - 4$;
- **Range-accettazione**: $0 - 8$.

2.7 Software Construction Process (7.1.5)

Il processo definisce le attività principali volte alla produzione di unità software eseguibili che riflettano quanto identificato a livello di progettazione.

2.7.1 Obiettivi di qualità

Le unità software prodotte dovranno risultare di qualità; a questo fine il $team_G$ si è posto i seguenti obiettivi:

- l'implementazione delle classi e dei metodi definiti in progettazione dovrà puntare a produrre codice a bassa complessità, in modo tale che quanto prodotto risulti facilmente comprensibile e testabile;

- l'uso di costrutti e tecniche che creano sdoppiamenti del flusso di esecuzione verrà valutato attentamente ed attuato solo se strettamente necessario;
- il codice prodotto dovrà risultare facilmente manutenibile;
- il codice prodotto risulterà privo di elementi inutilizzati.

2.7.2 Strategie

Durante l'attività di codifica, il *Programmatore* dovrà attenersi a quanto indicato nel documento *Definizione di Prodotto*, concentrandosi (in particolare) nel limitare la complessità del codice prodotto. Sarà necessario inoltre procedere con la codifica dei test individuati nell'attività di progettazione, in modo tale da consentire la verifica del corretto funzionamento delle varie unità prodotte.

2.7.3 Metriche

2.7.3.1 Produttività di codifica

Indica il numero medio di linee di codice prodotto per ora/persona.

- **Misurazione**: $P = \frac{N_{SLOC}}{h_P}$, dove N_{SLOC} è il numero di Source Lines Of Code prodotte e h_P è il numero di ore/persona utilizzate per la codifica;
- **Range-ottimale**: ≥ 10;
- **Range-accettazione**: ≥ 3.

2.7.3.2 Complessità Ciclomatica

Indica la complessità di funzioni, moduli, metodi o classi di un programma misurando il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso. Alti valori di *complessità ciclomatica*$_G$ implicano una ridotta manutenibilità del codice. Valori bassi di *complessità ciclomatica*$_G$ potrebbero però determinare scarsa efficienza dei metodi.

- **Misurazione**: indice numerico che indica il numero cammini percorribili nel grafo di controllo di flusso di un metodo;
- **Range-ottimale**: $1 - 10$;
- **Range-accettazione**: $1 - 15$.

2.7.3.3 Numero di livelli di annidamento

Indica il numero di funzioni o procedure chiamate all'interno di un metodo; un valore elevato di tale indice implica un'alta complessità ed un basso livello di astrazione del codice.

- **Misurazione**: indice numerico che indica il numero di chiamate a funzioni o procedure presenti all'interno di un metodo;
- **Range-ottimale**: $1 - 3$;
- **Range-accettazione**: $1 - 6$.

2.7.3.4 Linee di codice per linee di commento

Indica la percentuale di linee di commento presenti all'interno del codice sorgente; la loro presenza permette una più semplice comprensione ed un maggior livello di manutenibilità di quanto prodotto.

- **Misurazione**: $P = \frac{N_C}{N_{SLOC}} \cdot 100$, dove N_C è il numero di linee di commento presenti nel codice e N_{SLOC} è il numero di Source Lines Of Code prodotte;
- **Range-ottimale**: ≥ 30;
- **Range-accettazione**: ≥ 25.

2.7.3.5 Variabili inutilizzate

Indica la percentuale di variabili dichiarate che non vengono mai utilizzate durante l'esecuzione.

- **Misurazione**: $P = \frac{N_{VI}}{N_{VD}} \cdot 100$, dove N_{VI} è il numero di variabili che non vengono mai utilizzate e N_{VD} è il numero di variabili dichiarate;
- **Range-ottimale**: 0;
- **Range-accettazione**: 0.

2.7.3.6 Dipendenze

Misura il numero medio di chiamate require di una funzione, analizzate staticamente considerando la firma del metodo.

- **Misurazione**: $D = \frac{\sum_{i=1}^{n} R_i}{n}$, dove R_i è il numero di chiamate require effettuate dalla funzione i;
- **Range-ottimale**: $0 - 5$;
- **Range-accettazione**: $0 - 10$.

2.7.3.7 Halstead Difficulty per-function

Misura il livello di complessità di una funzione.

- **Misurazione**: $DIF = \frac{UOP}{2} \cdot \frac{OD}{UOD}$, dove UOP è il numero di operatori distinti, OD è il numero totale di operandi e UOD è il numero di operandi distinti;
- **Range-ottimale**: $0 - 15$;
- **Range-accettazione**: $0 - 25$.

2.7.3.8 Halstead Volume per-function

Indica la dimensione dell'implementazione di un algoritmo; si basa sul numero di operazioni eseguite e sugli operandi di una funzione.

- **Misurazione**: $VOL = (OP + OD) \cdot \log_2(UOP + UOD)$, dove OP è il numero totale di operatori, OD è il numero totale di operandi, UOP è il numero di operatori distinti e UOD è il numero di operandi distinti;
- **Range-ottimale**: $20 - 1000$;
- **Range-accettazione**: $20 - 1500$.

2.7.3.9 Halstead Effort per-function

Rappresenta il costo necessario a scrivere il codice di una funzione.

- **Misurazione**: $E = DIF \cdot VOL$, dove DIF indica l'Halstead Difficulty e VOL è l'Halstead Volume;
- **Range-ottimale**: $0 - 300$;
- **Range-accettazione**: $0 - 400$.

2.7.3.10 Indice di manutenibilità

Permette di stabilire quanto sarà semplice mantenere il codice prodotto.

- **Misurazione**: $MI = 171 - 3.42 \cdot \ln(aveE) - 0.23 \cdot \ln(aveV) - 16.2 \cdot \ln(aveLOC)$, dove $aveE$ è l'Halstead Effort medio per modulo, $aveV$ è la complessità ciclomatica media per modulo, e $aveLOC$ è il numero medio di linee di codice per modulo;
- **Range-ottimale**: $120 - 171$;
- **Range-accettazione**: $100 - 171$.

2.8 System/Software Integration Process (6.4.5 - 7.1.6)

Il processo si occupa di integrare fra loro gli elementi del sistema, rispettando quanto stabilito nell'attività di progettazione, al fine di produrre un prodotto completo tale da soddisfare quanto espresso dai requisiti identificati.

2.8.1 Obiettivi di qualità

Le attività previste da questo processo dovranno puntare a raggiungere un alto livello di automazione, in particolare:

- l'integrazione delle varie parti del sistema sarà completamente automatizzata utilizzando lo strumento di continuous integration Jenkins;
- il livello di integrazione raggiunto del sistema sarà sempre consultabile grazie all'utilizzo dello strumento di continuous integration Jenkins.

2.8.2 Strategie

Sarà necessario configurare accuratamente lo strumento di continuous integration Jenkins affinché esegua dei test di integrazione di quanto prodotto prima che le ultime modifiche diventino parte del sistema.

2.8.3 Metriche

2.8.3.1 Componenti integrate

Indica la percentuale di componenti progettate, attualmente implementate e correttamente integrate nel sistema.

- **Misurazione**: $I = \frac{N_{CI}}{N_{CP}} \cdot 100$, dove N_{CI} è il numero di componenti attualmente integrate nel sistema e N_{CP} è il numero di componenti delineate nell'attività di progettazione;
- **Range-ottimale**: 100;
- **Range-accettazione**: 100.

2.9 System/Software Qualification Testing Process (6.4.6 - 7.1.7)

Lo scopo del processo è quello di assicurare che ogni requisito individuato sia stato implementato nel prodotto.

2.9.1 Obiettivi di qualità

Durante lo svolgimento delle attività, ci si impegnerà affinché:

- le attività di test previste dal processo verranno svolte su un sistema le cui componenti sono verificate e correttamente integrate fra loro;
- il sistema dovrà implementare tutti i requisiti obbligatori individuati nell'attività di analisi.

2.9.2 Strategie

Bisognerà cercare di implementare il maggior livello possibile di automazione nell'esecuzione dei test di sistema, in modo tale che la loro esecuzione non richieda costi eccessivi (soprattutto in termini temporali) e sia possibile eseguirne un numero sufficiente a garantire un'ottima copertura dei requisiti; a tal fine molto importante sarà lo strumento di continuous integration Jenkins, il quale andrà opportunamente configurato per eseguire i test stabiliti.

2.9.3 Metriche

2.9.3.1 Test di Unità eseguiti

Indica la percentuale di test di unità eseguiti.

- **Misurazione**: $UE = \frac{N_{TUE}}{N_{TUP}} \cdot 100$, dove N_{TUE} è il numero di test di unità eseguiti e N_{TUP} è il numero di test di unità pianificati;
- **Range-ottimale**: 100;
- **Range-accettazione**: $90 - 100$.

2.9.3.2 Test di Integrazione eseguiti

Indica la percentuale di test di integrazione eseguiti.

- **Misurazione**: $IE = \frac{N_{TIE}}{N_{TIP}} \cdot 100$, dove N_{TIE} è il numero di test di integrazione eseguiti e N_{TIP} è il numero di test di integrazione pianificati;
- **Range-ottimale**: $70 - 100$;
- **Range-accettazione**: $60 - 100$.

2.9.3.3 Test di Sistema eseguiti

Indica la percentuale di test di sistema eseguiti in modo automatico.

- **Misurazione**: $SE = \frac{N_{TSE}}{N_{TSP}} \cdot 100$, dove N_{TSE} è il numero di test di sistema eseguiti e N_{TSP} è il numero di test di sistema pianificati;
- **Range-ottimale**: $80 - 100$;
- **Range-accettazione**: $70 - 100$.

2.9.3.4 Test di Validazione eseguiti

Indica la percentuale di test di validazione eseguiti manualmente.

- **Misurazione**: $VE = \frac{N_{TVE}}{N_{TVP}} \cdot 100$, dove N_{TVE} è il numero di test di validazione eseguiti e N_{TVP} è il numero di test di validazione pianificati;
- **Range-ottimale**: 100;
- **Range-accettazione**: 100.

2.9.3.5 Test superati

Indica la percentuale di test superati.

- **Misurazione**: $S = \frac{N_{TS}}{N_{TE}} \cdot 100$, dove N_{TS} è il numero di test superati e N_{TE} è il numero di test eseguiti;
- **Range-ottimale**: 100;
- **Range-accettazione**: $90 - 100$.

2.10 Software Documentation Management Process (7.2.1)

Il processo punta a produrre e manutenere le informazioni sul software prodotte dai processi attuati.

2.10.1 Obiettivi di qualità

Il processo di documentazione dovrà perseguire le seguenti direttive:

- la documentazione prodotta dovrà essere chiara e comprensibile a tutti gli *stakeholder*$_G$ e sarà resa disponibile alle parti interessate per la consultazione;
- ogni forma di ambiguità sul significato di un termine utilizzato verrà eliminata grazie al *Glossario*;
- la documentazione prodotta sarà sempre aggiornata ed allineata allo stato attuale del processo di sviluppo del prodotto.

2.10.2 Strategie

Durante la stesura della documentazione, ogni termine con significato ambiguo deve essere indicato (corredato di definizione) nel *Glossario*; gli script automatici provvederanno alla sua segnalazione all'interno del documento.
Ogni documento sarà dotato di numero di versione e corredato da un diario delle modifiche che consente di prendere visione di tutte le azioni effettuate sul testo in oggetto.

2.10.3 Metriche

2.10.3.1 Indice Gulpease

L'*indice Gulpease*$_G$ è un indice di leggibilità di un testo tarato sulla lingua italiana che permette di misurare la complessità dello stile di un documento considerando due variabili linguistiche, la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere.

- **Misurazione**: $G = 89 + \frac{300 \cdot N_F - 10 \cdot N_L}{N_P}$, dove N_F è il numero di frasi, N_L è il numero di lettere e N_P è il numero di parole presenti nel testo;

- **Range-ottimale**: $50 - 100$;
- **Range-accettazione**: $40 - 100$.

2.11 Software Verification Process (7.2.4)

Il processo punta a verificare se qualsiasi elemento del sistema soddisfa completamente i requisiti ad esso correlati.

2.11.1 Obiettivi di qualità

Al fine di garantire qualità nell'attuazione del processo:

- la documentazione verrà verificata attraverso *inspection*$_G$, poiché permette risparmio in termini di tempi e costi;
- i test dinamici effettuati sui vari elementi saranno il più possibile automatizzabili;
- i test dinamici effettuati sui vari elementi del software copriranno una grande parte delle possibili casistiche d'utilizzo.

2.11.2 Strategie

Durante le attività di correzione della documentazione, gli errori più frequenti rilevati saranno riportati in un documento, in modo tale da diventare uno dei punti chiave delle successive attività di *inspection*$_G$.
Per ogni test effettuato verrà tenuto tracciamento del suo esito.

2.11.3 Metriche

2.11.3.1 Branch Coverage

Indica la percentuale di rami decisionali percorsi dai test di unità utilizzati.

- **Misurazione**: $BC = \frac{R_P}{R_T} \cdot 100$, dove R_P è il numero dei rami decisionali percorsi dai test e R_T è il numero di rami decisionali definiti nel software;
- **Range-ottimale**: $80 - 100$;
- **Range-accettazione**: $70 - 100$.

2.11.3.2 Code Coverage

Indica la percentuale di istruzioni che sono eseguite durante i test. Maggiore è la percentuale di istruzioni coperte dai test eseguiti, maggiore sarà la probabilità che le componenti testate abbiano una ridotta quantità di errori. Il valore di tale indice può essere abbassato da metodi molto semplici che non richiedono testing. Esempi di questi metodi sono: get e set.

- **Misurazione**: $CC = \frac{L_M}{L_T} \cdot 100$, dove L_M è il numero di linee di codice monitorata dai test e L_T è il numero di linee di codice implementate nel software;
- **Range-ottimale**: $65 - 100$;
- **Range-accettazione**: $42 - 100$.

3 Qualità di prodotto

Per garantire una buona qualità di prodotto, il $team_G$ ha individuato dallo standard *ISO/IEC 9126*$_G$ le qualità che ritiene più importanti nell'arco del ciclo di vita del prodotto e le ha istanziate individuando obiettivi e metriche coerenti con i livelli di qualità perseguiti.

3.1 Funzionalità (6.1)

Rappresenta la capacità del prodotto di fornire tutte le funzioni che sono state individuate attraverso l'*Analisi dei Requisiti*

3.1.1 Obiettivi di qualità

Il $team_G$ si impegnerà affinché:

- **Adeguatezza (6.1.1)**: le funzionalità fornite siano conformi rispetto le aspettative;
- **Accuratezza (6.1.2)**: il prodotto fornisca i risultati attesi, con il livello di dettaglio richiesto;
- **Sicurezza (6.1.4)**: il prodotto protegga le informazioni e i dati da accessi e modifiche non autorizzati.

3.1.2 Metriche

3.1.2.1 Completezza dell'implementazione funzionale

Indica la percentuale di requisiti funzionali coperti dall'implementazione.

- **Misurazione**: $C = (1 - \frac{N_{FM}}{N_{FI}}) \cdot 100$, dove N_{FM} è il numero di funzionalità mancanti nell'implementazione e N_{FI} è il numero di funzionalità individuate nell'attività di analisi;
- **Range-ottimale**: 100;
- **Range-accettazione**: 100.

3.1.2.2 Accuratezza rispetto alle attese

Indica la percentuale di risultati concordi alle attese.

- **Misurazione**: $A = (1 - \frac{N_{RD}}{N_{TE}}) \cdot 100$, dove N_{RD} è il numero di test che producono risultati discordanti rispetto alle attese e N_{TE} è il numero di test-case eseguiti;
- **Range-ottimale**: 100;
- **Range-accettazione**: $90 - 100$.

3.1.2.3 Controllo degli accessi

Indica la percentuale di operazioni illegali non bloccate.

- **Misurazione**: $I = \frac{N_{IE}}{N_{II}} \cdot 100$, dove N_{IE} è il numero di operazioni illegali effettuabili dai test e N_{II} è il numero di operazioni illegali individuate;
- **Range-ottimale**: 0;
- **Range-accettazione**: $0 - 10$.

3.2 Affidabilità (6.2)

Rappresenta la capacità del prodotto software di svolgere correttamente le sue funzioni durante il suo utilizzo, anche nel caso in cui si presentino situazioni anomale.

3.2.1 Obiettivi di qualità

L'esecuzione del prodotto dovrà presentare le seguenti caratteristiche:

- **Maturità (6.2.1)**: evitare che si verifichino malfunzionamenti, operazioni illegali e restituzione di risultati errati (failure) in seguito a fault;
- **Tolleranza agli errori (6.2.2)**: nel caso in cui si presentino degli errori, dovuti a guasti o ad un uso scorretto dell'applicativo, questi devono essere gestiti in modo da mantenere alto il livello di prestazione.

3.2.2 Metriche

3.2.2.1 Densità di failure

Indica la percentuale di operazioni di testing che si sono concluse in failure.

- **Misurazione**: $F = \frac{N_{FR}}{N_{TE}} \cdot 100$, dove N_{FR} è il numero di failure rilevati durante l'attività di testing e N_{TE} è il numero di test-case eseguiti;
- **Range-ottimale**: 0;
- **Range-accettazione**: $0 - 10$.

3.2.2.2 Blocco di operazioni non corrette

Indica la percentuale di funzionalità in grado di gestire correttamente i fault che potrebbero verificarsi.

- **Misurazione**: $B = \frac{N_{FE}}{N_{ON}} \cdot 100$, dove N_{FE} è il numero di failure evitati durante i test effettuati e N_{ON} è il numero di test-case eseguiti che prevedono l'esecuzione di operazioni non corrette, causa di possibili failure;
- **Range-ottimale**: 100;
- **Range-accettazione**: $80 - 100$.

3.3 Usabilità (6.3)

Rappresenta la capacità del prodotto di essere facilmente comprensibile e attraente in ogni sua parte per qualsiasi utente che lo andrà ad utilizzare.

3.3.1 Obiettivi di qualità

Il prodotto dovrà puntare ai seguenti obiettivi di usabilità:

- **Comprensibilità (6.3.1)**: l'utente deve essere in grado di riconoscerne le funzionalità offerte dal software e deve comprenderne le modalità di utilizzo per riuscire a raggiungere i risultati attesi;
- **Apprendibilità (6.3.2)**: deve essere data la possibilità all'utente di imparare ad utilizzare l'applicazione senza troppo impegno;

- **Operabilità (6.3.3)**: le funzionalità presenti devono essere coerenti con le aspettative dell'utente;
- **Attrattiva (6.3.4)**: il software deve essere piacevole per chi ne fa uso.

3.3.2 Metriche

3.3.2.1 Comprensibilità delle funzioni offerte

Indica la percentuale di operazioni comprese in modo immediato dall'utente, senza la consultazione del manuale.

- **Misurazione**: $C = \frac{N_{FC}}{N_{FO}} \cdot 100$, dove N_{FC} è il numero di funzionalità comprese in modo immediato dall'utente durante l'attività di testing del prodotto e N_{FO} è il numero di funzionalità offerte dal sistema;
- **Range-ottimale**: $90 - 100$;
- **Range-accettazione**: $80 - 100$.

3.3.2.2 Facilità di apprendimento delle funzionalità

Indica il tempo medio impiegato dall'utente nell'imparare ad usare correttamente una data funzionalità.

- **Misurazione**: indicatore numerico, espresso in minuti che tiene traccia del tempo medio impiegato dall'utente nell'apprendere il corretto utilizzo di una funzionalità offerta dal sistema;
- **Range-ottimale**: $0 - 15$;
- **Range-accettazione**: $0 - 30$.

3.3.2.3 Consistenza operazionale in uso

Indica la percentuale di messaggi e funzionalità offerte all'utente che rispettano le sue aspettative riguardo al comportamento del software.

- **Misurazione**: $C = (1 - \frac{N_{MFI}}{N_{MFO}}) \cdot 100$, dove N_{MFI} è il numero di messaggi e funzionalità che non rispettano le aspettative dell'utente e N_{MFO} è il numero di messaggi e funzionalità offerti dal sistema;
- **Range-ottimale**: $90 - 100$;
- **Range-accettazione**: $80 - 100$.

3.4 Efficienza (6.4)

Rappresenta la capacità di eseguire le funzionalità offerte dal software nel minor tempo possibile utilizzando al tempo stesso il minor numero di risorse possibili.

3.4.1 Obiettivi di qualità

Il prodotto dovrà essere efficiente, in particolare:

- **Comportamento rispetto al tempo (6.4.1)**: per svolgere le sue funzioni il software deve fornire adeguati tempi di risposta ed elaborazione;
- **Utilizzo delle risorse (6.4.2)**: il software quando esegue le sue funzionalità deve utilizzare un appropriato numero e tipo di risorse.

3.4.2 Metriche

3.4.2.1 Tempo di risposta

Indica il periodo temporale medio che intercorre fra la richiesta al software di una determinata funzionalità e la restituzione del risultato all'utente.

- **Misurazione**: $T_{RISP} = \frac{\sum_{i=1}^{n} T_i}{n}$ (con T_{RISP} espresso in *secondi*) dove T_i è il tempo intercorso fra la richiesta i di una funzionalità ed il completamento delle operazioni necessarie a restituire un risultato a tale richiesta;
- **Range-ottimale**: $0 - 3$;
- **Range-accettazione**: $0 - 8$.

3.5 Manutenibilità (6.5)

Rappresenta la capacità del prodotto di essere modificato, tramite correzioni, miglioramenti o adattamenti del software a cambiamenti negli ambienti, nei requisiti e nelle specifiche funzionali.

3.5.1 Obiettivi di qualità

Le operazioni di manutenzione andranno agevolate il più possibile adottando le seguenti caratteristiche:

- **Analizzabilità (6.5.1)**: il software deve consentire una rapida identificazione delle possibili cause di errori e malfunzionamenti;
- **Modificabilità (6.5.2)**: il prodotto originale deve permettere eventuali cambiamenti in alcune sue parti;
- **Stabilità (6.5.3)**: non devono insorgere effetti indesiderati in seguito a modifiche effettuate sul software;
- **Testabilità (6.5.4)**: il software deve poter essere facilmente testato per validare le modifiche effettuate.

3.5.2 Metriche

3.5.2.1 Capacità di analisi di failure

Indica la percentuale di failure registrate, delle quali sono state individuate le cause.

- **Misurazione**: $I = \frac{N_{FI}}{N_{FR}} \cdot 100$, dove N_{FI} è il numero di failure delle quali sono state individuate le cause e N_{FR} è il numero di failure rilevate;
- **Range-ottimale**: $80 - 100$;
- **Range-accettazione**: $60 - 100$.

3.5.2.2 Impatto delle modifiche

Indica la percentuale di modifiche effettuate in risposta a failure che hanno portato all'introduzione di nuove failure in altre componenti del sistema.

- **Misurazione**: $I = \frac{N_{FRF}}{N_{FR}} \cdot 100$, dove N_{FRF} è il numero di failure risolte con l'introduzione di nuove failure e N_{FR} è il numero di failure risolte;
- **Range-ottimale**: $0 - 10$;
- **Range-accettazione**: $0 - 20$.

3.6 Portabilità (6.6)

Rappresenta la capacità del software di poter essere utilizzato su diversi ambienti.

3.6.1 Obiettivi di qualità

Sarà agevolata la portabilità del prodotto adottando i seguenti obiettivi:

- **Adattabilità (6.6.1)**: il prodotto deve adattarsi a tutti quegli ambienti di lavoro nei quali è stato previsto un suo utilizzo, senza dover apportare modifiche allo stesso;
- **Sostituibilità (6.6.4)**: l'applicativo deve poter sostituire un altro software che ha lo stesso scopo e lavora nel medesimo ambiente.

3.6.2 Metriche

3.6.2.1 Versioni dei browser supportate

Indica la percentuale di versioni di $browser_G$ attualmente supportate, fra quelle individuate dai requisiti.

- **Misurazione**: $S = \frac{N_{VS}}{N_{VI}} \cdot 100$, dove N_{VS} è il numero di versioni di $browser_G$ supportate dal prodotto e N_{VI} è il numero di versioni di $browser_G$ che devono essere supportate dal prodotto;
- **Range-ottimale**: 100;
- **Range-accettazione**: 100.

3.6.2.2 Inclusione di funzionalità da altri prodotti

Indica la percentuale di funzionalità del software utilizzato in precedenza dall'utente che produce risultati simili a quelli ottenuti dal prodotto in oggetto.

- **Misurazione**: $I = \frac{N_{FPA}}{N_{FPP}} \cdot 100$, dove N_{FPA} è il numero di funzionalità del software utilizzato in precedenza dall'utente che produce risultati simili a quelli ottenuti dal prodotto in oggetto e N_{FPP} è il numero di funzionalità offerte dal software utilizzato in precedenza dall'utente;
- **Range-ottimale**: $90 - 100$;
- **Range-accettazione**: $80 - 100$.

4 Specifica dei test

Il $team_G$ TheFellowshipOfTheCode, al fine di implementare del software che sia di qualità, ha strutturato dei test atti a verificare che il software prodotto rispecchi le funzionalità a fronte di risultati attesi. Questi test sono il frutto dell'applicazione delle tecniche di verifica dinamica che sono state introdotte nel documento Norme di Progetto. Tutte le attività di testing prodotte devono poter essere ripetibili e devono essere deterministiche, al fine di poter fornire delle informazioni utili a intraprendere azioni di correzione, nel caso in cui i risultati ottenuti siano diversi da quelli attesi. Per avere un tracciamento dei test prodotti e dei risultati ottenuti si è scelto di classificare il tutto producendo dei log che siano di facile consultazione e che possano fornire una precisa indicazione di quelli che sono stati gli output di queste attività di verifica, eventuali errori o eventuali risultati che siano non coerenti con quanto in precedenza fissato.

4.1 Tipi di test

Sono stati individuati quattro livelli di testing e sono rispettivamente:

- **Test di unità [TU]:** con questa tipologia di test si cerca di verificare la più piccola parte di lavoro prodotta da un programmatore. Questo si traduce tendenzialmente a verificare i metodi e le funzioni scritte;
- **Test di integrazione [TI]:** con questa tipologia di test si cerca di verificare le componenti di sistema. Più precisamente, l'obiettivo è quello di testare il funzionamento dei vari package prodotti, sia singolarmente che nel loro insieme;
- **Test di sistema [TS]:** con questa tipologia di test si cerca di verificare che il comportamento e il funzionamento dell'architettura siano corretti;
- **Test di validazione [TV]:** con questa tipologia di test si vuole verificare che il lavoro prodotto soddisfi quanto richiesto dal proponente.

4.1.1 Test di Validazione

I test di validazione hanno lo scopo di accertare che tutte le funzionalità richieste dal proponente siano state soddisfatte. Per questo motivo, attraverso delle macro azioni, si andrà a simulare il comportamento generale dell'applicativo e dell'utente che interagisce con esso. I test di validazione saranno organizzati nel modo seguente:

$$\textbf{TV}[\mathit{TipoRequisito}][\mathit{ImportanzaRequisito}][\mathit{IdRequisito}]$$

dove:

- **TipoRequisito** può assumere valori tra:
  - F per i requisiti funzionali;
  - Q per i requisiti di qualità;
  - V per i requisiti di vincolo;
  - P per i requisiti prestazionali.
- **ImportanzaRequisito** può assumere valori tra:
  - D per i requisiti desiderabili;
  - O per i requisiti di obbligatori;
  - F per i requisiti di facoltativi.
- **IdRequisito** assume un valore gerarchico che identifica il singolo requisito.

Id Test	Descrizione	Stato
TVFO1	L'utente intende registrarsi al sistema. All'utente è richiesto di: • Trovarsi nella sezione apposita; • Compilare il form di registrazione; • Premere il pulsante di conferma. • Verificare attraverso l'autenticazione che la registrazione sia effettivamente avvenuta.	*Non Implementato*
TVFO2	L'utente intende autenticarsi al sistema. All'utente è richiesto di: • Trovarsi nella sezione apposita; • Inserire le credenziali nell'apposito form; • Premere il pulsante di autenticazione; • Verificare che l'autenticazione sia effettivamente avvenuta.	*Non Implementato*
TVFO3	L'utente intende disconnettersi dal sistema. All'utente è richiesto di: • Essere Autenticato; • Trovarsi nella sezione apposita; • Premere il pulsante di logout; • Verificare che la disconnessione sia effettivamente avvenuta.	*Non Implementato*
TVFD4	L'utente autenticato intende gestire i propri dati. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Modificare i campi dati consentiti; • Premere il tasto conferma modifica; • Visualizzare il profilo dell'utente modificato.	*Non Implementato*

Id Test	Descrizione	Stato
TVFD4.3	L'utente autenticato intende impostare la propria foto profilo . All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Impostare la foto da inserire; • Visualizzare il profilo dell'utente modificato.	*Non Implementato*
TVFD4.8	L'utente autenticato intende modificare la tipologia di utenza. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Cambiare la tipologia di utenza; • Verificare la modifica effettuata.	*Non Implementato*
TVFD4.9	L'utente autenticato intende eliminare il proprio account. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Premere il tasto di elimanazione; • Verificare la disconnessione della sessione sia avvenuta; • Verificare che l'autenticazione con le credenziali dell'utente eliminato non avvenga.	*Non Implementato*
TVFD5	L'utente autenticato intende ricercare un questionario per iscriversi. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Ricercare un questionario; • Iscrizione ad un questionario trovato mediante l'apposito tasto; • Verificare l'operazione appena effettuata.	*Non Implementato*

Id Test	Descrizione	Stato
TVFO6	L'utente autenticato intende compilare un questionario a cui si è iscritto. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Selezionare il questionario da compilare; • Rispondere alle domande previste; • Confermare le risposte date al questionario; • Verificare il risultato del questionario.	*Non Implementato*
TVFD7.1	L'utente autenticato intende creare una nuova domanda tramite procedura guidata. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Premere il pulsante Crea nuova domanda; • Inserire i dati necessari alla creazione di una nuova domanda; • Premere il pulsante di conferma creazione nuova domanda; • Verificare che sia stato creata la nuova domanda.	*Non Implementato*
TVFD7.2	L'utente autenticato intende modificare una domanda esistente tramite procedura guidata. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Premere il pulsante modifica domanda; • Modificare i dati della domanda selezionata; • Premere il pulsante di conferma modifica; • Verificare che sia stata modificata la domanda.	*Non Implementato*

Id Test	Descrizione	Stato
TVFO7.3	L'utente autenticato intende creare una nuova domanda tramite QML. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Premere il pulsante Crea nuova domanda; • Inserire i dati necessari alla creazione di una nuova domanda; • Premere il pulsante di conferma creazione nuova domanda; • Verificare che sia stato creata la nuova domanda.	*Non Implementato*
TVFD7.4	L'utente autenticato intende modificare una domanda esistente tramite QML. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Premere il pulsante modifica domanda; • Modificare i dati della domanda selezionata; • Premere il pulsante di conferma modifica; • Verificare che sia stata modifcata la domanda.	*Non Implementato*
TVFD8.1	L'utente autenticato intende visualizzare i questionari creati. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Verificare che vengano visualizzati tutti i questionari creati	*Non Implementato*

Id Test	Descrizione	Stato
TVFD8.2	L'utente autenticato pro intende modificare un questionario esistente. All'utente è richiesto di: • Essere autenticato pro; • Trovarsi nella sezione apposita; • Selezionare il questionario da modificare tra quelli creati; • Dalla sezione di modifica effettuare tutte le modifiche consentite dal sistema • Premere il pulsante di conferma modifiche; • Verificare che sia stato modificato il questionario correttamente.	*Non Implementato*
TVFD8.3	L'utente autenticato pro intende eliminare un questionario esistente. All'utente è richiesto di: • Essere autenticato pro; • Trovarsi nella sezione apposita; • Selezionare un questionario esistente da eliminare; • Premere il tasto di conferma eliminazione; • Verificare l'operazione appena effettuata.	*Non Implementato*
TVFD8.4	L'utente autenticato pro intende controllare i risultati degli esaminandi di un suo questionario. All'utente è richiesto di: • Essere autenticato pro; • Trovarsi nella sezione apposita; • Selezionare il questionario da controllare gli esiti; • Verificare che si ci siano gli esiti di tutti gli utenti iscritti.	*Non Implementato*

Id Test	Descrizione	Stato
TVFD8.5	L'utente autenticato pro intende attivare la modalità esame di un suo questionario. All'utente è richiesto di: • Essere autenticato pro; • Trovarsi nella sezione apposita; • Selezionare il questionario da attivare; • Verificare l'operazione effettuata.	*Non Implementato*
TVFO8.6	L'utente autenticato pro intende creare un nuovo questionario. All'utente è richiesto di: • Essere autenticato pro; • Trovarsi nella sezione apposita; • Premere il pulsante Crea nuovo questionario; • Inserire il titolo del nuovo questionario; • Inserire l'argomento del nuovo questionario; • Ricercare e selezionare le domande che andranno a comporre il questionario; • Premere il pulsante di conferma creazione nuovo questionario; • Verificare che sia stato creato il nuovo questionario.	*Non Implementato*
TVFO8.7	L'utente autenticato pro intende gestire le domande del questionario. All'utente è richiesto di: • Essere autenticato pro; • Trovarsi nella sezione apposita; • Premere il pulsante per gestire il questionario; • Effettuare le operazioni consentite; • Verificare che le operazioni siano effettivamente avvenute.	*Non Implementato*

Id Test	Descrizione	Stato
TVFD8.8	L'utente autenticato pro intende gestire le iscrizione di un suo questionario. All'utente è richiesto di: • Essere autenticato pro; • Trovarsi nella sezione apposita; • Selezionare il questionario da gestire le iscrizioni; • Effettuare le operazioni consentite; • Verificare le operazioni apportate.	*Non Implementato*
TVFO9	L'utente intende esercitarsi effettuando un allenamento. All'utente è richiesto di: • Trovarsi nella sezione apposita; • Selezionare gli appositi filtri per focalizzare l'allenamento che si vuole effettuare; • Premere il pulsante per iniziare l'allenamento. • Rispondere alle domande proposte e controllare l'esito delle risposte date.	*Non Implementato*
TVFD10	L'utente autenticato intende visualizzare il proprio profilo. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Visualizzare il proprio profilo.	*Non Implementato*
TVFD10.4	L'utente autenticato intende visualizzare la cronologia dei questionari svolti. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Visualizzare la cronologia dei questionari svolti.	*Non Implementato*

Id Test	Descrizione	Stato
TVFD10.4.1	L'utente autenticato intende visualizzare il riepilogo di un questionario svolto. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Selezionare il questionario svolto da visualizzare il riepilogo.	*Non Implementato*
TVFO11	L'utente intende rispondere alle domande. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Rispondere alle domande; • Verificare la funzionalità dell'operazione.	*Non Implementato*
TVFO12	L'utente autenticato intende ricercare un utente. All'utente è richiesto di: • Essere autenticato; • Trovarsi nella sezione apposita; • Inserire le keywords nella barra di ricerca; • Premere l'apposito tasto per effettuare la ricerca; • Verificare l'operazione appena effettuata.	*Non Implementato*

Tabella 1: Test di Validazione

4.1.2 Test di Sistema

Con questa tipologia di test si vuole verificare il corretto funzionamento delle componenti architetturali.
I test di sistema saranno descritti nel modo seguente:

TS[*TipoRequisito*][*ImportanzaRequisito*][*IdRequisito*]

dove:

- **TipoRequisito** può assumere valori tra:
  - *F* per i requisiti funzionali;
  - *Q* per i requisiti di qualità;
  - *V* per i requisiti di vincolo;
  - *P* per i requisiti prestazionali.
- **ImportanzaRequisito** può assumere valori tra:
  - *D* per i requisiti desiderabili;
  - *O* per i requisiti di obbligatori;
  - *F* per i requisiti di facoltativi.
- **IdRequisito** può assumere un valore gerarchico che identifica il singolo requisito.

4.1.3 Test di Integrazione

Con questa tipologia di test si vuole determinare il corretto funzionamento delle componenti progettate durante la definizione dell'architettura ad alto livello.
I test di integrazione saranno descritti nel modo seguente:

TI[*IdComponente*]

dove:

- **IdComponente** rappresenta il codice identificativo crescente del componente considerato.

È stato scelto di utilizzare un approccio top-down nel determinare i test di integrazione. Di seguito viene riportato un diagramma informale per rendere chiara la struttura dei test identificati.

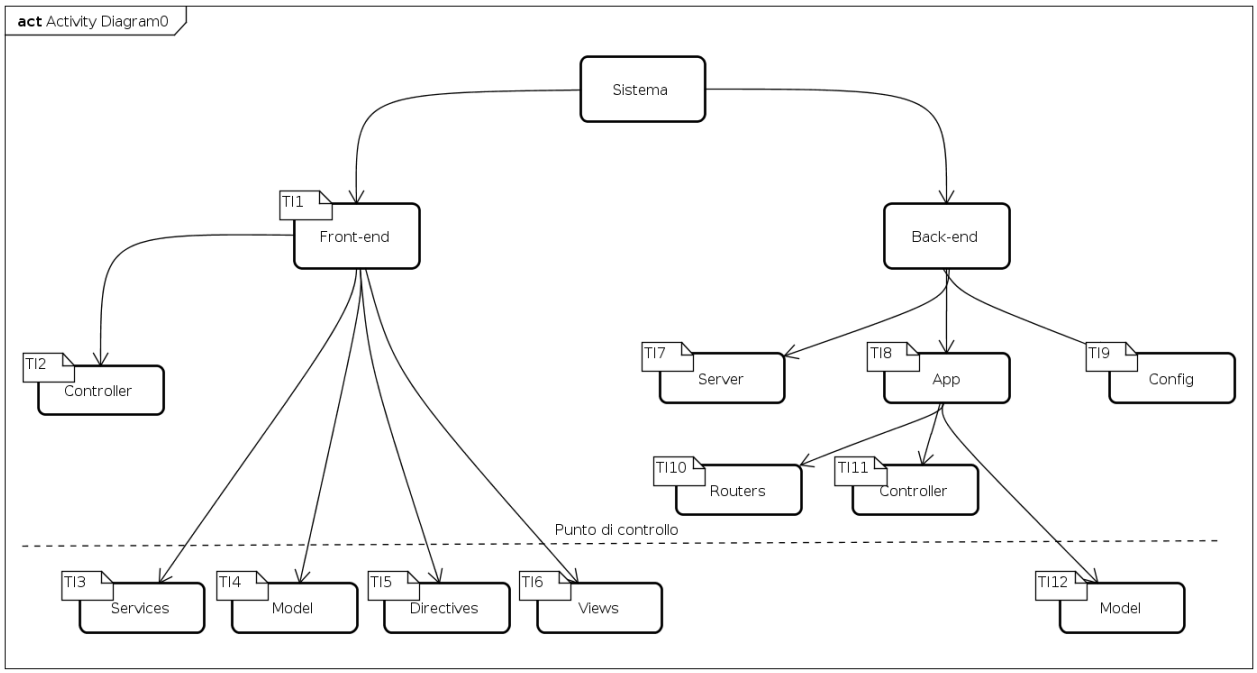

Figura 1: Albero di integrazione delle componenti

Nell'approccio top-down dei test di integrazione i moduli di livello più alto vengono sottoposti a test e integrati per primi. Così facendo anche la logica di alto livello e il flusso di dati vengono sottoposti a test fin da subito; sarà perciò necessario simulare le componenti di livello più basso con degli stub. Una volta codificate, le componenti di più basso livello dovranno a loro volta essere integrate e testate. L'approccio top-down rientra tra le strategie di integrazione incrementali, che conferiscono il vantaggio di poter determinare in modo più immediato quale componente causa problemi: i difetti rilevati dai test, infatti, nella maggioranza dei casi saranno da attribuirsi all'ultima componente aggiunta.

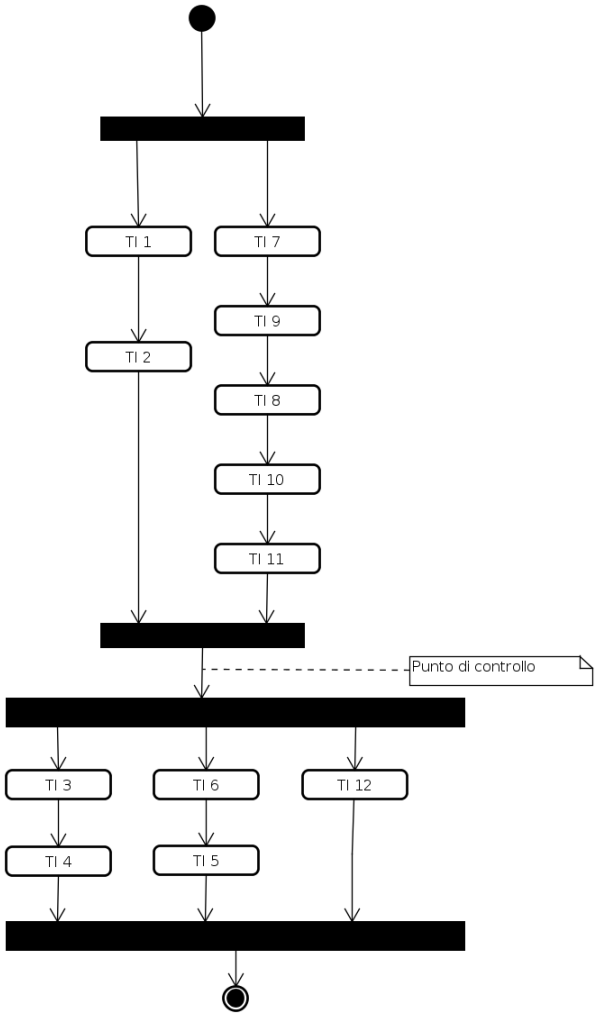

Figura 2: Diagramma di attività dei test di integrazione

Id Test	Descrizione	Stato
TI1	Viene verificato che l'applicazione Web gestisca correttamente il Front-End del prodotto e le sue interazioni con il Back-End.	*Non Implementato*
TI2	Viene verificato che i Controller del Front-End si integrino correttamente nell'applicazione Web.	*Non Implementato*
TI3	Viene verificato che i Services permettano di interagire correttamente con il Back-End.	*Non Implementato*
TI4	Viene verificato che il Model si integri correttamente con i Services e con le componenti dell'applicazione che lo utilizzano.	*Non Implementato*
TI5	Viene verificato che le Directives si integrino correttamente con le View.	*Non Implementato*
TI6	Viene verificato che le View si integrino correttamente con i Controller e che visualizzino in modo corretto i dati da essi ricevuti.	*Non Implementato*

Id Test	Descrizione	Stato
TI7	Viene verificato che il server si avvii correttamente, utilizzando Config per effettuare le configurazioni dell'applicazione, e che l'applicazione Web gestisca correttamente il Back-End del prodotto in modo tale da fornire al Front-End tutte le informazioni richieste.	*Non Implementato*
TI8	Viene verificato che App si integri correttamente con le librerie di Node.js utilizzate.	*Non Implementato*
TI9	Viene verificato che Config si integri con Server, carichi correttamente tutte le librerie per Node.js che utilizzerà e che istanzi le classi del package App in modo corretto.	*Non Implementato*
TI10	Viene verificato che Config si integri con Server, carichi correttamente tutte le librerie per Node.js che utilizzerà e che istanzi le classi del package App in modo corretto.	*Non Implementato*
TI11	Viene verificato che i Controller si integrino correttamente e gestiscano le richieste inoltrate dai Routers.	*Non Implementato*
TI12	Viene verificato che il Model si integri correttamente con i Controller per la gestione dell'inserimento, della modifica e dell'eliminazione dei dati.	*Non Implementato*

Tabella 2: Test di Integrazione

4.1.4 Test di Unità

Con questa tipologia di test si vuole verificare il corretto funzionamento delle unità individuate durante la definizione dell'architettura ad alto livello. I test di unità saranno descritti nel modo seguente:

TU[IdTest]

dove:

- IdTest rappresenta il codice identificativo crescente dell'unità considerata.

5 Tracciamento dei test

5.1 Tracciamento Test di Validazione-Requisiti

Test	Requisito
TVFO1	RFO1
TVFO2	RFO2
TVFO3	RFO3
TVFD4	RFD4
TVFD4.3	RFD4.3
TVFD4.8	RFD4.8
TVFD4.9	RFD4.9

Test	Requisito
TVFD5	RFD5
TVFO6	RFO6
TVFD7.1	RFD7.1
TVFD7.2	RFD7.2
TVFO7.3	RFO7.3
TVFD7.4	RFD7.4
TVFD8.1	RFD8.1
TVFD8.2	RFD8.2
TVFD8.3	RFD8.3
TVFD8.4	RFD8.4
TVFD8.5	RFD8.5
TVFO8.6	RFO8.6
TVFO8.7	RFO8.7
TVFD8.8	RFD8.8
TVFO9	RFO9
TVFD10	RFD10
TVFD10.4	RFD10.4
TVFD10.4.1	RFD10.4.1
TVFO11	RFO11
TVFO12	RFO12

Tabella 3: Tracciamento Test di Validazione-Requisiti

5.2 Tracciamento Requisiti-Test di Validazione

Requisito	Test
RFO1	TVFO1
RFO2	TVFO2
RFO3	TVFO3
RFD4	TVFD4
RFD4.3	TVFD4.3
RFD4.8	TVFD4.8
RFD4.9	TVFD4.9
RFD5	TVFD5
RFO6	TVFO6
RFD7.1	TVFD7.1
RFD7.2	TVFD7.2
RFO7.3	TVFO7.3
RFD7.4	TVFD7.4
RFD8.1	TVFD8.1
RFD8.2	TVFD8.2
RFD8.3	TVFD8.3
RFD8.4	TVFD8.4
RFD8.5	TVFD8.5
RFO8.6	TVFO8.6
RFO8.7	TVFO8.7
RFD8.8	TVFD8.8
RFO9	TVFO9
RFD10	TVFD10

Requisito	Test
RFD10.4	TVFD10.4
RFD10.4.1	TVFD10.4.1
RFO11	TVFO11
RFO12	TVFO12

Tabella 4: Tracciamento Requisiti-Test di Validazione

5.3 Tracciamento Test di Integrazione-Componenti

Test	Componente
TI1	`Quizzipedia::Front-End`
TI2	`Quizzipedia::Front-End::Controllers`
TI3	`Quizzipedia::Front-End::Services`
TI4	`Quizzipedia::Front-End::Model`
TI5	`Quizzipedia::Front-End::Directives`
TI6	`Quizzipedia::Front-End::Views`
TI7	`Quizzipedia::Back-End`
TI8	`Quizzipedia::Back-End::App`
TI9	`Quizzipedia::Back-End::Config`
TI10	`Quizzipedia::Back-End::App::Routers`
TI11	`Quizzipedia::Back-End::App::Controller`
TI12	`Quizzipedia::Back-End::App::Model`

Tabella 5: Tracciamento Test di Integrazione-Componenti

5.4 Tracciamento Componenti-Test di Integrazione

Componente	Test
`Quizzipedia::Back-End`	TI7
`Quizzipedia::Back-End::App`	TI8
`Quizzipedia::Back-End::App::Controller`	TI11
`Quizzipedia::Back-End::App::Model`	TI12
`Quizzipedia::Back-End::App::Routers`	TI10
`Quizzipedia::Back-End::Config`	TI9
`Quizzipedia::Front-End`	TI1
`Quizzipedia::Front-End::Controllers`	TI2
`Quizzipedia::Front-End::Directives`	TI5
`Quizzipedia::Front-End::Model`	TI4
`Quizzipedia::Front-End::Services`	TI3
`Quizzipedia::Front-End::Views`	TI6

Tabella 6: Tracciamento Componenti-Test di Integrazione

A Resoconto attività di verifica

In questa sezione del documento vengono descritti e analizzati gli esiti delle attività di verifica svolte su tutti i documenti che vengono consegnati nelle varie revisioni di avanzamento del progetto.

A.1 Revisione dei Requisiti

A.1.1 Tracciamento casi d'uso e requisiti

Il *team*$_G$ ha deciso di utilizzare il software interno *DocumentsDB* in modo da facilitare il tracciamento sia delle relazioni fra casi d'uso e requisiti, sia delle relazioni fra requisiti e fonti.

A.1.2 Analisi statica dei documenti

L'analisi dei documenti mediante *Walkthrough*$_G$ ha portato all'individuazione di alcuni errori frequenti a partire dai quali è stata stilata una lista di controllo che è stata inserita all'interno dei Processi Organizzativi nelle *Norme di Progetto*. Grazie a questa sarà possibile applicare l'*Inspection*$_G$ per le future attività di verifica.

A.1.3 Esiti verifiche automatizzate

Vengono qui riportati gli esiti delle verifiche automatizzate per il calcolo dell'*indice Gulpease*$_G$, alle quali sono stati sottoposti tutti i documenti.

Documento	Indice Gulpease	Esito
Norme di Progetto	68	Superato
Studio di Fattibilità	56	Superato
Piano di Progetto	60	Superato
Piano di Qualifica	57	Superato
Analisi dei Requisiti	72	Superato
Glossario	48	Superato
Verbale Interno 2015-12-03	75	Superato
Verbale Interno 2015-12-11	74	Superato
Verbale Interno 2015-12-29	75	Superato
Verbale Interno 2016-01-12	75	Superato
Verbale Esterno 2016-01-11	76	Superato

Tabella 7: Resoconto verifiche automatizzate - Revisione dei Requisiti