Piano Di Qualifica

Informazioni sul documento

Titolo documento	Piano Di Qualifica
Versione attuale	v4.0.0
Data versione attuale	2012/04/02
Data creazione	2011/12/11
Redazione	Umberto Dall'Est
	Stefano Faoro
	Luca Lorenzini
	Giacomo Lorigiola
	Luca Guerra
Revisione	Umberto Dall'Est [v4.0.0]
	Umberto Dall'Est [v3.0.0]
	Giacomo Lorigiola [v2.0.0]
	Andrea Zironda [v1.0.0]
Approvazione	Andrea Zironda
Stato documento	Formale
Uso	Esterno
Distribuito da	SevenFold
Destinato a	SevenFold

Sommario

Il presente documento descrive la strategia di verifica e validazione seguita nello sviluppo del progetto.

Diario delle modifiche

Versione	Data	Autore	Modifiche
v4.0.0	2012/04/02	Andrea Zironda	Approvazione e rilascio quarta versione
v3.3.0	2012/04/03	Umberto Dall'Est	Aggiornati test sottosistema mobile e metriche
v3.2.0	2012/03/30	Luca Lorenzini	Correzione Test
v3.1.0	2012/03/28	Luca Lorenzini	Inseriti test Activity sottosistema Mobile
v3.0.0	2012/03/19	Luca Guerra	Approvazione e rilascio terza versione
v2.7.0	2012/03/17	Stefano Faoro	Inserita sezione metriche Server e aggiornate metriche Mobile e Desktop
v2.6.0	2012/03/15	Giacomo Lorigiola	Inseriti test sottosistema server
v2.5.0	2012/03/12	Stefano Faoro	Inserita sezione metriche mobile
v2.4.0	2012/03/09	Stefano Faoro	Inserita sezione metriche desktop
v2.3.0	2012/03/01	Umberto Dall'Est	Inseriti test sottosistema mobile
v2.2.0	2012/02/24	Stefano Faoro	Inserita la sezione riguardante la RP nel "Dettaglio dell'esito delle Revisioni"
v2.1.0	2012/02/10	Luca Guerra	Inserito capitolo "Obiettivi di qualità"
v2.0.0	2012/01/30	Umberto Dall'Est	Approvazione e rilascio seconda versione
v1.9.0	2012/01/30	Umberto Dall'Est	Correzione e ampliamento capitolo "Gestione amministrativa della revisione"
v1.8.0	2012/01/29	Umberto Dall'Est	Terminata analisi e stesura di tecniche, strumenti, metodi e test
v1.7.0	2012/01/27	Stefano Faoro	Inserita sezione "Pianificazione delle attività di test"
v1.6.0	2012/01/27	Umberto Dall'Est	Riscritta sezione "Visione generale della strategia di verifica"
v1.5.0	2012/01/27	Umberto Dall'Est	Cambiata parzialmente struttura documento
v1.4.0	2012/01/25	Luca Lorenzini	Aggiunto capitolo "Attività pianificata di test"
v1.3.0	2012/01/24	Luca Lorenzini	Aggiunto capitolo "Resoconto delle verifiche"
v1.2.0	2012/01/17	Stefano Faoro	Aggiunto capitolo "Esito delle revisioni"
v1.1.0	2012/01/16	Stefano Faoro	Aggiunta paragrafi "Analisi statica","Analisi dinamica" e aggiornato paragrafo 2.3.1
v1.0.0	2011/12/16	Antonio Pretto	Approvazione e rilascio prima versione
v0.3.1	2011/12/15	Luca Lorenzini	Migliore definizione del punti 3.1, 3.2 e 3.3
v0.3.0	2011/12/15	Giacomo Lorigiola	Aggiunto "Trattamento discrepanze"
v0.2.1	2011/12/14	Luca Lorenzini	Modificato punto "Comunicazione e risoluzione di anomalie"
v0.2.0	2011/12/12	Luca Lorenzini	Aggiunto punto "Metodi"
v0.1.0	2011/12/11	Luca Lorenzini	Creazione documento

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Con questo documento si intende garantire e gestire la qualità del prodotto software definendo la strategia generale di verifica e di validazione messa in atto per il progetto PMAC[g]. La prima versione del documento avrà valore contrattuale per la gara d'appalto, poi verrà costantemente aggiornato per coprire ogni necessità nello sviluppo del progetto.

1.2 Scopo del prodotto

Il sistema software PMAC si pone come obiettivo la realizzazione di una piattaforma innovativa per l'apprendimento comportamentale nell'ambito della sicurezza del lavoro, che utilizzi le tecniche della gamification[g] per incentivare il coinvolgimento e la partecipazione degli utenti e per scardinare l'instaurarsi di abitudini errate.

1.3 Glossario

Per evitare ridondanze tutti i termini e gli acronimi presenti nel seguente documento che necessitano di definizione saranno seguiti da una "g" ad apice (E.g. User[g]) alla loro prima occorrenza e saranno riportati in un documento esterno denominato *Glossario.pdf*.
Tale documento accompagna e completa il presente e consiste in un listato ordinato di termini e acronimi con le rispettive spiegazioni.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'appalto:
  `http://www.math.unipd.it/~tullio/IS-1/2011/Progetto/C3.pdf` PMAC
- Norme generali di progetto:
  vedi documento fornito in allegato *NormeDiProgetto_v4.pdf*
- Analisi dei requisiti:
  vedi documento fornito in allegato *AnalisiDeiRequisiti_v3.pdf*
- Piano di progetto:
  vedi documento fornito in allegato *PianoDiProgetto_v4.pdf*
- Estensione dei Requisiti:
  vedi documento fornito in allegato *EstensioneRequisiti_v1.pdf*

2 Visione generale della strategia di verifica

2.1 Organizzazione

Ogni modifica effettuata su di un documento, testo o codice che sia, ne cambia lo stato e in particolare ne contesta lo stato ”verificato” provocando lo stanziamento di un nuovo processo di verifica.
Questo processo, a cura di un verificatore, richiederà l'analisi del diario delle modifiche.
Nel caso di una modifica circoscritta il verificatore dovrà controllare solamente la sezione interessata dalla modifica, altrimenti il processo dovrà esaminare ogni sezione influenzata dalla suddetta.
Nell'eventualità che vengano riscontrate anomalie o errori si dovrà procedere con l'impostare lo stato del documento nuovamente a ”instabile” e con la creazione di una segnalazione tramite Ticket.
In caso contrario il documento verrà marcato come ”verificato”.

2.2 Pianificazione strategica e temporale

La pianificazione delle attività di verifica è incaricata al responsabile di progetto il quale deve definire e assegnare tali attività ai verificatori accompagnate dalle relative date di scadenza.
Dovrà inoltre controllarne ed approvarne gli esiti.
Per un esame in dettaglio della distribuzione dei ruoli si veda il documento *Organigramma.pdf*.
Le principali revisioni sono previste nelle seguenti date:

- (RR) Revisione dei Requisiti: 2012/01/10
- (RP) Revisione di Progettazione: 2012/02/06
- (RQ) Revisione di Qualifica: 2012/03/21
- (RA) Revisione di Accettazione: 2012/04/05

2.3 Responsabilità

Per un esame in dettaglio dei ruoli e delle relative responsabilità si veda il documento *PianoDiProgetto.pdf*.

2.4 Risorse

Questo paragrafo ha lo scopo di illustrare in breve le risorse richieste dalle attività di verifica.

2.4.1 Risorse necessarie

Nei processi di revisione sono richieste diverse tipologie di risorse.
In primo luogo le risorse umane necessarie per revisionare i documenti.
Seguono le risorse software fra cui si richiedono strumenti per la valutazione delle metriche del codice, strumenti per la verifica del codice Android[g] e strumenti per la verifica del codice RoR[g].

2.4.2 Risorse disponibili

Tra le risorse umane si trovano il verificatore e il responsabile di progetto.
Seguono le risorse software fra cui si trovano Eclipse con alcuni plugin per la valutazione delle metriche del codice Android, la classe AndroidTestCase per i test di unità del codice Android, la gemma ”metric_fu” affiancata dal wrapper ”metrical” per la valutazione delle metriche del codice RoR, la gemma ”rails_best_practises” per verificare l'adesione alle best practises del codice RoR e il framework[g] Rspec per i test di unità del codice RoR.
Infine si richiede l'accesso esclusivo ai documenti da parte dei verificatori durante le fasi di verifica.

2.5 Strumenti

Le risorse software a disposizione dei verificatori sono i seguenti:

- **Apollo**^{g}: project manager utilizzato per assegnare le attività di verifica dal responsabile di progetto. Viene inoltre utilizzato dai verificatori per tener traccia dello stato del documento e per la segnalazione di anomalie o errori tramite Ticket.
- **GitHub**^{g}: utilizzato per segnalare un errore minore sul codice.
- **TexMaker**: IDE utilizzato per la redazione e la correzione ortografica dei documenti in LaTex^g.
- **Eclipse**: IDE utilizzato nella redazione, nell'analisi delle metriche e nei test di unità del codice Android.
- **Metrics**: plugin per Eclipse votato al calcolo delle metriche del codice Java.
- **CCCC**: tool votato al calcolo delle metriche del codice C++.
- **Gemma metric_fu**: gemma votata al calcolo delle metriche del codice RoR.
- **Gemma metrical**: gemma utilizzata per semplificare la gestione di "metric_fu".
- **AndroidTestCase**: principale classe dedicata ai test di unità del codice Android.
- **junit4**: framework dedicato ai test del codice java presente nel model del sottosistema mobile.
- **Mockito**: libreria per realizzare mock in ambiente java dalla sintassi immediata e intuitiva.
- **PowerMock**: framework che estende le potenzialità di Mockito.
- **Robolectric**: framework che permette di eseguire i test Android esternamente alla DVM.
- **Gemma rails_best_practises**: gemma che analizza il codice RoR e segnala la violazione delle best practise di RoR.
- **Gemma Rspec**: gemma utilizzata per effettuare i test di unità del codice RoR.
- **W3C**: si utilizzano gli strumenti di validazione del W3C per validare le pagine prodotte dal codice RoR.

2.6 Obiettivi di qualità

Nel corso della progettazione architetturale si è tenuti a rispettare i seguenti principi tramite i quali possiamo perseguire degli obiettivi di qualità:

- **Basso accoppiamento**: limitare al massimo l'intensità delle relazioni tra i moduli, in previsione di modifiche future e facilitando un eventuale lavoro di manutenzione.
- **Forte coesione**: all'interno di un singolo modulo è preferibile avere forte intensità di relazione tra i costituenti dello stesso,al fine di ottenere una buona caratterizzazione.
- **Decomposizione modulare**: identificare i componenti indipendenti al fine di avere una migliore suddivisione del lavoro di codifica e della pianificazione dei test.
- **Sufficienza e Completezza**:la definizione dell'astrazione fornisce tutte e sole le funzionalità e caratteristiche richieste alla stessa e null'altro di superfluo.
- **Incapsulazione**:le astrazioni definite devono essere separate dal dettaglio implementativo, in linea col principio di information hiding.
- **Atomicità**: l'astrazione è definita secondo un giusto livello di decomposizione, secondo il quale non sarebbe conveniente procedere in ulteriori astrazioni più primitive.

2.7 Tecniche

2.7.1 Analisi Statica

L'analisi statica verrà effettuata dai verificatori per tutta la durata del progetto su ogni documento prodotto e si applica con due diverse tecniche di lettura:

- **Walkthrough**: questa tecnica consiste nella lettura critica da parte del verificatore di tutto il contenuto del documento in analisi.
  Risulta essere molto onerosa in termini di tempo e risorse, ma è inevitabile ricorrervi soprattutto nella fase di correzione dei documenti per effettuarne la correzione ortografica e strutturale.
  Ogni errore riscontrato verrà discusso con l'autore del documento per evitare di incorrere in incomprensioni.
  La tecnica di walkthrough sarà molto utilizzata nelle fasi iniziali del ciclo di vita e prima del rilascio di ogni documento.
  Con l'acquisizione di esperienza da parte dei componenti del gruppo attraverso la revisione dei documenti e le discussioni sugli errori più comuni riscontrati durante lo sviluppo del progetto ci si affiderà sempre di più alla tecnica di inspection.
  Il gruppo sarà quindi tenuto a tenere traccia delle problematiche rilevate più spesso durante le fasi di verifica.

- **Inspection**: questa tecnica consiste nella lettura mirata dei documenti incentrata sul controllo dei punti più soggetti ad errori. Durante le fasi di verifica del progetto l'inspection verrà preferita al walthrough in quanto richiede meno risorse.

2.7.2 Analisi Dinamica

Durante lo sviluppo del progetto si seguirà un approccio TDD (Test Driven Development). La progettazione e l'implementazione delle classi di test sarà quindi precedente alla scrittura del codice del sistema e permetteranno lo svolgimento di test ripetibili.
Si prevedono le seguenti tipologie di test:

- **Test di unità**: si occupa della verifica di uno o più moduli che compongono un'unità software. Le statistiche riportano che di media i due terzi degli errori identificati nell'analisi dinamica vengono riscontrati grazie ai test di unità.
  L'esito del test è positivo se l'unità esaminata soddisfa tutti i requisiti per lei previsti.
  I principali strumenti utilizzati per questa tipologia di test saranno AndroidTestCase e Rspec.

- **Test di integrazione**: si occupa della verifica dei sottosistemi che si creano dall'unione di più unità del sistema che dovranno prima aver superato i test di unità per loro previsti. Oltre a venire identificati errori residui sul funzionamento interno delle unità, verranno inoltre controllati i risultati ottenuti dall'integrazione di queste ultime.
  I principali strumenti utilizzati per questa tipologia di test saranno AndroidTestCase e Rspec.

- **Test di sistema**: si occupa della verifica del comportamento del sistema completo rispetto ai requisiti software.
  I principali strumenti utilizzati per questa tipologia di test saranno valutati in seguito.
  Attualmente gli strumenti candidati sono Selenium per testare il codice RoR e i tool monkeyrunner e UI/Application Exerciser Monkey per testare il codice Android.

- **Test di regressione**: si occupano di verificare il corretto funzionamento del sistema al seguito di modifiche.
  Effettuata una modifica verranno quindi rieseguiti i test di unità sulle unità interessate, seguiti dai test di integrazione e al test di sistema.

- **Test di accettazione**: si occupa di sottoporre il sistema al proponente. Se costui riscontra il corretto funzionamento del prodotto e l'adesione a tutti i requisiti allora il sistema verrà rilasciato.

2.7.3 Misurazione delle metriche del codice

Per garantire una stesura lineare del codice ed avere quindi una maggiore leggibilità che aiuti poi la verifica ed il mantenimento del codice, per evitare che la sua complessità aumenti esponenzialmente con l'aggiunta di nuove funzionalità, ma soprattutto per garantire una soddisfacente efficienza prestazionale al nostro sistema, prevediamo la costante analisi delle metriche del codice. Compito del verificatore è quello di verificare che queste metriche rispettino dei valori accettabili. Una descrizione della metrica e un valore di riferimento ritenuto sufficiente per superare la verifica vengono esposti di seguito:

- **Numero di parametri di un metodo**: un numero di parametri formali troppo elevato potrebbe indicare che parte di essi, se correlati, potrebbero essere racchiusi in un'ulteriore classe aumentando la manutenibilità del codice.
  Si cercherà di limitare il numero dei parametri a 10.

- **Conteggio righe di codice di un metodo**: un numero di righe di codice troppo elevato all'interno di un metodo potrebbe indicare che parte delle sue funzionalità potrebbero venir suddivise in ulteriori metodi aumentando così la verificabilità e la manutenibilità del codice.
  Si cercherà di limitare il numero di linee di codice a 100.

- **Complessità ciclomatica di un metodo**: è il valore che indica la complessità strutturale del codice. Viene definita calcolando il numero di percorsi di codice linearmente indipendenti nel flusso del programma(e.g. blocchi if, cicli for, istruzioni switch, etc.) aggiungendo 1 al totale.
  Un numero troppo elevato riduce la riusabilità e la manutenibilità del codice.
  Si cercherà di limitare l'indice di complessità ciclomatica a 10.

- **Livello di annidamento di un metodo**: un livello di annidamento troppo elevato aumenta la complessità del codice rendendolo difficilmente verificabile e mantenibile.
  Si cercherà di limitare il livello di annidamento a 5.

- **Numero di campi dati di una classe**: un numero di attributi troppo elevato potrebbe indicare che parte di essi, se correlati, potrebbero essere racchiusi in un'ulteriore classe aumentando la verificabilità, la manutenibilità e l'astrazione del codice.
  Si cercherà di limitare il numero dei parametri raggruppandoli dove possibile.

- **Peso di una classe**: è la somma delle complessità ciclomatiche dei metodi contenuti in essa. Un valore troppo elevato potrebbe indicare che parte delle funzionalità offerte dalla classe possano essere delegate ad altre classi.
  Per questa metrica non si prevede un indice di riferimento, ma ci si affiderà alla discrezione del verificatore nel ritenere questo indice non accettabile.

- **Grado di accoppiamento di una classe**: rappresenta la quantità di dipendenze esterne richieste per il corretto funzionamento di una classe.
  Seppure non si calcolerà e analizzerà questo indice il codice verrà redatto cercando di mantenere un basso grado di accoppiamento fra classi e package in modo da rendere il codice più indipendente, modulare, verificabile e mantenibile possibile.

2.8 Metodi

2.8.1 Ticketing

La procedura di segnalazione degli errori e dell'assegnazione delle attività di correzione è denominata ticketing.
Con i ticket un verificatore assegna la correzione di un errore riscontrato ad un progettista o a un programmatore che deve seguire le norme descritte al capitolo 4.2 del documento *NormeDiProgetto.pdf*
La creazione di un ticket può comunque essere eseguita indipendentemente dal ruolo che si ricopre, previa identificazione di un errore. In caso un ticket sia creato in modo errato si deve procedere contattando il creatore del ticket chiedendo delucidazioni a riguardo per evitare di perdere l'utilità dello stesso.
In generale la segnalazione di un errore non deve contenere indicazioni sull'entità della correzione, ad eccezione di anomalie la cui spiegazione risulterebbe particolarmente complicata se priva di essa.
La procedura di creazione dei ticket prevede che per il riscontro di più errori in un'unica sessione di verifica vengano aperti più ticket per avere un resoconto e uno storico più dettagliato.

3 Gestione amministrativa della revisione

3.1 Comunicazione e risoluzione di anomalie

Un'anomalia si presenta quando in seguito ad una revisione si riscontrano degli errori di programmazione o di stesura dei documenti che non riguardano la progettazione, ma sono nati nelle fasi successive.
Nel caso si trovi un'anomalia se ne deve comunicare la presenza al redattore del documento utilizzando il ticketing. Per maggiori informazioni sul metodo del ticketing consultare l'apposito paragrafo sezione metodi di questo documento.
Le varie tipologie di anomalie sono:

1. **Anomalie di contenuto:** un'anomalia di contenuto è uno scostamento dei contenuti di un documento da quelli previsti per lo stesso dall'analista e dal progettista.
2. **Anomalia formale:** un'anomalia formale è identificata ogni volta che si incontra un errore sintattico od ortografico nei documenti.
3. **Anomalie di codice:** un'anomalia di codice consiste in un errore logico o sintattico commesso nel codice.
4. **Anomalia strutturale:** un'anomalia strutturale consiste in un errore riscontrato nella struttura di un documento.

3.2 Trattamento discrepanze

Sarà considerata discrepanza uno scostamento del prodotto dalle attese del cliente e dello stesso gruppo SevenFold in base ai requisiti specificati nell'*AnalisiDeiRequisiti.pdf* e quindi dal prodotto atteso. Solitamente una discrepanza è attribuibile ad un errore di analisi o di progettazione.
A seguito del riscontro di una discrepanza si procederà con l'apertura di un apposito ticket che descriverà la discrepanza in dettaglio e che sarà assegnato al responsabile del progetto il quale avrà il compito di decidere sul da farsi.
Di norma tali discrepanze verranno riscontrate dai verificatori durante la loro normale attività di verifica, ma potranno essere sollevate anche da una qualsiasi figura del gruppo nel momento in cui questa si accorga della loro presenza all'interno del prodotto.
Il responsabile, analizzata la discrepanza, dovrà assegnare alla stessa un livello di gravità valutato sulla base dell'importanza del scostamento dalle attese, e sulla base dei costi e dei tempi necessari per risolverla.
Verrà quindi definito un piano correttivo che analizzerà in dettaglio come procedere e si procederà con l'assegnazione delle attività di correzione e successivamente con quelle di verifica.
A discrepanza risolta verrà effettuato un test di regressione per verificare la perfetta compatibilità delle modifiche apportate con il resto del sistema.
A test concluso, verrà chiuso il ticket di notifica della discrepanza.

3.3 Procedure di controllo di qualità di processo

Per assicurare la qualità di processo si delega al ruolo di responsabile di progetto la misurazione dei tempi, delle risorse e dei costi richiesti al completamento di un processo.
Supervisionando costantemente tutti i membri del gruppo di lavoro e i documenti da loro prodotti il responsabile monitora potenziali scostamenti dalle norme previste per il processo corrente e può intervenire riprendendo il diretto interessato dell'infrazione.
Il miglioramento continuo della qualità di processo è garantito dal tracciamento di ogni considerazione a suo riguardo da parte del responsabile che maturerà quindi nel tempo una maggiore esperienza a riguardo che potrà poi applicare alla successiva iterazione del modello incrementale.

4 Pianificazione delle attività di test

4.1 Test di Sistema

Con i test di sistema si andrà a verificare che tutti i requisiti funzionali presenti nel documento *AnalisiDeiRequisiti_v3.pdf* siano stati effettivamente implementati nel sistema prima del rilascio finale.
Ogni requisito sarà testato con una prova dinamica utilizzando le modalità e gli strumenti descritti per i test di sistema nel paragrafo 2.6.2 "Analisi Dinamica".
La notazione usata per i codici delle prove consta del codice del requisito preceduto da una "T".

Tabella 1: Tracciamento requisito - verifica

Requisito	Codice prova	Descrizione prova
Fo-1	TFo-1	Si verifica che il sistema inserisca correttamente il workgroup creato dal PMAC Administrator[g].
Fo-2.1.1	TFo-2.1.1	Si verifica che il sistema inserisca correttamente l'achievement[g] creato dal PMAC Administrator.
Fo-2.1.2	TFo-2.1.2	Si verifica che il sistema inserisca correttamente il badge[g] creato dal PMAC Administrator.
Fo-2.2	TFo-2.2	Si verifica che il sistema inserisca correttamente il nuovo profilo aziendale creato dal PMAC Administrator, salvando tutti i campi del profilo descritti dai sotto-requisiti di Fo-2.2 nell'Analisi dei Requisiti.
Fo-2.3	TFo-2.3	Si verifica che il sistema abbia salvato tutte le modifiche effettuate ai campi del profilo aziendale da parte del PMAC Administrator.
Fo-2.4	TFo-2.4	Si verifica che il sistema elimini correttamente il profilo aziendale selezionato dal PMAC Administrator.
Fo-2.5	TFo-2.5	Si verifica che all'invio di un ticket da parte di un Super-user[g] il sistema riceva effettivamente il ticket destinato al PMAC Administrator.
Fo-3.1	TFo-3.1	Si verifica che il sistema inserisca correttamente la nuova quest[g] creata dal PMAC Administrator, salvando tutti i dati della stessa descritti dai sotto-requisiti di Fo-3.1 nell'Analisi dei Requisiti. Nell'assenza di un campo il sistema deve rifiutare l'inserimento della quest e riproporre la schermata di inserimento.
Fo-3.2	TFo-3.2	Si verifica che il sistema associ correttamente una quest al workgroup selezionato dal PMAC Administrator.
Fo-3.3	TFo-3.3	Si verifica che il sistema elimini correttamente la quest specificata dal PMAC Administrator.
Fo-4.1	TFo-4.1	Si verifica che il sistema inserisca correttamente il nuovo User registrato dal Super-user salvando tutti i dati dello stesso descritti dai sotto-requisiti di Fo-3.1 nell'Analisi dei Requisiti. Il sistema dovrà rispondere in maniera negativa nel caso in cui l'indirizzo e-mail del nuovo User sia già occupata da un altro User.

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

Fo-4.2	TFo-4.2	Si verifica che il sistema abbia salvato tutte le modifiche effettuate ai campi del profilo dello User specificato dal Super-user.
Fo-4.3	TFo-4.3	Si verifica che il sistema elimini correttamente lo User selezionato dal Super-user.
Fo-4.4	TFo-4.4	Si verifica che il sistema visualizzi correttamente i profili e le statistiche dello User selezionato dal Super-user.
Fo-4.6	TFo-4.6	Si verifica che il sistema esegua correttamente la creazione del ticket da parte del Super-user e che lo invii correttamente.
Fo-5	TFo-5	Si verifica che il sistema permetta la risoluzione di tutte le tipologie di quest dedicate all'Desktop-user[g].
Fo-6	TFo-6	Si verifica che il sistema permetta la risoluzione di tutte le tipologie di quest al Mobile-user[g].
Fo-7	TFo-7	Si verifica che il sistema permetta l'autenticazione di uno User, rispondendo negativamente nel caso in cui lo User inserisca indirizzo e-mail o password errati.
Fo-8	TFo-8	Si verifica che il sistema notifichi a video la presenza di nuove quest inviate allo User.
Fo-8.1	TFo-8.1	Si verifica che il sistema di notifica permetta il collegamento diretto alla pagina di risoluzione delle quest ricevute dallo User.
Fo-9	TFo-9	Si verifica che il sistema permetta la visualizzazione delle classifiche aggiornate a seguito della richiesta da parte di uno User.
Fo-10.1	TFo-10.1	Si verifica che il sistema abbia salvato correttamente tutte le modifiche effettuate dallo User al proprio profilo.
Fo-10.2	TFo-10.2	Si verifica che il sistema permetta la visualizzazione del profilo personale a seguito della richiesta da parte di uno User.
Fo-10.3	TFo-10.3	Si verifica che il sistema permetta la visualizzazione delle statistiche dello User che le richiede, visualizzando tutti i dati descritti dai sotto-requisiti di Fo-10.3 nell'Analisi dei Requisiti.
Fd-1	TFd-1	Si verifica che il sistema permetta allo User la visualizzazione dei profili e delle statistiche di un qualsiasi User selezionato.
Fd-2	TFd-2	Si testa che il sistema visualizzi la breve nota esplicativa dopo la risoluzione di una quest.

4.2 Test di Integrazione

Tabella 2: Sottosistema Mobile - test di integrazione

Codice Test	Descrizione
TIM-LG	Verifica del corretto funzionamento congiunto delle classi view.Login, view.Vista, presenter.LoginP, presenter.Presenter, model.User, model.Resource, model.Session, model.Configuration e util.JParser al fine di garantire una corretta autenticazione di un utente nel sistema.
TIM-HM	Verifica del corretto funzionamento congiunto delle classi view.Home, view.Vista, presenter.HomeP, presenter.Presenter, model.User, model.Resource, model.Session, model.Configuration e util.JParser al fine di garantire la visualizzazione del profilo dell'utente loggato.
TIM-MP	Verifica del corretto funzionamento congiunto delle classi view.ModProfile, view.Vista, presenter.ModProfileP, presenter.Presenter, model.User, model.Resource, model.Session, model.Configuration e util.JParser al fine di garantire la modifica dei dati del profilo dell'utente loggato.
TIM-PQ	Verifica del corretto funzionamento congiunto delle classi view.PendingQuests, view.Vista, presenter.PendingQuestsP, presenter.Presenter, model.QuestHistory, model.Resource, model.Session, model.Configuration e util.JParser al fine di garantire la visualizzazione delle quest in sospeso dell'utente loggato.
TIM-PP	Verifica del corretto funzionamento congiunto delle classi view.PlayerProfile, view.Vista, presenter.PlayerProfileP, presenter.Presenter, model.User, model.Resource, model.Session, model.Configuration e util.JParser al fine di garantire la visualizzazione del profilo di un utente selezionato.
TIM-RK	Verifica del corretto funzionamento congiunto delle classi view.UsersRanks, view.WorkgroupsRanks, view.Vista, presenter.UsersRanksP, presenter.WorkgroupsRanksP, presenter.Presenter, model.UsersRank, model.WorkgroupsRank, model.Resource, model.Session, model.Configuration e util.JParser al fine di garantire la visualizzazione delle classifiche.
TIM-SQ	Verifica del corretto funzionamento congiunto delle classi view.SolveQuest, view.Vista, presenter.SolveQuestP, presenter.Presenter, model.Quest, model.Description, model.TextDescription, model.ImageDescription, model.Challenge, model.ComboChallenge, model.MultiOptionChallenge, model.TrueFalseChallenge, model.Resource, model.Session, model.Configuration e util.JParser al fine di garantire lo svolgimento di una quest.

5 Dettaglio delle verifiche tramite analisi

5.1 Revisione dei requisiti

Durante questa fase iniziale si è fatto largo utilizzo della tecnica del walkthrough prestando particolare attenzione a:

- Controllare la correttezza ortografica utilizzando TexMaker
- Controllare la correttezza sintattica del documento
- Controllare la correttezza dei diagrammi, delle immagini e dei contenuti del documento
- Controllare che il redattore del documento abbia effettivamente seguito le norme definite dal documento *NormeDiProgetto.pdf*

Per ogni modifica apportata in seguito alla revisione del documento si è fatto uso della tecnica dell'inspection.
Prima del rilascio del documento è stata effettuata un'ultima rilettura completa del documento. Ogni documento in uscita dalla fase RR è stato verificato, corretto secondo le modalità previste e le modifiche/correzioni effettuate sono state riportate nel "Diario delle modifiche" inserendo l'autore della modifica, la data, il soggetto della modifica e la versione del documento.

5.2 Revisione di progettazione

Come nella fase precedente si è fatto largo uso delle tecniche di analisi statica e dei suoi strumenti.
La maggiore esperienza maturata nella fase precedente ci ha permesso di fare maggiore uso della tecnica dell'inspection preferendola al walkthrough dove possibile per ridurre l'impiego delle risorse.
Buona parte del tempo è stata utilizzata per la correzione e l'ampliamento dei documenti della fase precedente. Per maggiori dettagli consultare "Dettaglio dell'esito delle revisioni".

5.3 Revisione di qualifica

In questa fase, oltre alle consuete attività di verifica della documentazione, è stata effettuata analisi statica sul codice sorgente tramite la misurazione delle metriche definite nella sezione 2.7.3.
Di seguito i dati rilevati da tali misurazioni.

5.3.1 Metriche sottosistema Server

Nella misurazione delle metriche del codice RoR sono sorti dei problemi di compatibilità tra gli strumenti candidati allo scopo (`metric_fu` e `metrical`) e le configurazioni del sistema.
Si cercherà di risolvere tali conflitti per poter effettuare le misurazioni definite in sezione 2.7.3 entro la prossima fase di progetto.
Allo stato attuale possono essere fornite delle metriche ricavate utilizzando il comando `rake stats` (`http://robots.thoughtbot.com/post/159806234/simple-test-metrics-in-your-rails-app-and-what-they`).
Le statistiche riportate sono:

- **LOC**: Linee di codice del componente.
- **Classes**: Numero di classi del componente.
- **Methods**: Numero di metodi del componente.
- **M/C**: Numero di metodi per classe.
- **LOC/M**: Linee di codice per metodo.

Tabella 3: Metriche sottosistema Server ottenute con `rake stats`

Name	**LOC**	**Classes**	**Methods**	**M/C**	**LOC/M**
Controllers	3582	42	268	6	11
Helpers	163	0	19	0	6
Models	595	39	61	1	7
Libraries	171	1	20	20	6
Model specs	260	0	0	0	0
View specs	127	0	0	0	0
Controller specs	650	0	0	0	0
Helper specs	156	0	0	0	0
Routing specs	868	0	0	0	0
Request specs	95	0	0	0	0
Cucumber features	84	0	0	0	0
Total	6751	82	368	4	16

- **Code LOC**: 4511
- **Test LOC**: 2240
- **Code to Test Ratio**: 1:0.5

5.3.2 Metriche sottosistema Desktop

Si riportano di seguito gli esiti dell'analisi statica effettuata sul codice del sottosistema Desktop.
Per il calcolo delle metriche è stato utilizzato il tool `CCCC` (C and C++ Code Counter, `http://sourceforge.net/projects/cccc/`).

Tabella 4: Metriche codice sottosistema Desktop

Metrica misurata	**Media**	**Massimo**
Numero di parametri di un metodo	0,345	2
Conteggio righe di codice di un metodo	9,914	70
Complessità ciclomatica di un metodo	1,224	**16**
Numero di campi dati di una classe	7,16	18

Dalla tabella si può notare come i valori rispecchino quelli prefissati nella sezione 2.7.3 tranne il valore massimo della complessità ciclomatica.
Questo valore si ottiene dal metodo `void afterAuthentication(int,int )` della classe `Manager`, metodo che riceve le risposte dal server e appunto deve decidere in base a queste risposte l'azione successiva da svolgere.
Il suo comportamento varia inoltre a seconda del caso in cui viene invocato, se all'istanziazione della classe o in seguito ad una azione dell'utente.
La molteplicità dei casi aumenta la complessità di questo metodo che rimane però l'unico ad avere un valore più elevato del limite imposto.

5.3.3 Metriche sottosistema Mobile

Si riportano di seguito gli esiti dell'analisi statica effettuata sul codice dell'applicativo Android. I risultati sono stati ottenuti utilizzando il plugin `Metrics` per Eclipse.

Tabella 5: Metriche codice applicativo Android

Metrica misurata	**Media**	**Massimo**
Numero di parametri di un metodo	0,64	5
Conteggio righe di codice di un metodo	5,23	96
Complessità ciclomatica di un metodo	1,971	**17**
Livello di annidamento di un metodo	1,84	6
Numero di campi dati di una classe	2,176	21

Dalla tabella si può notare come i valori rispecchino quelli prefissati nella sezione 2.7.3 tranne il valore massimo della complessità ciclomatica.
Il valore più elevato di tale metrica si ottiene dal metodo `void initialize(Properties properties)` della classe `com.woty.android.model.Configuration`, funzione con la quale vengono caricati i dati del file di configurazione del sistema contenente le stringhe per le connessioni con il server.
L'alto valore della complessità ciclomatica (17) di questo metodo è dato dalla necessità di verificare il corretto caricamento di tutti i 17 campi del file.
Un altro valore elevato è ottenuto da `Object callMethod(String method, Object[] arguments)` della classe `com.woty.android.model.Resource`, funzione con la quale vengono effettuate chiamate a generici metodi `xml-rpc` del server.
L'alto valore della complessità ciclomatica (13) di questo metodo è dato dalla necessità di uniformarsi con le varie possibilità di chiamate a questi metodi a seconda dei parametri da passare.
Queste funzioni rimangono comunque le uniche ad avere un valore più elevato del limite imposto.

6 Dettaglio delle verifiche tramite prove

6.1 Sottosistema Server

Per ogni componente del sottosistema server viene presentata una tabella raffigurante il modulo da testare, il corrispettivo modulo di test, e l'esito del test.

6.1.1 Componente Controllers

Tabella 6: Sottosistema Server - test di unità

Modulo	Modulo di test	Esito
achievements_controller	achievements_controller_spec	Positivo
admins_controller	admins_controller_spec	Positivo
application_controller	application_controller_spec	Positivo
combo_challenges_controller	combo_challenges_controller_spec	Positivo
comments_controller	comments_controller_spec	Positivo
completed_achievementes_controller	completed_achievementes_controller_spec	Positivo
conditions_controller_controller	conditions_controller_controller_spec	Positivo
contact_modules_controller	contact_modules_controller_spec	Positivo
custom_quests_controller	custom_quests_controller_spec	Positivo
customers_controller_controller	customers_controller_controller_spec	Positivo
desktop_users_controller	desktop_users_controller_spec	Positivo
image_descriptions_controller	image_descriptions_controller_spec	Positivo
likes_controller	likes_controller_spec	Positivo
mobile_users_controller	mobile_users_controller_spec	Positivo
multi_option_challenges_controller	multi_option_challenges_controller_spec	Positivo
pages_controller_controller	pages_controller_controller_spec	Positivo
plans_controller	plans_controller_spec	Positivo
quest_associations_controller	quest_associations_controller_spec	Positivo
quest_categories_controller	quest_categories_controller_spec	Positivo
quest_challenges_controller	quest_challenges_controller_spec	Positivo
quest_descriptions_controller	quest_descriptions_controller_spec	Positivo
quest_histories_controller	quest_histories_controller_spec	Positivo
quests_controller	quests_controller_spec	Positivo
shared_quests_controller	shared_quests_controller_spec	Positivo
sessions_controller	sessions_controller_spec	Positivo

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

stream_elements_controller	stream_elements_controller_spec	Positivo
super_users_controller	super_users_controller_spec	Positivo
text_descriptions_controller	text_descriptions_controller_spec	Positivo
ticket_controller	ticket_controller_spec	Positivo
true_false_challenges_controller	true_false_challenges_controller_spec	Positivo
users_controller	users_controller_spec	Positivo
wgcategories_controller	wgcategories_controller_spec	Positivo
workgroups_controller	workgroups_controller_spec	Positivo

6.1.2 Componente Decorators

Tabella 7: Sottosistema Server - test di unità

Modulo	**Modulo di test**	**Esito**
achievement_decorator	achievement_decorator_spec	Positivo
application_decorator	application_decorator_spec	Positivo
condition_decorator	condition_decorator_spec	Positivo
customer_decorator	customer_decorator_spec	Positivo
plan_decorator	plan_decorator_spec	Positivo
stream_element_controller	stream_element_controller_spec	Positivo
user_decoator	user_decoator_spec	Positivo
workgroup_decoator	workgroup_decoator_spec	Positivo

6.1.3 Componente Helpers

Tabella 8: Sottosistema Server - test di unità

Modulo	**Modulo di test**	**Esito**
achievement_helper	achievement_helper_spec	Positivo
admins_helper	admins_helper_spec	Positivo
combo_challenges_helper	combo_challenges_helper_spec	Positivo
comments_helper	comments_helper_spec	Positivo
completed_achievements_helper	completed_achievements_helper_spec	Positivo
conditions_helper	conditions_helper_spec	Positivo

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

contact_modules_helper	contact_modules_helper_spec	Positivo
custom_quests_helper	custom_quests_helper_spec	Positivo
customers_helper	customers_helper_spec	Positivo
desktop_users_helper	desktop_users_helper_spec	Positivo
images_descriptions_helper	images_descriptions_helper_spec	Positivo
likes_helper	likes_helper_spec	Positivo
mobile_users_helper	mobile_users_helper_spec	Positivo
multi_option_challenges_helper	multi_option_challenges_helper_spec	Positivo
pages_helper	pages_helper_spec	Positivo
plans_helper	plans_helper_spec	Positivo
quest_associations_helper	quest_associations_helper_spec	Positivo
quest_categories_helper	quest_categories_helper_spec	Positivo
quest_challenges_helper	quest_challenges_helper_spec	Positivo
quest_description_helper	quest_description_helper_spec	Positivo
quest_histories_helper	quest_histories_helper_spec	Positivo
quest_helper_spec	quest_helper_spec	Positivo
shared_quest_helper	shared_quest_helper_spec	Positivo
stream_elements_helper	stream_elements_helper_spec	Positivo
super_users_helper	super_users_helper_spec	Positivo
text_descriptions_helper	text_descriptions_helper_spec	Positivo
tickets_helper	tickets_helper_spec	Positivo
true_false_challenges_helper	true_false_challenges_helper_spec	Positivo
wg_ranking_helper	wg_ranking_helper_spec	Positivo
wgcategories_helper	wgcategories_helper_spec	Positivo
workgroups_helper	workgroups_helper_spec	Positivo

6.1.4 Componente Models

Tabella 9: Sottosistema Server - test di unità

Modulo	**Modulo di test**	**Esito**
achievements	achievements_spec	Positivo
admin	admin_spec	Positivo

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

combo_challenge	combo_challenge_spec	Positivo
comment	comment_spec	Positivo
completed_achievement_element	completed_achievement_element_spec	Positivo
completed_achievement	completed_achievement_spec	Positivo
condition	condition_spec	Positivo
contact_module	contact_module_spec	Positivo
custom_quest	custom_quest_spec	Positivo
customer	customer_spec	Positivo
desktop_user	desktop_user_spec	Positivo
image_description	image_description_spec	Positivo
like	like_spec	Positivo
mobile_user	mobile_user_spec	Positivo
multi_option_challenge	multi_option_challenge_spec	Positivo
plan	plan_spec	Positivo
quest_association	quest_association_spec	Positivo
quest_category	quest_category_spec	Positivo
quest_challenge	quest_challenge_spec	Positivo
quest_description	quest_description_spec	Positivo
quest_history	quest_history_spec	Positivo
quest	quest_spec	Positivo
shared_quest	shared_quest_spec	Positivo
signup_element	signup_element_spec	Positivo
status_change_element	status_change_element_spec	Positivo
stream_element	stream_element_spec	Positivo
super_user	super_user_spec	Positivo
text_description	text_description_spec	Positivo
ticket	ticket_spec	Positivo
true_false_challenge	true_false_challenge_spec	Positivo
user	user_spec	Positivo
video_description	video_description_spec	Positivo
wgcategory	wgcategory_spec	Positivo
workgroup	workgroup_spec	Positivo

6.1.5 Componente Requests

Tabella 10: Sottosistema Server - test di unità

Modulo	Modulo di test	Esito
achievements	achievements_spec	Positivo
admins	admins_spec	Positivo
combo_challenges	combo_challenges_spec	Positivo
comments	comments_spec	Positivo
completed_achievements	completed_achievements_spec	Positivo
conditions	conditions_spec	Positivo
contact_modules	contact_modules_spec	Positivo
custom_quest	custom_quest_spec	Positivo
customers	customers_spec	Positivo
desktop_user	desktop_user_spec	Positivo
image_descriptions	image_descriptions_spec	Positivo
likes	likes_spec	Positivo
mobile_users_spec	mobile_users_spec	Positivo
plan	plan_spec	Positivo
quest_associations	quest_associations_spec	Positivo
quest_categories	quest_categories_spec	Positivo
quest_challenges	quest_challenges_spec	Positivo
quest_descriptions	quest_descriptions_spec	Positivo
quest_histories	quest_histories_spec	Positivo
quest	quest_spec	Positivo
shared_quest	shared_quest_spec	Positivo
stream_elements	stream_elements_spec	Positivo
super_users	super_users_spec	Positivo
true_false_challenges	true_false_challenges_spec	Positivo
users	users_spec	Positivo
workgroups	workgroups_spec	Positivo

6.1.6 Componente Routing

Tabella 11: Sottosistema Server - test di unità

Modulo	**Modulo di test**	**Esito**
achievements_routing	achievements_routing_spec	Positivo
admins_routing	admins_routing_spec	Positivo
combo_challenges_routing	combo_challenges_routing_spec	Positivo
comments_routing	comments_routing_spec	Positivo
completed_achievements_routing	completed_achievements_routing_spec	Positivo
conditions_routing	conditions_routing_spec	Positivo
contact_modules_routing	contact_modules_routing_spec	Positivo
custom_quests	custom_quests_spec	Positivo
customer_routing	customer_routing_spec	Positivo
desktop_users_routing	desktop_users_routing_spec	Positivo
image_descriptions_routing	image_descriptions_routing_spec	Positivo
likes_routing	likes_routing_spec	Positivo
mobile_users_routing	mobile_users_routing_spec	Positivo
plans_routing	plans_routing_spec	Positivo
quest_association_routing	quest_association_routing_spec	Positivo
quest_categories_routing	quest_categories_routing_spec	Positivo
quest_challenges_routing	quest_challenges_routing_spec	Positivo
quest_description_routing	quest_description_routing_spec	Positivo
quest_histories_routing	quest_histories_routing_spec	Positivo
quests_routing	quests_routing_spec	Positivo
stream_elements_routing	stream_elements_routing_spec	Positivo
super_users_routing	super_users_routing_spec	Positivo
ticket_routing	ticket_routing_spec	Positivo
true_false_challenges_routing	true_false_challenges_routing_spec	Positivo
true_false_challenges_routing	true_false_challenges_routing_spec	Positivo
users_routing	users_routing_spec	Positivo
workgroups_routing	workgroups_routing_spec	Positivo

6.1.7 Componente View

Tabella 12: Sottosistema Server - test di unità

Modulo	**Modulo di test**	**Esito**
achievements	achievements_spec	Positivo
admins	admins_spec	Positivo
combo_challenges	combo_challenges_spec	Positivo
comments	comments_spec	Positivo
completed_achievements	completed_achievements_spec	Positivo
conditions	conditions_spec	Positivo
contact_modules	contact_modules_spec	Positivo
custom_quest	custom_quest_spec	Positivo
customers	customers_spec	Positivo
desktop_user	desktop_user_spec	Positivo
image_descriptions	image_descriptions_spec	Positivo
likes	likes_spec	Positivo
mobile_users	mobile_users_spec	Positivo
plans	plans_spec	Positivo
quest_association	quest_association_spec	Positivo
quest_categories	quest_categories_spec	Positivo
quest_challenges	quest_challenges_spec	Positivo
quest_descriptions	quest_descriptions_spec	Positivo
quest_histories	quest_histories_spec	Positivo
quest	quest_spec	Positivo
shared_quests	shared_quests_spec	Positivo
stream_elements	stream_elements_spec	Positivo
super_users	super_users_spec	Positivo
tickets	tickets_spec	Positivo
true_false_challenges	true_false_challenges_spec	Positivo
users	users_spec	Positivo
workgroups	workgroups_spec	Positivo

6.2 Sottosistema Desktop

Tabella 13: Sottosistema Desktop - test di unità

Componente	**Classe di test**	**Classe testata**	**Esito Prova**
Manager	ManagerTest	Manager	Positivo
	ExceptionTest	Exception	Positivo
Client	ClientTest	Client	Positivo
Account	AccountTest	Account	Positivo
	PreferencesTest	Preferences	Positivo
	LoginWindowTest	LoginWindow	Positivo

6.3 Sottosistema Mobile

6.3.1 Test di unità

Per testare il componente di modello del sottosistema mobile sono stati combinati diversi framework per rendere la fase di test più immediata e intuitiva.
Utilizzando Robolectric si è riusciti ad emulare la DVM(Dalvik virtual machine) di Android nella JVM di java riducendo così il tempo richiesto dall'esecuzione dei test da svariati minuti a pochi secondi.
Si è quindi scelto di eseguire i test con junit4 con il supporto del framework Mockito per la realizzazione di mock. Quest'ultimo è stato scelto per la semplicità, la compattezza e la leggibilità della sua sintassi.
Infine sono state supplite le carenze di Mockito con PowerMock, rendendo possibile il test di classi astratte, metodi statici e metodi privati.

Per la parte della View del design pattern MVP, è stato utilizzato il framework Robotium che grazie al suo supporto di activity e menù ha reso possibile tastare a fondo l'interfaccia grafica dell'applicazione.

In particolare per ogni Activity, quindi per ogni schermata, sono stati testati diversi aspetti, quali:

- Struttura: tutti gli elementi grafici devono essere in uno stato iniziale conosciuto a priori, per esempio il campo di testo password nella schermata login deve essere vuoto finché l'utente non lo modifica.
- Comportamento: si verifica che come risposta ad una azione dell'utente, il sistema si comporti in modo corretto, ad esempio alla pressione di un bottone che, se le condizioni lo permettono, mostra una determinata schermata, si verifica che la schermata mostrata sia effettivamente quella desiderata.

Componente com.woty.android.view

Tabella 14: Riassunto test di unità componente View - Mobile

Classe di test	Classe testata	Esito della prova
`HomeTest`	`Home`	Positivo
`LoginTestRight`	`Login`	Positivo
`LoginTestWrong`	`Login`	Positivo
`ModProfileTestPre`	`ModProfile`	Positivo
`PendingQuestTestUI`	`PendingQuest`	Positivo
`PlayerProfileTestPre`	`PlayerProfile`	Positivo
`UsersRanksTestPre`	`UsersRanks`	Positivo
`UsersRanksTestUi`	`UsersRanks`	Positivo
`WorkgroupsRanksTestPre`	`WorkgroupsRanks`	Positivo
`WorkgroupsRanksTestUi`	`WorkgroupsRanks`	Positivo

- **Classe HomeTest**

  Descrizione:

  - Struttura: si verifica che tutte le `TextView`, cioè le label che descrivono i vari campi del profilo, presentino le corrette diciture.
  - Comportamento: si testa che alla pressione dei bottoni il sistema si comporti correttamente, quindi rispettivamente mostrino le schermate di quest, rank, quella per la modifica della password e quella adibita alla modifica del profilo dell'utente autenticato.

- **Classe LoginTestRight**

  Descrizione:

  - Struttura: viene testato che le `TextEdit` siano vuote all'avvio dell'applicazione e che gestiscano correttamente l'inserimento delle credenziali.
  - Comportamento: si verifica che alla pressione del bottone per accedere al sistema, i dati corretti per il login vengano trasmessi al server e che la schermata visualizzata sia effettivamente la schermata Home.

  Dettagli: Per poter eseguire correttamente questo test il sistema deve necessariamente essere nello stato iniziale di logout, altrimenti l'applicazione tramite l'auto-login, con le credenziali inserite in precedenza, salta la schermata di login e il test non può essere svolto.

- **Classe LoginTestWrong**

  Descrizione:

  - Struttura: viene testato che le `TextEdit` siano vuote all'avvio dell'applicazione e che gestiscano correttamente l'inserimento delle credenziali.
  - Comportamento: si verifica che alla pressione del bottone per accedere al sistema i dati sbagliati per il login vengano trasmessi e che la schermata visualizzata sia invariata, quindi la schermata di login.

  Dettagli: Per poter eseguire correttamente questo test il sistema deve necessariamente essere nello stato iniziale di logout, altrimenti l'applicazione tramite l'auto-login, con le credenziali inserite in precedenza, salta la schermata di login e il test non può essere svolto.

- **Classe ModProfileTestPre**

  Descrizione:

  - Struttura: si verifica che tutte le `TextView`, cioè le label che descrivono i vari campi modificabili e il bottone per la conferma delle modifiche apportate presentino le corrette diciture.

- **Classe PendingQuestTestUI**

  Descrizione:

  - Comportamento: si verifica che dalla schermata di visualizzazione delle quest da svolgere, selezionando una quest, venga effettivamente mostrata la schermata per affrontare la quest.

  Dettagli: Per poter eseguire correttamente questo test l'utente autenticato nel sistema deve avere almeno una quest da svolgere.

- **PlayerProfileTestPre**

  Descrizione:

  - Struttura: si verifica che tutte le `TextView`, cioè le label che descrivono i vari campi del profilo, presentino le corrette diciture rispetto alla lingua selezionata nel dispositivo.
  - Comportamento: questa classe non offre funzionalità ulteriori rispetto la semplice visualizzazione di un utente.

- **UsersRanksTestPre**

  Descrizione:

  * Struttura: si verifica che tutti i bottoni per selezionare le varie classifica, presentino le corrette diciture in base alla lingua selezionata nel dispositivo.

- **UsersRanksTestUI**

  Descrizione:

  * Comportamento: si verifica che dalla schermata di visualizzazione delle classifiche degli utenti, selezionando un utente venga effettivamente mostrato il profilo dell'utente.

- **WorkgroupsRanksTestPre**

  Descrizione:

  * Struttura: si verifica che tutti i bottoni per selezionare le varie classifica, presentino le corrette diciture in base alla lingua selezionata nel dispositivo.

- **WorkgroupsRanksTestUi**

  Descrizione:

  * Comportamento: si verifica che dalla schermata di visualizzazione delle classifiche dei workgroup, selezionando un workgroup venga effettivamente mostrata la schermata delle classifiche degli user.

Componente com.woty.android.model

Tabella 15: Riassunto test di unità componente Model - Mobile

Classe di test	Classe testata	Esito della prova
ChallengeTest	Challenge	Positivo
ComboChallengeTest	ComboChallenge	Positivo
ConfigurationTest	Configuration	Positivo
DescriptionTest	Description	Positivo
ImageDescriptionTest	ImageDescription	Positivo
MultiOptionChallengeTest	MultiOptionChallenge	Positivo
QRChallengeTest	QRChallenge	Positivo
QuestHistoryTest	QuestHistory	Positivo
QuestTest	Quest	Positivo
TextDescriptionTest	TextDescription	Positivo
TrueFalseChallengeTest	TrueFalseChallenge	Positivo
UsersRankTest	UsersRank	Negativo
UserTest	User	Positivo
VideoDescriptionTest	VideoDescription	Positivo
WorkgroupsRankTest	WorkgroupsRank	Positivo
WorkgroupTest	Workgroup	Positivo

- **Classe ChallengeTest**

  Classe testata: `Challenge`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di Challenge, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe ComboChallengeTest**

  Classe testata: `ComboChallenge`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di ComboChallenge, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe ConfigurationTest**

  Classe testata: `Configuration`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di Configuration, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione.

  Errori riscontrati: Nessuno.

- **Classe DescriptionTest**

  Classe testata: `Description`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di Description, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe ImageDescriptionTest**

  Classe testata: `ImageDescription`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di ImageDescription, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe MultiOptionChallengeTest**

  Classe testata: `MultiOptionChallenge`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di MultiOptionChallenge, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

Errori riscontrati: Nessuno.

- **Classe QRChallengeTest**

  Classe testata: `QRChallenge`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di QRChallenge, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe QuestHistoryTest**

  Classe testata: `QuestHistory`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di QuestHistory, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione.

  Errori riscontrati: Nessuno.

- **Classe QuestTest**

  Classe testata: `Quest`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di Quest, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione.

  Errori riscontrati: Nessuno.

- **Classe TextDescriptionTest**

  Classe testata: `TextDescription`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di TextDescription, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe TrueFalseChallengeTest**

  Classe testata: `TrueFalseChallenge`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di TrueFalseChallenge, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe UsersRankTest**

  Classe testata: `UsersRank`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di UsersRank, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione.

Errori riscontrati: A seguito di diversi cambiamenti apportati ad una delle funzioni del server alcuni valori prima considerati corretti hanno perso di validità. Cambiando quindi i valori di riferimento all'interno del test ed eseguendolo ad uno dei periodici controlli del codice è stato riscontrato un errore. A seguito di una verifica sono quindi stati aggiornati tutti i controlli nei metodi che utilizzavano gli indici errati. Ora il test da esito positivo.

- **Classe UserTest**

  Classe testata: `User`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di User, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe VideoDescriptionTest**

  Classe testata: `VideoDescription`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di VideoDescription, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

- **Classe WorkgroupsRankTest**

  Classe testata: `WorkgroupsRank`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di WorkgroupsRank, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione.

  Errori riscontrati: Nessuno.

- **Classe WorkgroupTest**

  Classe testata: `Workgroup`

  Descrizione: l'obiettivo primario di questa classe è testare che, all'invocazione di uno dei principali metodi di Workgroup, se i parametri passati non sono corretti, venga restituito un opportuno valore invalido o un'opportuna eccezione. Verrà inoltre testata la correttezza della rappresentazione di un'istanza di questa classe offerta dal proprio metodo toStrings().

  Errori riscontrati: Nessuno.

Componente com.woty.android.util

Tabella 16: Riassunto test di unità componente Util - Mobile

Classe di test	Classe testata	Esito della prova
`JParserTest`	`JParser`	Positivo
`UtilTest`	`Util`	Positivo

- **Classe JParserTest**

Classe testata: `JParser`

Descrizione: utilizzando una libreria di parsing sviluppata e testata da Google si è ritenuto necessario testare solamente la validità dei parametri di invocazione dei metodi. Nel caso venga passato un parametro non corretto, deve venir restituito un opportuno valore invalido o un'opportuna eccezione.

Errori riscontrati: Nessuno.

- **Classe UtilTest**

  Classe testata: `Util`

  Descrizione: classe che testa il corretto comportamento dei metodi contenuti in questa classe di utilità. Principalmente viene testata la corretta concatenazione tra array di stringhe.

  Errori riscontrati: Nessuno.

6.3.2 Test di integrazione

I test di integrazione pianificati sono stati realizzati usando in congiunzione gli strumenti e le tecniche usati per testare le componenti model e view.
A livello di model si è proceduto con l'eseguire lo stub delle funzioni di comunicazione con il server per restituire dei dati valori alla richiesta di specifiche risorse. Questi valori restituiti vengono normalmente gestiti dalle classi interessate(model, presenter, view, util, etc.) e mostrati all'utente dalla view. Si è quindi testato che il valore atteso nella view corrispondesse a quello ottenuto.
Grazie ai test di integrazione copriamo quindi anche tutte le classi presenter non considerate dai test di unità.

Tabella 17: Sottosistema Mobile - test di integrazione

Codice Test	Classi Testate	Esito dei test
TIM-LG	view.Login view.Vista presenter.LoginP presenter.Presenter model.User model.Resource model.Session model.Configuration util.JParser	Positivo
TIM-HM	view.Home view.Vista presenter.HomeP presenter.Presenter model.User model.Resource model.Session model.Configuration util.JParser	Positivo

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

TIM-MP	view.ModProfile view.Vista presenter.ModProfileP presenter.Presenter model.User model.Resource model.Session model.Configuration util.JParser	Positivo
TIM-PQ	view.PendingQuests view.Vista presenter.PendingQuestsP presenter.Presenter model.QuestHistory model.Resource model.Session model.Configuration util.JParser	Positivo
TIM-PP	view.PlayerProfile view.Vista presenter.PlayerProfileP presenter.Presenter model.User model.Resource model.Session model.Configuration util.JParser	Positivo
TIM-RK	view.UsersRanks view.WorkgroupsRanks view.Vista presenter.UsersRanksP presenter.WorkgroupsRanksP presenter.Presenter model.UsersRank model.WorkgroupsRank model.Resource model.Session model.Configuration util.JParser	Positivo

(Continua alla pagina successiva)

(Continua dalla pagina precedente)

TIM-SQ	view.SolveQuest view.Vista presenter.SolveQuestP presenter.Presenter model.Quest model.Description model.TextDescription model.ImageDescription model.Challenge model.ComboChallenge model.MultiOptionChallenge model.TrueFalseChallenge model.Resource model.Session model.Configuration util.JParser	Positivo

6.4 Test di Sistema

Si riporta di seguito l'esito dei test di sistema. Per la pianificazione degli stessi si rimanda alla sezione 4.1.

Tabella 18: Tracciamento requisito - verifica

Requisito	Codice prova	Esito prova
Fo-1	TFo-1	Superato
Fo-2.1.1	TFo-2.1.1	Superato
Fo-2.1.2	TFo-2.1.2	Non effettuato
Fo-2.2	TFo-2.2	Superato
Fo-2.3	TFo-2.3	Superato
Fo-2.4	TFo-2.4	Superato
Fo-2.5	TFo-2.5	Superato
Fo-3.1	TFo-3.1	Superato
Fo-3.2	TFo-3.2	Superato
Fo-3.3	TFo-3.3	Superato
Fo-4.1	TFo-4.1	Superato
Fo-4.2	TFo-4.2	Superato
Fo-4.3	TFo-4.3	Superato
Fo-4.4	TFo-4.4	Superato
Fo-4.6	TFo-4.6	Superato
Fo-5	TFo-5	Superato
Fo-6	TFo-6	Superato
Fo-7	TFo-7	Superato
Fo-8	TFo-8	Superato
Fo-8.1	TFo-8.1	Superato
Fo-9	TFo-9	Superato
Fo-10.1	TFo-10.1	Superato
Fo-10.2	TFo-10.2	Superato
Fo-10.3	TFo-10.3	Superato
Fd-1	TFd-1	Superato
Fd-2	TFd-2	Non effettuato

TFo-2.1.2 e TFd-2 non sono stati effettuati in quanto non sono stati implementati i requisiti corrispondenti. Si rimanda al documento *EstensioneRequisiti_v1.pdf* per ulteriori informazioni sulla completezza dei requisiti e le possibili estensioni degli stessi.

7 Dettaglio dell'esito delle revisioni

Ad ogni revisione il gruppo riceverà da parte del committente un esito riguardante la documentazione consegnata, che indicherà il grado di correttezza del lavoro svolto dal gruppo fino a quel momento.
Di seguito verranno descritte e le principali attività di verifica e correzione svolte dal gruppo a seguito delle revisioni.

7.1 Revisione dei Requisiti

A seguito della Revisione dei Requisiti sono stati riesaminati e corretti i seguenti documenti:

- **Analisi dei Requisiti**: sono stati modificati i paragrafi "Contesto d'uso" e "Funzionalità del prodotto" cercando di definirne meglio i contenuti, inoltre è stato rimosso il paragrafo "Assunzioni e dipendenze".
  Sono stati corretti e revisionati gran parte dei diagrammi degli Use Case[g], aggiungendone anche di nuovi per aumentarne il livello di dettaglio.
  Sono stati rivisitati di conseguenza i relativi requisiti e sono stati modificati i requisiti segnalati come ambigui in maniera da renderli più specifici e verificabili durante il corso del progetto.
- **Piano di Progetto**: aggiustato in minima parte il ciclo di vita per ovviare ai ritardi maturati durante la fase di revisione dei requisiti e aggiustate di conseguenza alcune sezioni del documento. Per maggiori dettagli consultare il capitolo 2 del documento *PianoDiProgetto.pdf*.
- **Piano di Qualifica**: dopo aver aggiustato la struttura del documento sono stati aggiunti dettagli riguardanti le tecniche di analisi, le attività di verifica effettuata nella prima fase e i resoconti delle attività di verifica per le fasi Revisione dei Requisiti e Revisione di progettazione.
  Sono inoltre stati aggiunti dettagli sulle metriche del codice e, dopo un'approfondita ricerca, anche gli strumenti che si sono scelti per tutte le future verifiche sul codice.

7.2 Revisione di Progettazione

A seguito della Revisione dei Progettazione sono stati riesaminati e corretti i seguenti documenti:

- **Analisi dei Requisiti**: sono stati aggiunti gli scenari alternativi nei casi di estensione degli Use Case e sono stati scomposti in sotto-requisiti i requisiti indicati nella correzione.
- **Piano di Progetto**: aggiornata la sezione "Analisi dei Rischi" aggiungendo il riscontro effettivo per i vari rischi. Aggiornata la sezione "Consuntivi" aggiungendo le sottosezioni "Considerazioni" e "Preventivo a finire".
- **Piano di Qualifica**: aggiunta la sezione riguardante gli obiettivi di qualità del software. Aggiunta inoltre la pianificazione dei test di integrazione e aggiornata la sezione "Dettaglio delle verifiche tramite prove".
- **Specifica Tecnica**: l'architettura del sottosistema mobile è stata completamente rivisitata, passando dal design pattern Model View Controller al pattern Model View Presenter. La modifica si è resa necessaria in seguito a difficoltà nel disaccoppiare coerentemente View e Controller, complice anche la struttura delle Activity imposta da Android.

7.3 Revisione di Qualifica

A seguito della Revisione dei Qualifica sono stati riesaminati e corretti i seguenti documenti:

- **Norme di Progetto**: aggiornate norme sui diagrammi UML. Aggiornate norme di codifica e di verifica. Inserite sezione dedicate alle norme sui test di unità e integrazione.
- **Specifica Tecnica**: sono state migliorate le descrizioni dei design pattern e aggiunto il design pattern MVP tra le descrizioni. Sono state specificate meglio le librerie e le gemme utilizzate nello sviluppo del progetto.
- **Definizione di Prodotto**: è stata uniformata la struttura del documento soprattutto nella prima parte dello stesso. Aggiunti ulteriori diagrammi delle classi e tipi di ritorno dei metodi nella prima parte. Aggiornate classi e metodi, e aggiunto il file di configurazione nella parte Mobile.
- **Manuali**: aggiornati messaggi di errore e screenshot dell'applicazione in ogni manuale. Inserita la specifica della dipendenza del plugin `growl` nel *Manuale utente desktop*.
- **Piano di Progetto**: aggiunto consuntivo della fase RQ-RA. Aggiunto consuntivo finale e totale del progetto.
- **Piano di Qualifica**: aggiornati test di unità e di integrazione e metriche. Inseriti gli esiti dei test di sistema.