Università
Ca'Foscari
Venezia

23 NOVEMBRE 2017

PLUTO

VERSIONE 1.0

PIANO DI TESTING

GRUPPO: M.A.M.A.

ALBERTO VENERI 860028
ANDREA ANTONIAZZI 857392
MARCO MARANGONI 858450
MAURO MOLIN 857855

Sommario

Introduzione

Il presente documento espone le modalità in cui verrà eseguito il testing dell'applicazione Pluto. I punti trattai saranno i seguenti:

- **Processo di testing adottato:** descrizione dettagliata della tipologia di testing adottata e delle motivazioni che ci hanno portato a tale scelta
- **Modalità di verifica dei singoli requisiti:** verranno descritte le modalità con cui verranno testati i requisiti descritti nel documento di specifica, ponendo particolare attenzione alla checklist degli errori più comuni
- **Elementi testati:** suddivisione delle attività di testing da svolgere
- **Schedule del testing:** la specifica del tempo e delle risorse riservato alle attività necessarie a testare correttamente il progetto
- **Procedure di registrazione dei test:** descrizione dei metodi di tracciamento dell'attività di testing
- **Requisiti hardware e software utilizzati:** requisiti hardware e software adottati per il testing
- **Vincoli che condizionano il testing:** vincoli di cui tener conto durante lo svolgimento delle attività di testing al fine di rispecchiare gli standard di qualità che ci siamo imposti

Il processo di testing adottato

Dato che stiamo sviluppando questo progetto utilizzando il paradigma ad oggetti, riteniamo sia molto vantaggioso utilizzare un processo di testing della tipologia bottom-up. Inizieremo il processo testando le componenti a basso livello (le singole classi), verificandone le funzionalità di base e controllando come interagiscono con gli altri moduli presenti nel sistema, passando poi gradualmente alle componenti più ad alto livello.

Ponendo particolare attenzione in fase di design e progettazione cercheremo di minimizzare il lato negativo del paradigma bottom up, ovvero di riscontrare errori di progettazioni solo in ultima analisi. In ogni caso, avendo noi adottato un modello di processo di tipo evolutivo, eventuali errori di progettazione verranno corretti durante i successivi cicli di evoluzione e testati nuovamente prima della conclusione del progetto.

Per la fase di apha testing intendiamo utilizzare un approccio combinato di black-box e white-box così da ridurre al minimo la possibilità di errore. Ispezionare il codice ci permetterà di osservare errori di tipo off-by-one rari oppure verificare errori in branch del programma difficili da raggiungere con il semplice utilizzo del programma.

Per la fase di beta test naturalmente sarà utilizzato esclusivamente un approccio a black-box, a meno che il tester non abbia intenzione di crearsi un clone del repository pubblico GitHub al fine di analizzare più a fondo il codice.

Modalità di verifica dei singoli requisiti

Verrà verificata la presenza di tutti i requisiti presenti nel documento di specifica (ID: RF-X) testandone la correttezza sotto opportuni input. Sarà quindi controllato il codice per eseguire un testing con approccio white-box. Per tutti i requisiti non funzionali specificati nel documento (ID: RNF-X) verrà svolto un test black-box molto approfondito, dato che questa tipologia di requisito necessita una verifica molto dettagliata per quanto riguarda la parte che sarà esposta all'utente.

Ogni test sarà effettuato verificato i possibili errori di sviluppo presenti nella seguente checklist:

- Casting: effettuati solo se strettamente necessari e sicuramente corretti
- Streams: eventuali errori di gestione di streams (mancanta chiusura, apertura di stream su file non esistenti, ecc.)
- Object Serialization: controllare che il processo di serializzazione e deserializzazione sia effettuato in modo corretto
- NullPointerException: errore comune nella gestione delle referenze degli oggetti
- IllegalArgumentException: errore nel passaggio di parametri ad un metodo
- IOException: errori nell'I/O dell'applicazione
- FileNotFoundException: errore dovuto alla mancata presenza di un file in memoria persistente

Inoltre, per avere una lista esaustiva di possibili vulnerabilità che possono occorrere, teniamo in considerazione la risorsa https://www.securecoding.cert.org/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java. Questa documento, rilasciata dalla Carnegie Mellon University e nota nel campo della sicurezza informatica, espone una serie di vulnerabilità ed errori nella programmazione Java in ordine di importanza o frequenza.

I test saranno effettuati anche effettuando rotazioni dello schermo di 90 e 180 gradi, cambiando la lingua e l'ora del device. Parte dei test saranno svolti utilizzando la versione Android 4.1 emulata attraverso gli strumenti che Android Studio fornisce, per verificare che i requisiti di sistema posti siano effettivamente sufficienti tenendo presenti le inevitabili limitazioni di velocità nell'esecuzione che un emulatore porta.

Elementi testati

Il progetto è composto da due parti:

- SiopeDownloader: il modulo Java (Maven) utile al download e al parsing dei dati forniti dal portale siope.it. Questo fornisce i dati grezzi senza introdurre alcuna manipolazione
- PLUTO: l'applicazione che seguirà le indicazioni presenti nel documento di specifica.

Intendiamo testare ogni singola classe e activity di entrambi i moduli sviluppati al fine di verificare che essa effettivamente assolva le sue funzioni in modo esaustivo ed individuare gli errori nei servizi che essa offre. Inoltre, saranno svolte verifiche sulle integrazioni che riteniamo più delicate.

Le macro aree che intendiamo analizzare comprendono:

- Download
- Parsing dei dati
- Elaborazione dei dati
- Design grafico e UX

Schedule del testing

Come specificato nel piano di progetto il tempo dedicato al testing sarà complessivamente 30 ore ed è suddiviso nei tre cicli previsti nell'evoluzione dell'app come segue:

- 5 ore durante il primo ciclo mirato al testing delle componenti principali e a trovare gli errori derivanti dal design, al fine di sapere come rimodellare il design del progetto in maniera più corretta
- 10 ore al secondo ciclo di evoluzione svolto con metodologia white-box per comprendere e correggere i possibili errori di integrazione fra le componenti
- 15 ore a conclusione del progetto con tipologia black-box per eliminare il maggior numero di errori visibili all'utente. I requisiti non funzionali saranno testati, per assicurarsi che siano conformi al piano di progetto.

Procedure di registrazione dei test

Essendo il nostro un gruppo di piccole dimensioni, formato da persone a stretto contatto, riteniamo non sia necessario stilare una lista molto dettagliata dei bug trovati, bensì verranno comunicati a voce a colui che ha sviluppato il modulo interessato dall'errore. Per mantenere una traccia ripercorribile dei test e degli errori trovati, procederemo nel seguente modo.

Sarà presente una lista condivisa di issues, aggiornata su GitHub, che conterrà per ogni errore:

- Un titolo e una descrizione esaustiva a parole
- Un'etichetta che agevolerà la categorizzazione delle issues
- Una milestone che svolgerà il ruolo di "contenitore" per la issue, cioè ci aiuterà ad indentificare le features con cui quell'issue sono associate
- Lo studente incaricato alla risoluzione
- Un eventuale commento

Questa lista annullerà il rischio di dimenticanze su errori trovati e non ancora corretti.

Per chiarezza verrà stilata una seconda lista contenente i test effettuati. Per ogni test sarà presente:

- Una breve descrizione del test
- Casi limite (se presenti) testati

La fase di test è incaricata ad una persona, il che significa che sicuramente non vi sarà una inutile ridondanza nei test. Tre dei ruoli presenti usualmente nel team di test ricadranno su Antoniazzi Francesco Andrea (857392), il quale si preoccuperà di svolgere il ruolo di ispettore, lettore e moderatore e che manterrà una continua comunicazione con l'autore del codice sotto analisi.

Requisiti hardware e software utilizzati

Requisiti Software

L'applicazione necessita di Android 4.1 per avere un corretto funzionamento. Tutti i pacchetti aggiuntivi necessari all'app saranno integrati all'interno di essa.

Requisiti Hardware

Come specificato nel documento di specifica dei requisiti ci limitiamo alle caratteristiche minime che un dispositivo devi esporre per essere certificato compatibile con Android 4.1. Pertanto, i requisiti hardware necessari al corretto funzionamento dell'applicazione PLUTO sono esaustivamente descritti nel documento https://source.android.com/compatibility/4.1/android-4.1-cdd.pdf .

Vincoli che condizionano il testing

I vincoli imposti sono prettamente di carattere temporale dato che la scadenza fissata per la pubblicazione della seconda beta è prevista per il 20 dicembre 2017.

Naturalmente, il testing dovrà inoltre rispettare le modalità imposte dal presente documento per poter rispettare i requisiti di qualità che il gruppo si è stabilito.