Piano di Progetto

Smart Meal

Gruppo: Quadcore

Facci Matteo 875377
Guidobene Davide 877150
Pittini Enrico 877345
Rosin Giacomo 875724

Versione 1.0

16/11/2020

Indice

Indice 2

Introduzione 3

1. Metodologie di testing 4

2. Tracciabilità dei requisiti 5

3. Elementi testati 8

4. Schedule del testing: tempo e risorse allocate 9

5. Procedure di registrazione dei test 10

6. Requisiti hardware e software utilizzati 11

7. Vincoli che condizionano il testing 12

Introduzione

Tale documento ha lo scopo di fornire una descrizione delle attività e delle metodologie di testing svolte per la realizzazione dell'applicazione "Smart meal".

In particolare saranno trattati i seguenti punti:

1. Metodologie di testing, ovvero le principali tecniche e strategie usate per il testing;
2. Tracciabilità dei requisiti, ovvero per ogni requisito funzionale diamo una descrizione del relativo test;
3. Elementi testati, ovvero diamo una descrizione dei principali elementi dell'applicazione che andremo a testare ;
4. Schedule del testing, ovvero pianificazione temporale delle varie attività di testing ;
5. Procedure di registrazione dei test, ovvero descriviamo il form che verrà usato nel registrare i vari test;
6. Requisiti hardware e software, ovvero descriviamo le componenti hardware e software necessarie per la realizzazione della fase di testing ;
7. Vincoli per il testing, descrive tutti i vincoli che devono essere applicati nella fase di testing, per ottenere un testing che sia valido.

1. Metodologie di testing

Useremo varie metodologie di testing.

- Useremo **metodologie di testing statiche**, ed in particolare sia la metodologia Inspection che la metodologia Walkthrough. Tali tecniche le applicheremo in modo regolare e continuo durante tutto lo sviluppo del progetto. Grazie a queste possiamo catturare molte categorie di errori, sia di sintassi ma anche di semantica, senza far effettivamente girare il codice. Queste metodologie le applicheremo a coppie (in particolare il Walkthrough), in modo da garantire un controllo più approfondito.
- Useremo la metodologia **incremental testing** per testare i vari moduli del programma e in particolare la loro aggregazione. Tramite questa metodologia ogni volta che creiamo un nuovo modulo testiamo nuovamente tutti i moduli già creati e testiamo nuovamente tutti i vari possibili gruppi di moduli: a questi test ripetuti aggiungiamo i test che riguardano il nuovo modulo. Tale metodologia di testing è una metodologia che sonda la correttezza dell'applicazione in modo disciplinato e accorto.
- Useremo la **metodologia bottom-up** per testare i vari moduli del programma e in particolare la loro aggregazione. Partiamo dunque con il testare i vari moduli più piccoli separatamente per poi procedere gradualmente al testing delle componenti più complesse e strutturate. Abbiamo scelto tale approccio, preferendolo a quello top-down, in quanto il nostro programma è orientato agli oggetti ed in quanto vogliamo dare maggiore peso alla rilevazione degli errori di basso livello.
- Useremo anche la **metodologia thread testing**, testando così in modo approfondito i vari flussi di esecuzione della nostra applicazione e le relative possibili interazioni. Tale testing è particolarmente importante per la nostra applicazione, in quanto si basa in modo massiccio sulla programmazione ad eventi. Ci sono molti eventi di interesse a cui sono appese varie callback (azioni da eseguire): all'accadere di un evento si diramano vari flussi di esecuzione.
- Useremo inoltre la **metodologia back-to-back testing**. Il nostro modello di ciclo di vita del software evolutivo, avremo a disposizione varie versioni dell'applicazione: ciascuna di esse è risultante da un'iterazione del ciclo evolutivo. Tale modello si presta bene al back-to-back testing. Ogni volta che abbiamo una nuova versione dell'applicazione, testiamo che tale versione realizzi almeno tutte le funzionalità realizzate dalla precedente versione. Ogni versione più raffinata deve essere buona almeno quanto la precedente.
- Infine specifichiamo che nel testare un certo modulo applicheremo la **metodologia black-box e/o la metodologia white-box** per scegliere il criterio con cui testare tale modulo. In particolare, abbiamo deciso che applicheremo la metodologia black-box sicuramente: in questo modo vogliamo testare in modo trasparente che almeno le funzionalità e le post condizioni del modulo siano rispettate. Per i moduli di maggiore importanza applichiamo, oltre appunto ai test black-box, anche test white-box: in questo modo testiamo il modulo in modo più accorto e approfondito, andando a prendere in considerazione la sua implementazione e i suoi punti critici interni.

2. Tracciabilità dei requisiti

Segue ora una tabella in cui per ciascun requisito funzionale (vedi documento di Analisi e Specifica dei requisiti) è indicato il relativo test.

ID requisito	Nome requisito	Test
RF-01	Selezione ruolo cliente/gestore	Al primo avvio dell'applicazione, selezionando la modalità desiderata (o cliente o gestore) l'applicazione deve aprirsi in quella modalità.
RF-02	Cambio lingua italiano/inglese automatico	Aprendo l'applicazione, la lingua visualizzata deve essere la stessa lingua impostata per il dispositivo Android (o inglese o italiano).
RF-03	Notifica quando il cliente entra nel locale	Portando un dispositivo Android, con installata l'applicazione in modalità cliente, all'interno di un locale con all'interno un dispositivo Android con installata l'applicazione in modalità gestore, il dispositivo cliente deve ricevere una notifica di benvenuto.
RF-04	Visione menu	Dalla schermata principale, cliccando sulla visualizzazione del menù deve essere effettivamente visualizzato il menu del locale.
RF-05	Visione descrizione (sia per i clienti che per il gestore).	Dalla schermata principale, cliccando sulla visualizzazione della descrizione deve essere effettivamente visualizzata la descrizione del locale.
RF-06	Visione posizione mappa	Dalla schermata principale, cliccando sulla visualizzazione della posizione deve essere effettivamente visualizzata la posizione del locale sulla mappa.
RF-07	Accesso stanza virtuale	Usando un dispositivo Android modalità cliente, che è all'interno del locale, cliccando sull'accesso alla stanza virtuale deve essere effettivamente fatta iniziare la sessione di comunicazione con il

		locale.
RF-08	Scelta tavolo	Dalla stanza virtuale il cliente può cliccare sul tavolo su cui si siederà. Il gestore dovrà controllare nella sua applicazione se il posto selezionato dal cliente corrisponderà con quello occupato nella visione dei posti.
RF-09	Chiamata cameriere tramite shake del telefono	Scuotendo il dispositivo bisogna verificare se effettivamente arrivi la richiesta nel dispositivo del gestore. E' necessario per garantire una corretta comunicazione tra il gestore e il cliente.
RF-10	Pin per il primo accesso del gestore	Dopo aver effettuato la selezione del ruolo, il gestore deve accedere all'applicazione solamente se il pin corrisponde. Lo scopo è permettere l'autenticazione.
RF-11	Creazione stanza virtuale	Il gestore dalla homepage può creare una stanza virtuale, ed è necessario verificare che i vari clienti possano averne accesso tramite l'apposita sezione dell'applicazione.
RF-12	Visione lista richieste shake	Dalla stanza virtuale, il gestore può vedere tutte le notifiche di shake da parte dei clienti del locale. E' necessario verificare che tutte le richieste di shake siano comprese nella lista disponibile al gestore.
RF-13	Visione elenco tavoli occupati	Dalla stanza virtuale il gestore può vedere quali tavoli sono occupati e quali non lo sono. Bisognerà testare se la richiesta da parte di un cliente di un tavolo sarà rappresentata da un tavolo occupato nella sezione del gestore.
RF-14	Aggiungere/rimuovere/modificare tavoli occupati	Il gestore dalla sezione stanza virtuale cliccando su una delle seguenti voci può effettuare la corrispondente operazione, che dovrà riflettersi nella visione dei

		tavoli occupati, per permettere al gestore di controllare i posti del locale.

3. Elementi testati

Gli elementi che saranno sottoposti al testing saranno tutti gli elementi che costituiscono il codice che scriveremo, quali metodi, gestione dei sensori, callback, oggetti, interfacce grafiche ed interazioni dell'utente, che saranno specificate nel Documento di Progettazione. Particolare attenzione (numerosi cicli di testing) verrà dedicata alle parti del progetto più complesse e delicate, al fine di ottenere un funzionamento esente il più possibile da bug.

Inoltre verrà controllato che ci sia coerenza e completezza tra i requisiti funzionali, definiti nel Documento di Analisi e Specifica, e le funzionalità rese disponibili dalla nostra applicazione.

Infine ci assicureremo che quanto presentato dalla nostra applicazione rispetti i requisiti non funzionali descritti nel Documento di Analisi e Specifica di requisiti ed in particolare le relative misure di qualità.

4. Schedule del testing: tempo e risorse allocate

La gestione del tempo e delle risorse seguirà le seguenti specifiche:

- **Inspection e Walkthrough**: dal momento che il nostro team ha deciso di adottare il pair programming nel corso della fase di programmazione, è già intrinsecamente presente una fase di analisi statica. Quindi riteniamo che sarà sufficiente svolgere un lavoro puramente dedicato a un'analisi statica approfondita del codice (di circa 2 ore) solamente ogni 2 settimane.
- **Incremental Testing**: avremo cura di usarlo all'aggiunta di ogni nuova funzione, per assicurarci che non solo essa sia corretta, ma anche che non porti a una regressione del sistema nel suo complesso. Coerentemente con il linguaggio adottato, che fa uso del paradigma ad oggetti, la modalità di testing scelta è bottom-up.
- **Thread Testing**: data la natura asincrona delle applicazioni Android, abbiamo ritenuto che fosse necessario adottare il thread testing, per ridurre la probabilità di eventuali problemi dovuti alla concorrenza. Avremo cura di eseguirlo al termine dello sviluppo di ogni iterazione del modello evolutivo. In aggiunta svolgeremo dei test supplementare in itinere quando ci sembrerà necessario, per esempio in presenza di porzioni di codice che richiedono particolare attenzione alla concorrenza tra thread.
- **Back-to-back Testing**: ad ogni nuova versione dell'applicazione, questa verrà confrontata con le versioni precedenti in modo da controllare che non avvengano regressioni.
- **Black-box and White-box**: nel corso della fase di testing faremo principalmente uso dell'approccio black-box per controllare ogni nuovo modulo nel momento in cui viene aggiunto e il sistema nel suo complesso, riservandoci di usare in aggiunta anche l'approccio white-box per i casi più delicati e critici per il funzionamento complessivo dell'applicazione.

5. Procedure di registrazione dei test

Per la registrazione dei risultati dei test useremo una tabella con i seguenti campi:

- **CRF (Codice Requisito Funzionale)**: codice del requisito funzionale testato
- **Descrizione**: descrizione del test svolto
- **Data**: data di esecuzione del test
- **Tester**: membro del gruppo che ha eseguito il test
- **Risultato atteso *(% successi)***: numero di successi, in percentuale, che il test deve raggiungere per considerarsi superato
- **Risultato effettivo *(% successi)***: numero di successi, in percentuale, che il test ha raggiunto
- **Numero test**: numero di test svolti
- **Note**: osservazioni e eventuali misure correttive

CRF	**Descrizione**	**Data**	**Tester**	**Risultato atteso** ***(% successi)***	**Risultato effettivo** ***(% successi)***	**N° test**	**Note**

6. Requisiti hardware e software utilizzati

Riteniamo che per una più esaustiva esecuzione dei test sia utile adoperare dispositivi con caratteristiche hardware e software diverse. Per questo motivo ciascun membro del team metterà a disposizione il proprio cellulare, fornendo al team un totale di 4 dispositivi dotati di WI-FI, Bluetooth, accelerometro e GPS.

Marca	**Nome**	**Versione Android**	**CPU (GHz)**	**RAM (GB)**	**Display (‘’)**
Samsung	Galaxy A40	10	2x1.8 + 6x1.6	4	5.9
Motorola	Moto X Play	6.0	4x1.7 + 4x1.0	2	5.5
Honor	6X	7.0	4x2.1 + 4x1.7	3	5.5
OnePlus	One	6.0	4x2.5	3	5.5

Specifiche dei dispositivi di testing.

In aggiunta useremo strumenti già presenti nell’ambiente di sviluppo Android Studio, quali emulatore, debugger e profiler.

Allo scopo di aumentare l’efficienza del nostro lavoro di testing, una parte di questi test sarà automatizzata usando il framework JUnit.

7. Vincoli che condizionano il testing

Con l'inizio della fase di programmazione, avrà inizio anche la fase di testing che continuerà fino alla consegna del progetto in data 15/01/2021.
Siccome il progetto e lo sviluppo dell'applicazione è no-profit, abbiamo deciso di limitarci a usare le risorse già a nostra disposizione (come ad esempio i nostri cellulari) per testarne il funzionamento senza fare grandi investimenti monetari.