1. **INTRODUZIONE**

Il testing consiste nel trovare le differenze tra il comportamento atteso specificato attraverso il modello di sistema e il comportamento che si ha dopo l’implementazione del sistema.

Lo scopo principale di questa attività è quello di testare il sistema e di rilevare gli errori e i problemi che si vengono a verificare in fase di implementazione. Questo vuol dire che nella fase di testing andiamo a cercare di creare il massimo numero di errori in modo tale che questi possano essere corretti in questa suddetta fase. Lo scopo del documento è quello di andare a definire i test case su cui verranno testate le funzionalità del sistema. Per ogni funzionalità saranno forniti un numero sufficiente di istanze di input in modo tale da fornire almeno un test case composto da dati corretti e quindi appartenenti a classi valide per ogni campo di input che rispettano le condizioni definite nel documento di test plan. Ci sarà inoltre un test case per ogni input che prevede una classe non valida e almeno un test case che non soddisfa le condizioni definite nel test plan.

1. **RIFERIMENTI E RELAZIONI**

In questa parte vengono descritte le relazioni con gli altri documenti prodotti durante la fase di sviluppo del progetto software TuttoElettronica. Tali relazioni sono necessarie al fine di produrre una fase di testing che sia coerente con tutte le specifiche dedotte durante l’analisi ed il raffinamento dei requisiti e delle funzionalità che il sistema dovrà fornire. Le relazioni che andiamo a prendere in considerazione sono con:

* RAD – documento di analisi dei requisiti
* SDD – documento di system design
* ODD – documento di object design
  1. **RELAZIONE CON IL DOCUMENTO DI ANALISI DEI REQUISITI (RAD)**

Tali relazioni riguardano i requisiti funzionali e non funzionali del sistema. I test delle funzionalità terranno conto delle specifiche espresse nel RAD. I test devono tener presente dei diversi attori descritti nel documento di analisi (RAD):

* Cliente
* Gestore
* Prodotti

Per ciascun cliente e gestore dovrà essere testata la validità dei campi associati.

Per i prodotti dovranno essere testati inserimento, cancellazione, ricerca, quantità.

Inoltre deve essere considerato il flusso di controllo descritto nei diagrammi delle sequenze di operazioni, in modo da testare determinate funzionalità all’interno e all’esterno del proprio flusso standard.

* 1. **RELAZIONI CON IL DOCUMENTO DI SYSTEM DESIGN (SDD)**

La pianificazione dei test delle componenti rispecchia la decomposizione in sottosistemi specificata nel documento del System Design.

* 1. **RELAZIONI CON IL DOCUMENTO DI OBJECT DESIGN (ODD)**

La fase di testing deve considerare il documento di Object Design in quanto esso fornisce la base per realizzare l’implementazione. Pertanto è necessario effettuare il testing delle unità per individuare le differenze tra il comportamento atteso e quello specificato.

1. **PANORAMICA DEL SISTEMA**

Il sito di commercio elettronico che si vuole realizzare è concepito sull’idea di dare la possibilità, a dei clienti, di riparare prodotti elettronici. Il sito, inoltre, intende pubblicizzare prodotti da esso venduti permettendo, inoltre, ai clienti la possibilità di effettuare prenotazioni online.

Il sistema TuttoElettronica segue il modello MVC per la suddivisione dei livelli:

* Model: che rappresenta i dati dell’applicazione;
* View: che rappresenta la visualizzazione degli oggetti;
* Controller: che rappresenta l’insieme di regole che permettono la trasformazione degli input sulle viste in modifiche del modello.

Ogni livello, come descritto nel documento SDD, è stato suddiviso in sottosistemi in modo da raggruppare le componenti.

Ciascun sottosistema, appartenente a ciascun livello, deve essere testato.

Il presente test plan relativo al sistema TuttoElettronica si pone i seguenti obiettivi:

* Dettagliare le attività richieste per preparare e condurre il testing;
* Definire le fonti usate per preparare la pianificazione;
* Affinchè il testing riesca a raggiungere al meglio i suoi obiettivi, si è ritenuto necessario non condurre tale processo in modo casuale, cioè guidato solo dalla fantasia e dalla esperienza dei test-engineer, ma definendo i punti generali al quale far riferimento:
* **Individuare quando deve partire il processo di testing e quale è il criterio di terminazione da utilizzare**

Identificare a priori lo stato di inizio e di fine dell’interno di un processo è fondamentale per evitare che il processo cominci quando ancora non sono presenti tutte le informazioni necessarie o che si prolunghi eccessivamente.

* **Definire per ogni test i risultati attesi**
* La descrizione dei risultati ottenuti dall’esecuzione di un test case è determinante poiché, senza una precisa connotazione degli output attesi, si corre il rischio di interpretare come corretti dei risultati che viceversa non lo sono.
* **Analizzare a fondo i risultati di ciascun test**
* **Non pianificare il testing assumendo tacitamente che nessun errore sarà individuato**

Questo è un comportamento che deriva direttamente da un’errata interpretazione del documento di testing, laddove si crede che il testing sia la verifica delle funzionalità del sistema, mentre il vero processo di testing è la ricerca del maggior numero di errori.

* **Il miglior test case è quello che ha la più alta probabilità di scoprire errori non ancora rilevati.**

Questo principio è poco pragmantico, in quanto non è possibile a priori valutare la probabilità di scoprire errore da parte di un test case, ma possiamo affermare che si deve sempre tendere ad una produzione limitata il più possibile di test case di qualità, al fine di massimizzare il rapporto tra gli errori scoperti ed i test utilizzati.

Ovviamente non sempre è possibile riuscire a rispettare tutti i principi guida elencati, per cui si cerca, all’interno delle metodologie, di seguire il più possibile le linee guida che essi tracciano.

**PASS/FAIL CRITERIA**

La fase di testing necessita di criteri formali per la determinazione del successo o dell’insuccesso di un determinato test. I dati di ogni input relativi ai test verranno divisi in classi di equivalenza: un input appartenente ad una classe specifica supera il test se l’output ottenuto corrisponde ai risultati attesi, specificati nell’oracolo.

**PASS CRITERIA**

I pass criteria determinano l’insuccesso del test e quindi la correttezza del comportamento della componente testata. I pass criteria per la fase di testing del sistema TUTTO-ELETTRONICA sono essenzialmente raggruppati in due categorie principali, che possono essere raffinate qualora il test necessiti di raffinamento. Le categorie individuate sono le seguenti:

* Comportamento atteso:
  + L’oracolo rileva che il comportamento della componente rispetti il comportamento atteso
* Nessun errore rilevato dalla componente:

La componente non genera alcuna eccezione non prevista.

**FAIL CRITERIA**

I fail criteria determinano il successo del test e quindi la presenza di errori nel comportamento della componente testata. I failcriteria per la fase di testing del sistema SHOP-AUTORICAMBI sono essenzialmente raggruppati in 2 categorie principali che possono essere raffinate nei casi specifici. Le categorie individuate sono le seguenti:

* Errore rilevato dalla componente:
  + La componente genera una eccezione non prevista.
* Comportamento non atteso:

L’oracolo rileva che il comportamento della componente si discosti dal comportamento atteso rispetto agli input.

**APPROCCI**

Nella sessione di testing di TuttoElettronica sarà utilizzato un approccio di tipo “BLACK BOX”, il quale prevede che i test siano effettuati ad un livello di astrazione più alto, in modo da non entrare nei dettagli del codice, ma basandosi sulle specifiche delle funzionalità da testare.

L’approccio alla fase di testing si compone di tre fasi:

* Testing di unità, che controlla i singoli componenti (classi, metodi)
* Testing di integrazione, che testa l'integrazione dei vari sottosistemi

Testing funzionale, che verifica la funzionalità dell’intero sistema assemblato.

**TESTING DELLE UNITA’**

Con il testing di unità sarà effettuato un controllo delle varie classi e metodi del sistema, quindi saranno ricercate le condizioni di fallimento evidenziando gli errori. Il testing di unità sarà eseguito utilizzando il framework JUnit. In particolare, per ogni classe che esegue operazioni complesse, sarà sviluppata la relativa classe JUnit.

**TESTING DI INTEGRAZIONE**

Con il testing di integrazione si effettuerà un controllo sull’integrazione delle varie componenti del sistema. Si adotterà una strategia di tipo “Bottom-up”.

Per effettuare questi test di integrazione, spesso sarà necessario l’utilizzo di driver, dato che tale strategia va ad integrare man mano i sottosistemi partendo dal layer che si trova più in basso nella scala gerarchica.

**SOSPENSIONE E RIPRISTINO**

La fase di testing del sistema TuttoElettronica può essere interrotta e ripresa più volte, se il fine ultimo è di rendere il sistema corretto e ogni funzionalità completa. Di seguito sono riportati i criteri secondo cui è necessario sospendere la fase di testing e le modalità secondo cui poi deve essere ripresa.

**CRITERI DI SOSPENSIONE**

La sospensione della fase di testing deve avvenire qualora un test abbia esito positivo ovvero si è riscontrato un errore all’interno di una componente.

Questi criteri comprendono tutti gli errori che hanno un impatto dannoso sul progresso dell’attività di testing, come il crash del database, il crash del sistema operativo, il fallimento di intere funzionalità ed anche problemi relativi all’ambiente di sviluppo del testing.

**CRITERI DI RIPRISTINO**

La ripresa del test avviene soltanto quando tutti i problemi relativi alla sospensione del test sono stati risolti. L’attività di testing riprenderà a partire dal test case che ha causato la sospensione.

1. **STRUMENTI PER IL TESTING**

Gli strumenti per il testing verranno divisi in due tipi differenti:

* Strumenti hardware
* Strumenti software

**STRUMENTI HARDWARE**

Il sistema sarà testato su un personal computer con la necessaria dotazione software. Non sono richieste particolari specifiche hardware se non quelle adatte a far funzionare il DBMS server e il Web server apache.

* 1. **STRUMENTI SOFTWARE**

Per i test delle unità e di integrazione si cercherà di adoperare JUnit. JUnit è un framework interamente sviluppato in Java Open-Source. Fornisce un insieme di API che assistono gli sviluppatori nel crearti in modo semplice ed automatico dei test per il proprio software. Permette di separare i test dal codice. Fornisce una struttura di asserzioni per confrontare i risultati attesi (oracolo) con quelli ottenuti. Fornisce una interfaccia grafica molto elementare per la valutazione dei risultati dei test: barra verde se i test case in esecuzione falliscono, barra rossa se qualche test case della test suite fallisce.

Consente di definire:

* Test case: singolo test su di una specifica funzionalità (un metodo, una classe, etc.).
* Test suite: una collezione di test case che possono essere raggruppati in base a caratteristiche omogenee e possono essere eseguite in blocco.

In particolare i software che si utilizzeranno per l’attività di testing sono i seguenti:

* JUnit
* MySQL
* Apache Tomcat 8 o superiore
* Java Runtime Environment (JRE) 7 o superiore
* Java Development kit (JDK) 7 o superiore
* Java Enterprise Edition (J2EE)
* Selenium IDE per l’elecuzione dei test case;
* Browser (Google Chrome, FirefoxMozilla, Safari, Opera, Internet Explorer, etc.).

1. **TEST CASE**

I test case sono diretti a scoprire a eventuali malfunzionamenti o comportamenti errati da parte del sistema. Per fare ciò è necessario testare il sistema su diverse istanze di input, ognuna diretta a testare comportamenti del sistema in determinate condizioni, un tipo o classe di input piuttosto di un'altra. Il testing viene quindi strutturato in base alle funzionalità fornite dal sistema. Per ognuna di queste verranno fornite delle istanze di input che costituiscono i test case dei vari scenari. I test case potranno appartenere a tre diverse categorie di input in base al tipo di dati da cui sono costituiti relativamente allo use case in esame che possono essere:

* Grammaticalmente non validi: fanno parte di questa categoria gli input che contengono caratteri non validi, che sono a loro volta suddivisi in classi di equivalenza.
* Logicamente non validi: fanno parte di questa categoria i dati che potrebbero essere grammaticalmente validi ma che nel contesto non lo sono, ad esempio in un form che richiede l’inserimento di due numeri per la definizione di un range di valori, due input logicamente non validi possono essere un lowerbound (il valore inferiore del range) più alto dell’upperbaund (valori superiori del range).
* Validi: fanno parte di questa categoria i dati validi, cioè che sono grammaticalmente validi e hanno senso nel contesto in cui sono utilizzati.

Quindi per ogni funzionalità verranno presentati un insieme di test. Ognuno di questi insiemi è diviso a sua volta in tre sottoinsiemi:

* Input grammaticalmente non validi, ma logicamente validi
* Input logicamente non validi, ma grammaticalmente validi
* Input grammaticalmente e logicamente validi.

**REGISTRAZIONE**

**CLASSI DI EQUIVALENZA**

|  |  |  |
| --- | --- | --- |
| INPUT | Nome, cognome | |
| CLASSI VALIDE | C\_0001 | Stringa alfabetica |
| CLASSI NON VALIDE | C\_0002 | Stringa contiene numeri |
| C\_0003 | Stringa minore di 1 carattere. |

|  |  |  |
| --- | --- | --- |
| INPUT | Indirizzo | |
| CLASSI VALIDE | C\_0004 | Stringa alfanumerica con lunghezza compresa tra 1 e 40 caratteri. |
| CLASSI NON VALIDE | C\_0005 | Stringa maggiore di 40 caratteri. |
| C\_0006 | Stringa minore di 1 carattere. |

|  |  |  |
| --- | --- | --- |
| INPUT | Codice fiscale | |
| CLASSI VALIDE | C\_0007 | Stringa alfanumerica con lunghezza fissa di 16 caratteri. |
| CLASSI NON VALIDE | C\_0008 | Stringa alfanumerica minore di 16 caratteri. |
| C\_0009 | Stringa alfanumerica maggiore di 16 caratteri. |
| C\_0010 | Stringa alfanumerica discorde con la forma standard di un codice fiscale. |

|  |  |  |
| --- | --- | --- |
| INPUT | e-mail | |
| CLASSI VALIDE | C\_0011 | Stringa alfanumerica con formato “nomeUtente”@”provider” |
| CLASSI NON VALIDE | C\_0012 | Stringa alfanumerica discorde dal formato. |

|  |  |  |
| --- | --- | --- |
| INPUT | Password | |
| CLASSI VALIDE | C\_0013 | Stringa alfanumerica almeno di 5 caratteri |
| CLASSI NON VALIDE | C\_0014 | Stringa alfanumerica minore di 5 caratteri. |
|  | C\_0015 | Stringa alfanumerica vuota |

|  |  |  |
| --- | --- | --- |
| INPUT | Username | |
| CLASSI VALIDE | C\_0016 | Stringa alfanumerica. |
| CLASSI NON VALIDE | C\_0017 | Stringa alfanumerica vuota. |
| C\_0018 | Stringa alfanumerica uguale a 0. |

* + 1. **FORMAL TEST SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| INPUT | Nome, cognome | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0001 | | [propertynome\_cognome\_indirizzo OK] |
| C\_0002 | | [errore] |
| C\_0003 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Indirizzo | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0004 | | [propertynome\_cognome\_indirizzo OK] |
| C\_0005 | | [errore] |
| C\_0006 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Codice fiscale | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0007 | | [propertycodice\_fiscaleOK] |
| C\_0008 | | [errore] |
| C\_0009 | | [errore] |
| C\_0010 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Email | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0011 | | [propertyEmailOK] |
| C\_0012 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Password | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0013 | | [propertyPasswordOK] |
| C\_0014 | | [errore] |
| C\_0015 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Username | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0016 | | [propertyusernameOK] |
| C\_0017 | | [errore] |
| C\_0018 | | [errore] |

* + 1. **TEST CASE**

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Registrazione\_01 | |
| PARAMETRO | SCELTA | VALORE |
| Codice fiscale | C\_0007 | SSSGPP95E10H703E |
| Nome | C\_0001 | Raffaele |
| Cognome | C\_0001 | Villani |
| Email | C\_0011 | giuseppe@hotmail.it |
| Password | C\_0013 | Universita95 |
| Indirizzo | C\_0004 | Nocera Via Iroma 105 |
| Username | C\_0016 | Raff97 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Registrazione\_02 | |
| PARAMETRO | SCELTA | VALORE |
| Codice fiscale | C\_0007 | SSSGPP95E10H703E |
| Nome | C\_0003 |  |
| Cognome | C\_0001 | Villani |
| Email | C\_0011 | giuseppe@hotmail.it |
| Password | C\_0013 | Universita95 |
| Indirizzo | C\_0004 | Nocera Via Iroma 105 |
| Username | C\_0016 | Raff97 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Registrazione\_03 | |
| PARAMETRO | SCELTA | VALORE |
| Codice fiscale | C\_0007 | SSSGPP95E10H703E |
| Nome | C\_0001 | Raffaele |
| Cognome | C\_0001 | Villani |
| Email | C\_0011 | giuseppe@hotmail |
| Password | C\_0013 | Università95 |
| Indirizzo | C\_0006 |  |
| Username | C\_0016 | Raff97 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Registrazione\_04 | |
| PARAMETRO | SCELTA | VALORE |
| Codice fiscale | C\_0007 | SSSGPP95E10H703E |
| Nome | C\_0001 | Raffaele |
| Cognome | C\_0001 | Villani |
| Email | C\_0014 | giuseppe@hotmail.it |
| Password | C\_0015 |  |
| Indirizzo | C\_0004 | Nocera Via Iroma 105 |
| Username | C\_0016 | Raff97 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Registrazione\_05 | |
| PARAMETRO | SCELTA | VALORE |
| Codice fiscale | C\_0008 | SSSGPP95E10H70 |
| Nome | C\_0001 | Raffaele |
| Cognome | C\_0001 | Villani |
| Email | C\_0014 | giuseppe@hotmail.it |
| Password | C\_0013 | Universita95 |
| Indirizzo | C\_0004 | Nocera Via Iroma 105 |
| Username | C\_0016 | Raff97 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Registrazione\_05 | |
| PARAMETRO | SCELTA | VALORE |
| Codice fiscale | C\_0008 | SSSGPP95E10H70 |
| Nome | C\_0001 | Raffaele |
| Cognome | C\_0001 | Villani |
| Email | C\_0014 | giuseppe@hotmail.it |
| Password | C\_0013 | Universita95 |
| Indirizzo | C\_0004 | Nocera Via Iroma 105 |
| Username | C\_0017 |  |

* 1. **LOGIN** 
     1. **CLASSI DI EQUIVALENZA**

|  |  |  |
| --- | --- | --- |
| INPUT | Username | |
| CLASSI VALIDE | C\_0027 | Stringa alfanumerica. |
| CLASSI NON VALIDE | C\_0028 | Stringa alfanumerica vuota. |
| C\_0029 | Stringa alfanumerica uguale a 0. |

|  |  |  |
| --- | --- | --- |
| INPUT | Password | |
| CLASSI VALIDE | C\_0030 | Stringa alfanumerica almeno di 5 caratteri. |
| CLASSI NON VALIDE | C\_0031 | Stringa alfanumerica minore di 5 caratteri. |
| C\_0032 | Stringa alfanumerica vuota. |

* + 1. **FORMAL TEST SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| INPUT | Username | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0027 | | [propertyusernameOK] |
| C\_0028 | | [errore] |
| C\_0029 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Password | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0030 | | [propertyPasswordOK] |
| C\_0031 | | [errore] |
| C\_0032 | | [errore] |

* + 1. **TEST CASE**

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Login\_01 | |
| PARAMETRO | SCELTA | VALORE |
| Username | C\_0027 | admin |
| Password | C\_0030 | 123456 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Login\_02 | |
| PARAMETRO | SCELTA | VALORE |
| Username | C\_0028 |  |
| Password | C\_0030 | Babbeo12 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Login\_03 | |
| PARAMETRO | SCELTA | VALORE |
| Username | C\_0027 | Gae1998 |
| Password | C**\_**0032 |  |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Login\_04 | |
| PARAMETRO | SCELTA | VALORE |
| Username | C\_0029 | 0 |
| Password | C\_0031 | uni |

**6 CARICAMENTO PRODOTTO (GESTORE PRODOTTI)**

**9.4.1 CASI DI EQUIVALENZA**

|  |  |  |
| --- | --- | --- |
| INPUT | Nome | |
| CLASSI VALIDE | C\_0044 | Stringa alfanumerica. |
| CLASSI NON VALIDE | C\_0045 | Stringa alfanumerica uguale a 0. |
| C\_0046 | Stringa alfanumerica vuota. |

|  |  |  |
| --- | --- | --- |
| INPUT | Marchio | |
| CLASSI VALIDE | C\_0047 | Stringa alfanumerica. |
| CLASSI NON VALIDE | C\_0048 | Stringa alfanumerica uguale a 0. |
| C\_0049 | Stringa alfanumerica vuota. |

|  |  |  |
| --- | --- | --- |
| INPUT | Categoria | |
| CLASSI VALIDE | C\_0050 | Stringa alfanumerica. |
| CLASSI NON VALIDE | C\_0051 | Stringa alfanumerica uguale a 0. |
| C\_0052 | Stringa alfanumerica vuota. |

|  |  |  |
| --- | --- | --- |
| INPUT | Prezzo | |
| CLASSI VALIDE | C\_0053 | Numero decimale con due numeri dopo la virgola. |
| CLASSI NON VALIDE | C\_0054 | Carattere alfabetico all’interno del prezzo. |

|  |  |  |
| --- | --- | --- |
| INPUT | Descrizione | |
| CLASSI VALIDE | C\_0057 | Stringa alfanumerica di lunghezza arbitraria. |
| CLASSI NON VALIDE | C\_0058 | Stringa alfanumerica vuota. |

|  |  |  |
| --- | --- | --- |
| INPUT | Quantità | |
| CLASSI VALIDE | C\_0059 | Numero intero maggiore o uguale a 0. |
| CLASSI NON VALIDE | C\_0060 | Numero intero minore di 0. |
|  | C\_0061 | Carattere alfabetico all’interno della quantità. |

**9.4.2. FORMAL TEST SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| INPUT | Nome | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0044 | | [propertycodice\_pezzoOK] |
| C\_0045 | | [errore] |
| C\_0046 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Marchio | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0047 | | [propertymarchioOK] |
| C\_0048 | | [errore] |
| C\_0049 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Categoria | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0050 | | [propertymodelloOK] |
| C\_0051 | | [errore] |
| C\_0052 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Prezzo | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0053 | | [propertyPrezzoOK] |
| C\_0054 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Descrizione | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0057 | | [propertyDescrizioneOK] |
| C\_0058 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Quantità | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0059 | | [propertyQuantitàOK] |
| C\_0060 | | [errore] |
| C\_0061 | | [errore] |

**9.4.3. TEST CASE**

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Caricamento\_01 | |
| PARAMETRO | SCELTA | VALORE |
| Nome | C\_0044 | Batteria 12A |
| Marchio | C\_0047 | Duracell |
| Categoria | C\_0050 | batterie |
| Prezzo | C\_0053 | 2.00 |
| Descrizione | C\_0057 | Batteria per telecomandi |
| Quantità | C\_0059 | 10 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Caricamento\_03 | |
| PARAMETRO | SCELTA | VALORE |
| Nome | C\_0044 | Arduino shield |
| Marchio | C\_0047 | arduino |
| Categoria | C\_0051 |  |
| Prezzo | C\_0053 | 18.00 |
| Descrizione | C\_0058 |  |
| Quantità | C\_0059 | 10 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Caricamento\_04 | |
| PARAMETRO | SCELTA | VALORE |
| Nome | C\_0046 |  |
| Marchio | C\_0047 | Phonocar |
| Categoria | C\_0050 | Accessori |
| Prezzo | C\_0053 | 10.15 |
| Descrizione | C\_0057 | Mascherina auto |
| Quantità | C\_0060 | -1 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Caricamento\_05 | |
| PARAMETRO | SCELTA | VALORE |
| Nome | C\_0045 | 0 |
| Marchio | C\_0048 | 0 |
| Categoria | C\_0050 | Illuminazione |
| Prezzo | C\_0053 | 15.00 |
| Descrizione | C\_0057 | Lampadina 16A |
| Quantità | C\_0059 | 10 |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_Caricamento\_06 | |
| PARAMETRO | SCELTA | VALORE |
| Nome | C\_0046 |  |
| Marchio | C\_0047 | GBC |
| Categoria | C\_0052 |  |
| Prezzo | C\_0053 | 14.60 |
| Descrizione | C\_0058 |  |
| Quantità | C\_0061 | 10ab |

**RICERCA PRODOTTO**

**CLASSI DI EQUIVALENZA**

|  |  |  |
| --- | --- | --- |
| INPUT | Nome | |
| CLASSI VALIDE | C\_0035 | Stringa alfanumerica compresa tra 1 e 20 caratteri. |
| CLASSI NON VALIDE | C\_0036 | Stringa alfanumerica minore di 1 caratteri. |
| C\_0037 | Stringa alfanumerica maggiore di 20 caratteri. |

|  |  |  |
| --- | --- | --- |
| INPUT | Categoria | |
| CLASSI VALIDE | C\_0038 | Stringa alfanumerica corrispondente a una categoria presente nel sistema. |
| CLASSI NON VALIDE | C\_0039 | Stringa alfanumerica non presente nel sistema. |
| C\_0040 | Stringa alfanumerica minore di 3 caratteri. |

**FORMAL TEST SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| INPUT | Nome | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0035 | | [propertynomeOK] |
| C\_0036 | | [errore] |
| C\_0037 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Categoria | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0035 | | [propertycategoriaOK] |
| C\_0036 | | [errore] |
| C\_0037 | | [errore] |

**TEST CASE**

|  |  |  |  |
| --- | --- | --- | --- |
| TEST CASE | TC\_RicercaProdotto\_01 | |  |
| PARAMETRO | SCELTA | VALORE | |
| Nome | C\_0035 | Arduino UNO | |
| Categoria | C\_0038 | Arduino | |

|  |  |  |  |
| --- | --- | --- | --- |
| TEST CASE | TC\_RicercaProdotto\_02 | |  |
| PARAMETRO | SCELTA | VALORE | |
| Nome | C\_0035 | MSI motherboard | |
| Categoria | C\_0039 |  | |

**PRENOTAZIONE RIPARAZIONE**

**9.4.1 CASI DI EQUIVALENZA**

|  |  |  |
| --- | --- | --- |
| INPUT | Descrizione | |
| CLASSI VALIDE | C\_0062 | Stringa alfanumerica. |
| CLASSI NON VALIDE | C\_0064 | Stringa alfanumerica vuota. |

|  |  |  |
| --- | --- | --- |
| INPUT | Categoria | |
| CLASSI VALIDE | C\_0065 | Stringa alfanumerica. |
| CLASSI NON VALIDE | C\_0067 | Stringa alfanumerica vuota. |

**9.4.2. FORMAL TEST SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| INPUT | Descrizione | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0062 | | [propertycodice\_pezzoOK] |
| C\_0064 | | [errore] |

|  |  |  |
| --- | --- | --- |
| INPUT | Categoria | |
| ID\_CODICE | | SPECIFICA FORMALE |
| C\_0065 | | [propertymarchioOK] |
| C\_0067 | | [errore] |

**9.4.3. TEST CASE**

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_PrenotazioneRiparazione \_01 | |
| PARAMETRO | SCELTA | VALORE |
| Descrizione | C\_0062 | Tv rotta |
| Categoria | C\_0065 | televisori |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_PrenotazioneRiparazione\_02 | |
| PARAMETRO | SCELTA | VALORE |
| Descrizione | C\_0062 | Radio rotta |
| Categoria | C\_0067 |  |

|  |  |  |
| --- | --- | --- |
| TEST CASE | TC\_PrenotazioneRiparazione\_02 | |
| PARAMETRO | SCELTA | VALORE |
| Descrizione | C\_0064 |  |
| Categoria | C\_0065 | televisori |

1. **TESTING SCHEDULE**

Di seguito sono elencate la gesstione dei rischi che occorre seguire durante la fase di testing, l’organizzazione delle attività di testing e la schedulazione delle attività durante il periodo stabilito per il testing.

**10.1 GESTIONE DEI RISCHI**

I possibili rischi generati dalle attività di testing sono stati minimizzati diminuendo le componenti del sistema da implementare e, quindi, testare. Inoltre, effettuando un testing di tipo funzionale viene limitato lo sviluppo di stub e driver per il testing delle singole componenti e quindi l’introduzione di nuovi errori nel nuovo codice di cui sarebbero composti. D’altro canto il testing di funzionalità rallenta l’individuazione di errori qualora un caso di test avesse esito positivo, poiché l’utilizzo di più componenti per il test di una singola funzionalità viene eseguita ma non in modo completamente corretto. Qualora la fase di testing evidenziasse un numero di errori maggiore rispetto alla media attesa, viene pianificato un impegno maggiore dei membri del team sulle attività di testing ed in casi estremi l’abbandono delle altre attività finchè errori gravi (funzionalità non corretta, risultati errati, modifiche apportate in modo errato) non vengano risolti.

* 1. **ORGANIZZAZIONE DELLE ATTIVITA’**

Le attività di testing devono svolgersi sulle singole funzionalità divise nei livelli di suddivisione del sistema, rispettando le direttive indicate dal documento di system design.