*Test Plan*

*Di Unità*

*Progetto*

*OctoPlus*



**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 22/01/2024 | **0.1** | **Prima stesura** | **Tutto il team** |
| 25/01/2024 | **0.2** | **Aggiunti ICardDao e IProdottoDao** | **Tutto il team** |

Indice

1. Introduzione
2. Relazione coi documenti
3. Componenti da testare
4. Test case

4.1 Package Gestione Utenti

4.1.1 TCU1\_1: IUserDao

4.1.2 TCU2\_1: ICardDao

4.2 Package Gestione Prodotti

4.2.1 TCU2\_1: IProdottoDao

4.2.2 TCU2\_2: IPhotoDao

4.3 Package Gestione Ordini

4.3.1 TCU3\_1: OrdineDaoDataSource

4.4 Package Carrello

4.4.1 TCU4\_1: ICarrelloDao

4.5 Package Gestione Gestore

4.5.1 TCU5\_1: IGestoreDao

1. **Introduzione**

L’obbiettivo di questo documento è quello di pianificare e progettare i test case di riferimento per l’implementazione dei test di unità.

1. **Relazioni con altri documenti**

* Test Plan: le informazioni riguardanti l’approccio che verrà utilizzato per il testing sono presenti nel Test Plan
* O.D.D: faremo riferimento alle interfacce definite nell’ODD.

1. **Componenti da testare**

- Package Gestione Utenti

* IUserDao
* ICardDao

- Package Gestione Prodotti

* IProdottoDao
* IPhotoDao

- Package Gestione Ordini

* OrdineDaoDataSource

- Package Carrello

* ICarrelloDao

- Package Gestione Gestore

* IGestoreDao

1. **Test case**

**4.1 TCU\_1\_1: IUserDAO**

**TCU1\_1\_1 doRetrieveByEmail(User user):**

|  |  |
| --- | --- |
| **Parametri** | user.email |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| Email | ER1: formato valido [property email\_corretta]  ER2: formato null or vuoto or non valido  EP1: email già presente nel DB [if email\_corretta]  EP2: email non presente nel DB if[email\_corretta] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| ER1-EP1 | Restituisce Utente user relativo all’email inserita |
| ER1-EP2 | Restituisce null |
| ER2 | Lancia una CheckException |

**TCU1\_1\_2 doRetrieveByKey(User user):**

|  |  |
| --- | --- |
| **Parametri** | user.idutente |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| idutente | ID1: formato valido [property: id\_corretto]  ID2: non valido  IP1: idutente già presente nel DB [if id\_corretto]  IP2: idutente non presente nel DB [if id\_corretto] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| ID1-IP1 | Restituisce Utente user relativo all’email inserita |
| ID1-IP2 | Restituisce null |
| ID2 | Lancia una CheckException |

**TCU1\_1\_2 doRetrieveAllUser(String order):**

|  |  |
| --- | --- |
| **Parametri** | order |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| order | OR1: order = “email”  OR2: order = “nome”  OR3: order = null  OR4: order = “”  OR5 : order non “mail” e non “nome” |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| OR1 | Restituisce una Collection<Utente> di tutti gli utenti presenti nel database in ordine di email (crescente) |
| OR2 | Restituisce una Collection<Utente> di tutti gli utenti presenti nel database in ordine di  nome (crescente) |
| OR3 | Restituisce una Collection<Utente> di tutti gli utenti presenti nel database in ordine di  idutente (crescente) |
| OR4 | Restituisce una Collection<Utente> di tutti gli utenti presenti nel database in ordine di  idutente (crescente) |
| OR5 | Lancia una CheckException |

**TCU1\_1\_3 doSaveUser (User user):**

|  |  |
| --- | --- |
| **Parametri** | user |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| User | US1: user non vuoto  US2: user vuoto |
| user.email | FE1: formato valido [property email\_corretta]  FE2: formato non valido  EP1: email non presente nel DB [if email\_corretta]  EP2: email presente nel DB [if email\_corretta] |
| user.nome | NU1: formato valido  NU2: formato non valido |
| user.cognome | CU1: formato valido  CU2: formato valido |
| user.password | LP1: lunghezza password >= 12  LP2: lunghezza password < 12 |
| user.numero\_telefono | FT1: formato numero di telefono  FT2: formato numero di telefono non valido |
| user.ruolo | FR1: formato valido  FR2: formato non valido |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| US1-FE1-EP1-NU1-CU1-LP1-FT1-FR1 | Salva l’utente nel DB |
| US1-FE2-EP1-NU1-CU1-LP1-FT1-FR1 | Lancia una CheckException |
| US1-FE1-EP2-NU1-CU1-LP1-FT1-FR1 | Lancia un SQLDBException |
| US1-FE1-EP1-NU2-CU1-LP1-FT1-FR1 | Lancia una CheckException |
| US1-FE1-EP1-NU1-CU2-LP1-FT1-FR1 | Lancia una CheckException |
| US1-FE1-EP1-NU1-CU1-LP2-FT1-FR1 | Lancia una CheckException |
| US1-FE1-EP1-NU1-CU1-LP1-FT2-FR1 | Lancia una CheckException |
| US1-FE1-EP1-NU1-CU1-LP1-FT1-FR2 | Lancia una CheckException |
| US2 | Lancia una NullException |

**TCU1\_1\_4 changePass(User user, String pass):**

|  |  |
| --- | --- |
| **Parametri** | User, pass |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| user | US1: user non vuoto  US2: user vuoto |
| user.email | FE1: formato valido [property email\_corretta]  FE2: formato non valido  EM1: email già presente [if email\_corretta]  EM2: email non presente [if email\_corretta] |
| pass | FP1: formato valido  FP2: formato non valido |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| US1-FE1-EM1-FP1 | Cambia il campo password relativo all’utente con email corrispondente all’email passata |
| US1-FE2-EM1-FP1 | Lancia una CheckException |
| US1-FE1-EM2-FP1 | Lancia una CheckException |
| US1-FE1-EM1-FP2 | Lancia una CheckException |
| US2 | Lancia una CheckException |

**4.2TCU2\_1: ICardDao**

**TCU2\_1 cancellaCarta():**

|  |  |
| --- | --- |
| **Parametri** | Carta |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| carta | CS1: carta non vuoto  CS2: carta vuoto |
| Carta.idCarta | CF1: formato valido [property id\_corretto]  CF2: formato non valido  CV1: id già presente [if id\_corretto]  CV2: id non presente [if id\_corretto] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| CS1-CF1-CV1 | Cancella la carta dal database |
| CS1-CF2-CV1 | Lancia una CheckException |
| CS1-CF1-CV2 | Lancia una CheckException |
| CS2 | Lancia una CheckException |

**TCU2\_1 salvaCarta():**

|  |  |
| --- | --- |
| **Parametri** | Carta |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| carta | CS1: carta non vuota  CS2: carta vuota |
| Carta.idCarta | CF1: formato valido [property id\_corretto]  CF2: formato non valido  CV1: id non presente [if id\_corretto]  CV2: id già presente [if id\_corretto] |
| Carta.proprietario | CP1: formato valido  CP2: formato non valido |
| Carta.numero\_carta | CN1: formato valido  CN2: formato non valido |
| Carta.data\_scadenza | CD1: formato valido  CD2: formato non valido |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| CS1-CF1-CV1-CP1-CN1-CD1 | Salva la carta nel DB |
| CS1-CF2-CV1-CP1-CN1-CD1 | Lancia una CheckException |
| CS1-CF1-CV2-CP1-CN1-CD1 | Sovrascrivere una carta già esistente |
| CS1-CF1-CV1-CP2-CN1-CD1 | Lancia una CheckException |
| CS1-CF1-CV1-CP2-CN2-CD1 | Lancia una CheckException |
| CS1-CF1-CV1-CP2-CN1-CD2 | Lancia una CheckException |
| CS2 | Lancia una CheckException |

**TCU2\_1 recuperaCarta():**

|  |  |
| --- | --- |
| **Parametri** | Carta |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| carta | CS1: carta non vuoto  CS2: carta vuoto |
| Carta.idCarta | CF1: formato valido [property id\_corretto]  CF2: formato non valido  CV1: id non presente [if id\_corretto]  CV2: id già presente [if id\_corretto] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| CS1-CF1-CV2 | Recupera la carta dal DB |
| CS1-CF2-CV2 | Lancia una CheckException |
| CS1-CF1-CV1 | null |

**4.3 TCU\_1\_2: IProdottoDAO**

**TCU1\_2\_1 doSave(Product product):**

|  |  |
| --- | --- |
| **Parametri** | product |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| Product | PR1: product non vuoto  PR2: product vuoto |
| product.nome | NP1: formato valido  NP2: formato non valido |
| product.categoria | CAT1: formato valido  CAT2: formato non valido |
| product.prezzo | P1: prezzo>0  P2: prezzo<=0 |
| product.descrizione | DP1: formato valido  DP2: formato valido |
| product.image | IMG1: formato valido  IMG: formato non valido |
| product.statistiche | ST1: formato valido  ST2: formato non valido |
| product.taglia | TF1: formato valido [property taglie\_corrette]  TF2: formato non valido  TP1: Taglie.S +Taglie.M + Taglie.L + Taglie.XL >=0 [if taglie\_corrette]  TP2: Taglie.S +Taglie.M + Taglie.L + Taglie.XL<0 [if taglie\_corrette] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| PR1-NP1-CAT1-P1-DP1-IMG1-ST1-TF1-TP1 | Salva il prodotto nel DB |
| PR1-NP2-CAT1-P1-DP1-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-NP1-CAT2-P1-DP1-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-NP1-CAT1-P2-DP1-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-NP1-CAT1-P1-DP2-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-NP1-CAT1-P1-DP1-IMG2-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-NP1-CAT1-P1-DP1-IMG1-ST2-TF1-TP1 | Lancia una CheckException |
| PR1-NP1-CAT1-P1-DP1-IMG1-ST1-TF2-TP1 | Lancia una CheckException |
| PR1-NP1-CAT1-P1-DP1-IMG1-ST1-TF1-TP2 | Lancia una CheckException |
| PR2 | Lancia una NullException |

**TCU1\_2\_2 doRetrieveByKey(Prodotto product):**

|  |  |
| --- | --- |
| **Parametri** | prodotto.idProdotto |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| idProdotto | ID1: formato valido [property: id\_corretto]  ID2: formato non valido  IP1: idProdotto già presente nel DB [if id\_corretto]  IP2: idProdotto non presente nel DB [if id\_corretto] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| ID1-IP1 | Restituisce Prodotto product relativo all’id inserito |
| ID1-IP2 | Restituisce null |
| ID2 | Lancia una CheckException |

**TCU1\_2\_3 doRetrieveAllProducts(String order):**

|  |  |
| --- | --- |
| **Parametri** | order |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| order | OR1: order = “categoria”  OR2: order = “prezzo”  OR3: order = null  OR4: order = “”  OR5: order non “categoria” e non “prezzo” |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| OR1 | Restituisce una Collection<Prodotto> di i prodotti presenti nel database in ordine di categoria |
| OR2 | Restituisce una Collection<Prodotto> di i prodotti presenti nel database in ordine di prezzo (crescente) |
| OR3 | Restituisce una Collection<Prodotto> di i prodotti presenti nel database in ordine di idProdotto (crescente) |
| OR4 | Restituisce una Collection<Prodotto> di i prodotti presenti nel database in ordine di idProdotto (crescente) |
| OR5 | Lancia una CheckException |

**TCU1\_2\_4 doDelete(Product product):**

|  |  |
| --- | --- |
| **Parametri** | Prodotto |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| prodotto | P1: prodotto non vuoto  P2: prodotto vuoto |
| Prodotto.idProdotto | IPF1: formato valido [property: id\_corretto]  IPF2: formato non valido  IP1: idProdotto già presente nel DB [if id\_corretto]  IP2: idProdotto non presente nel DB [if id\_corretto] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| P1-IPF1-IP1 | Il prodotto viene rimosso dal DB |
| P1-IPF2-IP1 | Lancia una CheckException |
| P1-IPF1-IP2 | null |
| P2 | Lancia una CheckException |

**TCU1\_2\_5 doUpdate(Product product):**

|  |  |
| --- | --- |
| **Parametri** | Prodotto |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| prodotto | PR1: prodotto non vuoto  PR2: prodotto vuoto |
| product.id | ID1: formato valido [property: id\_corretto]  ID2: formato non valido  IP1: idProdotto presente nel DB [if id\_corretto]  IP2: idProdotto non presente nel DB [if id\_corretto] |
| product.nome | NP1: formato valido  NP2: formato non valido |
| product.categoria | CAT1: formato valido  CAT2: formato non valido |
| product.prezzo | P1: prezzo>0  P2: prezzo<=0 |
| product.descrizione | DP1: formato valido  DP2: formato valido |
| product.image | IMG1: formato valido  IMG: formato non valido |
| product.statistiche | ST1: formato valido  ST2: formato non valido |
| product.taglia | TF1: formato valido [property taglie\_corrette]  TF2: formato non valido  TP1: Taglie.S +Taglie.M + Taglie.L + Taglie.XL >=0 [if taglie\_corrette]  TP2: Taglie.S +Taglie.M + Taglie.L + Taglie.XL<0 [if taglie\_corrette] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| PR1-ID1-IP1-NP1-CAT1-P1-DP1-IMG1-ST1-TF1-TP1 | Salva le modifiche del prodotto nel DB |
| PR1-ID2-IP1-NP1-CAT1-P1-DP1-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-ID1-IP2-NP1-CAT1-P1-DP1-IMG1-ST1-TF1-TP1 | null |
| PR1-ID1-IP1-NP2-CAT1-P1-DP1-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-ID1-IP1-NP1-CAT2-P1-DP1-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-ID1-IP1-NP1-CAT1-P2-DP1-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-ID1-IP1-NP1-CAT1-P1-DP2-IMG1-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-ID1-IP1-NP1-CAT1-P1-DP1-IMG2-ST1-TF1-TP1 | Lancia una CheckException |
| PR1-ID1-IP1-NP1-CAT1-P1-DP1-IMG1-ST2-TF1-TP1 | Lancia una CheckException |
| PR1-ID1-IP1-NP1-CAT1-P1-DP1-IMG1-ST1-TF2-TP1 | Lancia una CheckException |
| PR1-ID1-IP1-NP1-CAT1-P1-DP1-IMG1-ST1-TF1-TP2 | Lancia una CheckException |
| PR2 | Lancia una NullException |

**TCU1\_2\_5 setTaglieByKey(Taglie taglie):**

|  |  |
| --- | --- |
| **Parametri** | Taglie |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| prodotto | PR1: prodotto non vuoto  PR2: prodotto vuoto |
| product.id | ID1: formato valido [property: id\_corretto]  ID2: formato non valido  IP1: idProdotto presente nel DB [if id\_corretto]  IP2: idProdotto non presente nel DB [if id\_corretto] |
| product.taglia | TF1: formato valido [property taglie\_corrette]  TF2: formato non valido |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| PR1-ID1-IP1-TF1 | Le taglie del prodotto vengono aggiornate |
| PR1-ID2-IP1-TF1 | Lancia una CheckException |
| PR1-ID1-IP2-TF1 | null |
| PR1-ID1-IP1-TF2 | Lancia una CheckException |
| PR2 | Lancia una CheckException |

**TCU1\_2\_6 getTaglieByKey(Product product):**

|  |  |
| --- | --- |
| **Parametri** | Prodotto |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| prodotto | PR1: prodotto non vuoto  PR2: prodotto vuoto |
| product.id | ID1: formato valido [property: id\_corretto]  ID2: formato non valido  IP1: idProdotto presente nel DB [if id\_corretto]  IP2: idProdotto non presente nel DB [if id\_corretto] |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| PR1-ID1-IP1 | Restituisce una Collection<Taglia> di taglie presenti nel database per il prodotto specificato |
| PR1-ID2-IP1 | Lancia una CheckException |
| PR1-ID1-IP2 | null |
| PR2 | Lancia una CheckException |

**TCU1\_2\_7 decraseTaglie(Taglia taglie):**

|  |  |
| --- | --- |
| **Parametri** | Taglia |
| **Oggetti dell’ambiente** | Database |
| **Categorie** | **Scelte** |
| product.taglia | T1: taglie>=0  T2: taglie<0 |

**Test Frame**

|  |  |
| --- | --- |
| **Combinazione** | **Oracolo** |
| T1 | Operazione eseguita |
| T2 | Impossibile eseguire l’operazione |