*Test Plan*

*Di Unità*

*Progetto*

*OctoPlus*



**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 22/01/2024 | **0.1** | **Prima stesura** | **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**

* T.P. : 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] |

**\*/ho deciso di accorpare “formato non valido/vuoto/null” come category partition poiché in questo caso vanno a restituire lo stesso risultato, ossia il lancio di una eccezione per il controllo sul formato. In generale questo accorpamento l’ho fatto per tutti i prossimi dati.**

**\*/**

**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 doSaveUse(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 |

**//Non ho messo login perché essenzialmente è un retrieveByEmail, pertanto ho minimizzato il più possibile i metodi da analizzare poiché sono uguali e ridondanti.**

**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 |