

Università degli Studi di Salerno
Corso di Ingegneria del Software

MalTour
Test Execution Report
Versione 1.0



Data: 25/12/2021

Progetto: MalTour	Versione: 2.0
Documento: SSD	Data: 25/11/2021

Coordinatore del progetto:

Nome	Matricola
Teresa Vitagliano	052105622

Partecipanti:

Nome	Matricola
Teresa Vitagliano	0512105622

Scritto da:	Teresa Vitagliano
--------------------	-------------------

Revision History

Data	Versione	Descrizione	Autore
04/01/2022	1.0	stesura del documento	Teresa Vitagliano

1.	Introduzione.....	3
2.	Relazioni con altri documenti.....	3
3.	Approccio.....	3
4.	Test Generale	4
1.1.	UtenteTest	5
1.2.	OffertaTest	5
1.3.	ServizioTest.....	6
1.4.	OrdineTest.....	6

1. Introduzione

Per testing di unità si intende il testing delle singole unità software del sistema. Il test verrà effettuato con il framework JUnit insieme alla libreria Mockito.

2. Relazioni con altri documenti


Per Individuare i test da effettuare si utilizzerà la tecnica Black-Box, quindi ci baseremo sui documenti prodotti precedentemente:




























- TP_Maltour
- TC_Maltour

3. Approccio

Il testing di unità verrà diviso in testing DAO, testing Bean e testing Servlet, dove verranno testati singolarmente i DAO, i bean e le Servlet del sistema.

4. Test Generale

Runs: 22/28  Errors: 6

- ▼  UtenteTest [Runner: JUnit 5]
 -  testUtente()
- ▼  OrdineTest [Runner: JUnit 5]
 -  testOrdine()
- ▼  ServizioTest [Runner: JUnit 5]
 -  testCategoriaBean()
- ▼  OffertaTest [Runner: JUnit 5]
 -  testProdotto()
- ▼  LoginTest [Runner: JUnit 5]
 -  testLogin()
- ▼  CarrelloTest [Runner: JUnit 5]
 -  testCarrelloBean()
- ▼  UtenteDAOTest [Runner: JUnit 5] (0,686 s)
 -  doSave (0,676 s)
 -  doRetrieveByUsername (0,002 s)
 -  doRetrieveById (0,001 s)
 -  doRetrieveByUsernamePassword (0,002 s)
 -  doRetrieveByEmail (0,001 s)
 -  doDelete (0,002 s)
 -  doRetrieveAll (0,002 s)
- ▼  OffertaDAOTest [Runner: JUnit 5] (0,039 s)
 -  doSave (0,005 s)
 -  doRetrieveById (0,005 s)
 -  doDelete (0,002 s)
 -  doRetrieveByServizio (0,010 s)
 -  doUpdate (0,004 s)
 -  doRetrieveByNomeOrDescrizione (0,001 s)
 -  doRetrieveAll (0,012 s)
- ▼  ServizioDAOTest [Runner: JUnit 5] (0,011 s)
 -  doSave (0,002 s)
 -  doDelete (0,003 s)
 -  doUpdate (0,002 s)
 -  doRetrieveByDescrizione (0,001 s)
 -  doRetrieveByNome (0,002 s)
 -  doRetrieveAll (0,001 s)
- ▼  OrdineDAOTest [Runner: JUnit 5] (0,012 s)
 -  doSave (0,003 s)
 -  doRetrieveByUtente (0,009 s)

1.1. UtenteTest

The screenshot displays the UtenteDAOTest.java file in an IDE. The code defines a class UtenteDAO with methods for retrieving and saving users. The test class UtenteDAOTest contains several test methods: doRetrieveAll(), doRetrieveById(), doRetrieveByUsernamePassword(), doRetrieveByUsername(), doRetrieveByEmail(), doSave(), doDelete(), and doRetrieveAll(). The test results show that all tests passed successfully.

```
package Test;
import org.junit.Test;

public class UtenteDAOTest {
    UtenteDAO utenteDAO = new UtenteDAO();
    Utente utenteSucc = new Utente();
    List<Utente> utente = new ArrayList<>();
    Utente utenteDaSalvare = new Utente(7, "Giovanni81", "Giov234", "Giovanni", "giovanni@gmail.com", 1);
    Utente utenteToUpdate = new Utente(2, "Giovanni81", "Giov234", "Giovanni", "serenagmailtour.com", 1);

    @Test
    public void doRetrieveAll() {
        assertNotEquals(utente, utenteDAO.doRetrieveAll(1, 10));
    }

    @Test
    public void doRetrieveById() {
        UtenteSucc = utenteDAO.doRetrieveById(1);
        assertEquals(1, utenteSucc.getId());
    }

    @Test
    public void doRetrieveByUsernamePassword() {
        UtenteSucc = utenteDAO.doRetrieveByUsernamePassword("Angela", "titolare81");
        assertEquals("Angela", utenteSucc.getUsername());
    }

    @Test
    public void doRetrieveByUsername() {
        UtenteSucc = utenteDAO.doRetrieveByUsername("Angela");
        assertEquals("Angela", utenteSucc.getUsername());
    }

    @Test
    public void doRetrieveByEmail() {
        UtenteSucc = utenteDAO.doRetrieveByEmail("angelamaltour.com");
        assertEquals("angelamaltour.com", utenteSucc.getEmail());
    }

    @Test
    public void doSave() {
        boolean succ = utenteDAO.doSave(utenteDaSalvare);
        assertEquals(true, succ);
    }

    @Test
    public void doDelete() {
        int succ = utenteDAO.doDelete(1);
        assertEquals(1, succ);
    }
}
```

Finished after 0,879 seconds

Runs: 22/22 Errors: 0 Failures: 0

Test:UtenteDAOTest [Runner: JUnit 4] (0,012 s)

- doSave (0,002 s)
- doRetrieveByUsername (0,002 s)
- doRetrieveById (0,001 s)
- doRetrieveByUsernamePassword (0,001 s)
- doRetrieveByEmail (0,002 s)
- doDelete (0,002 s)
- doRetrieveAll (0,002 s)

1.2. OffertaTest

The screenshot displays the OffertaDAOTest.java file in an IDE. The code defines a class OffertaDAO with methods for retrieving and saving offers. The test class OffertaDAOTest contains several test methods: doRetrieveAll(), doRetrieveById(), doSave(), doUpdate(), doDelete(), doRetrieveByNomeOrDescrizione(), and doRetrieveByServizio(). The test results show that all tests passed successfully.

```
package Test;
import org.junit.Test;

public class OffertaDAOTest {
    OffertaDAO offertaDAO = new OffertaDAO();
    List<Servizio> servizi = new ArrayList<>();
    List<Offerta> offertaList = new ArrayList<>();
    Offerta offerta = new Offerta(20, "Maurilio", "meta mare", "30/01/2022", "19:30", "12/02/2022", "20:15", "roma fiumicino", "sir Semoosagur", "hotel la maison", 5, 1500, servizi);
    Offerta offertaToUpdate = new Offerta(4, "Maurilio", "meta mare", "30/01/2022", "19:30", "12/02/2022", "20:15", "roma fiumicino", "sir Semoosagur", "hotel la maison", 5, 1500, servizi);
    Offerta offertaServizio = new Offerta(6, "Nord Italia Tour", "Il tour si fa strada attraverso incantevoli località come Milano e Torino; costeggia laghi meravigliosi come quelli di Como. Smetti di pensarci troppo: fai rotta verso un");

    @Test
    public void doRetrieveAll() {
        assertNotEquals(offertaList, offertaDAO.doRetrieveAll(1, 10));
    }

    @Test
    public void doRetrieveById() {
        assertNotEquals(offertaList, offertaDAO.doRetrieveById(1));
    }

    @Test
    public void doSave() {
        int yes = offertaDAO.doSave(offerta);
        assertEquals(1, yes);
    }

    @Test
    public void doUpdate() {
        boolean succ = offertaDAO.doUpdate(offertaToUpdate);
        assertEquals(true, succ);
    }

    @Test
    public void doDelete() {
        int yes = offertaDAO.doDelete(1);
        assertEquals(1, yes);
    }

    @Test
    public void doRetrieveByNomeOrDescrizione() {
        assertEquals(offertaList, offertaDAO.doRetrieveByNomeOrDescrizione("Singapore", 1, 10));
    }

    @Test
    public void doRetrieveByServizio() {
        assertNotEquals(offertaServizio, offertaDAO.doRetrieveByServizio(1, 0, 10));
    }
}
```

Finished after 0,879 seconds

Runs: 22/22 Errors: 0 Failures: 0

Test:OffertaDAOTest [Runner: JUnit 4] (0,035 s)

- doSave (0,005 s)
- doRetrieveById (0,003 s)
- doDelete (0,003 s)
- doRetrieveByServizio (0,010 s)
- doUpdate (0,003 s)
- doRetrieveByNomeOrDescrizione (0,001 s)
- doRetrieveAll (0,010 s)

1.3. ServizioTest

The screenshot shows the `ServizioDAOTest.java` file in an IDE. The code defines a `ServizioDAO` interface and a `ServizioDAOImpl` implementation. The `ServizioDAO` interface has methods `doRetrieveAll()`, `doSave()`, `doUpdate()`, `doDelete()`, `doRetrieveByName()`, and `doRetrieveByDescrizione()`. The `ServizioDAOImpl` implementation implements these methods. The `ServizioDAOTest` class contains JUnit tests for each method. The test results show that all tests passed successfully.

```
1 package Test;
2
3 import org.junit.Test;
4
5 public class ServizioDAOTest {
6
7     ServizioDAO servizioDAO = new ServizioDAOImpl();
8     List<Servizio> servizio = new ArrayList<>();
9     Servizio servizio1 = new Servizio();
10    Servizio servizioEsistente = new Servizio(1, "TOUR", "tour enogastronomici e culturali, alla scoperta dell'Italia");
11    Servizio servizioDaSalvare = new Servizio(5, "DESTINAZIONE MONTAGNA", "Ci sono tantissimi luoghi che vi attendono, qui abbiamo realizzato una piccola guida");
12    Servizio servizioUpdate = new Servizio(3, "FERIE ESTIVE", "solo qualche giorno libero");
13
14    @Test
15    public void doRetrieveAll() {
16        Assert.assertEquals(servizio, servizioDAO.doRetrieveAll());
17    }
18
19    @Test
20    public void doSave() {
21        int succ = servizioDAO.doSave(servizioDaSalvare);
22        assertEquals(1, succ);
23    }
24
25    @Test
26    public void doUpdate() {
27        boolean succ = servizioDAO.doUpdate(servizioUpdate);
28        assertEquals(true, succ);
29    }
30
31    @Test
32    public void doDelete() {
33        int succ = servizioDAO.doDelete(2);
34        assertEquals(1, succ);
35    }
36
37    @Test
38    public void doRetrieveByName() {
39        servizio1 = servizioDAO.doRetrieveByName("MINICROCIERE");
40        assertEquals("MINICROCIERE", servizio1.getName());
41    }
42
43    @Test
44    public void doRetrieveByDescrizione() {
45        servizio1 = servizioDAO.doRetrieveByDescrizione("Hai solo qualche giorno libero e stai pensando ad una vacanza in alto mare? Di seguito ti proponiamo un elenco di mini croci");
46        assertEquals("Hai solo qualche giorno libero e stai pensando ad una vacanza in alto mare? Di seguito ti proponiamo un elenco di mini crociere. Si tratta di Crociere che dura");
47    }
48 }
```

Finished after 0,879 seconds

Runs: 22/22 Errors: 0 Failures: 0

Test.ServizioDAOTest [Runner: JUnit 4] (0,806 s)

- doSave (0,796 s)
- doDelete (0,002 s)
- doUpdate (0,002 s)
- doRetrieveByDescrizione (0,002 s)
- doRetrieveByName (0,001 s)
- doRetrieveAll (0,003 s)

1.4. OrdineTest

The screenshot shows the `OrdineDAOTest.java` file in an IDE. The code defines an `OrdineDAO` interface and an `OrdineDAOImpl` implementation. The `OrdineDAO` interface has methods `doSave()` and `doRetrieveByUtente()`. The `OrdineDAOImpl` implementation implements these methods. The `OrdineDAOTest` class contains JUnit tests for each method. The test results show that all tests passed successfully.

```
1 package Test;
2
3 import org.junit.Test;
4
5 public class OrdineDAOTest {
6
7     OrdineDAO ordineDAO = new OrdineDAOImpl();
8     ArrayList<ComponentiViaggio> componenti = new ArrayList<>();
9     List<Ordine> ordineList = new ArrayList<>();
10    Ordine ordineDaSalvare = new Ordine(4, 2, 4, "serena@malto.com", false, componenti);
11
12    @Test
13    public void doSave() {
14        int yes = ordineDAO.doSave(ordineDaSalvare);
15        assertEquals(1, yes);
16    }
17
18    @Test
19    public void doRetrieveByUtente() {
20        Assert.assertEquals(ordineList, ordineDAO.doRetrieveByUtente(2));
21    }
22 }
```

Finished after 0,879 seconds

Runs: 22/22 Errors: 0 Failures: 0

Test.ServizioDAOTest [Runner: JUnit 4] (0,806 s)

Test.UtenteDAOTest [Runner: JUnit 4] (0,012 s)

Test.OfferaDAOTest [Runner: JUnit 4] (0,035 s)

Test.OrdineDAOTest [Runner: JUnit 4] (0,004 s)

- doSave (0,003 s)
- doRetrieveByUtente (0,001 s)

