



UNIVERSITÀ DEGLI STUDI DI SALERNO

Corso di Laurea in Informatica, a.a. 2021-22

Progetto del corso di Ingegneria del Software

prof. A. De Lucia, dott. F. Pecorelli

Repository GitHub: [/is-shodan-21-22/](#)

SHODAN

- › Project Proposal
- › Problem Statement
- › Requirement Analysis Document
- › System Design Document
- › Object Design Document
- › Test Plan
- › Test Execution Report
- › Test Summary Report**
- › Teamwork Report

Shodan [IS-2021/2022-DE LUCIA]

Partecipanti del progetto

Nome	Matricola	E-mail
Antonio Gravino	05121 07161	a.gravino3@studenti.unisa.it
Dario Trinchese	05121 07479	d.trinchese2@studenti.unisa.it
Raffaele Zheng	05121 09015	r.zheng@studenti.unisa.it
Carmine Fabbri	05121 07353	c.fabbri@studenti.unisa.it
Carmine Napolitano	05121 06417	c.napolitano44@studenti.unisa.it

Questo documento tratta esclusivamente il Test Summary Report.

Per ulteriore documentazione valida ai fini dell'esame, consultare la repository [/deliverables/](#).

Relazione con altri documenti

Per individuare i test da effettuare si utilizzeranno la tecnica Black-Box e White-Box, quindi ci baseremo sui nostri documenti prodotti:

- Test Plan
- Test Execution Report

Introduzione

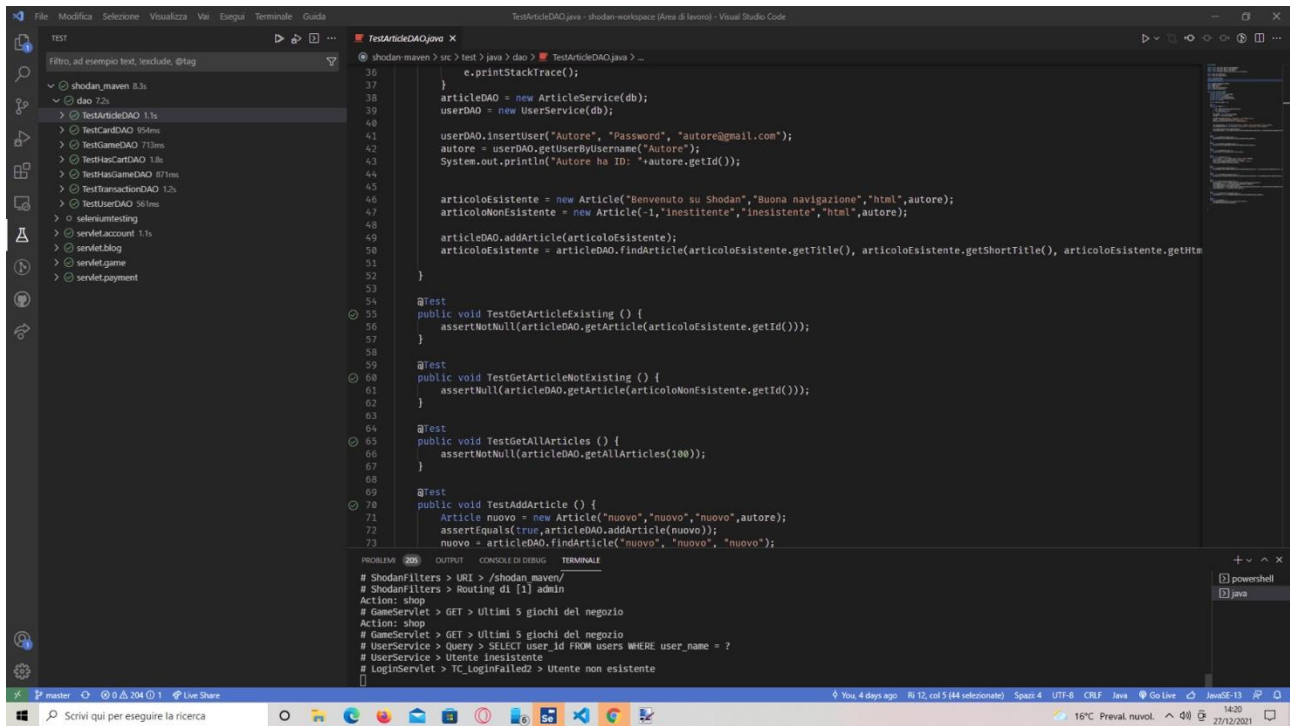
Il Testing di unità consiste nel testare le singole unità del software del sistema suddivisi in sottosistemi individuali, con l'obiettivo di testare che ogni sottosistema è stato codificato correttamente.

Approccio

Il Testing di unità verrà diviso in Testing DAO, Testing Servlet e Testing tramite Selenium. Di seguito, sono riportati alcuni screen che mostrano l'effettuato testing delle componenti.

Testing DAO

Test Article



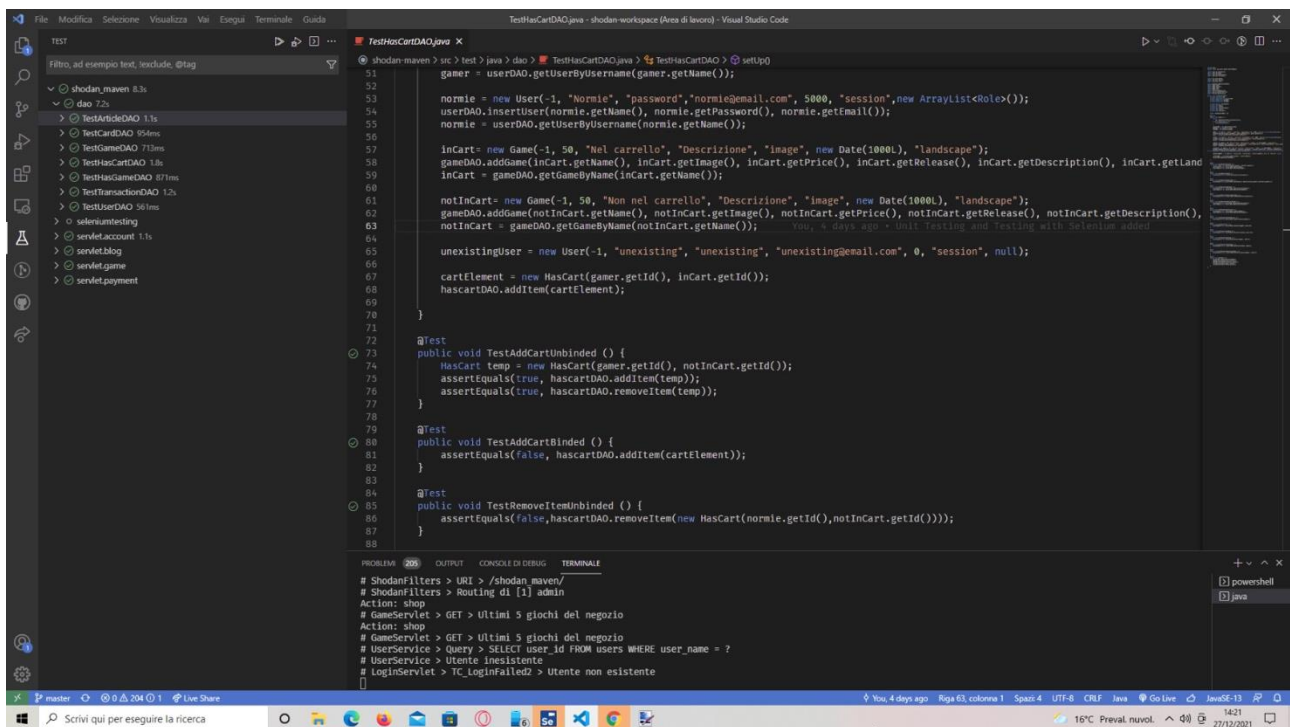
```
TestArticleDAO.java - shodan-workspace (Area di lavoro) - Visual Studio Code

TEST
Filtro, ad esempio test, testcode, @tag
▼ shodan_maven 8.3s
  ▼ dao 7.2s
    ▼ TestArticleDAO 1.1s
      ▼ TestCardDAO 95ms
      ▼ TestGameDAO 73ms
      ▼ TestHasCardDAO 1.8s
      ▼ TestHasGameDAO 87ms
      ▼ TestTransactionDAO 1.2s
      ▼ TestUserDAO 36ms
      ▼ seleniumtesting
      ▼ servlet.account 1.1s
      ▼ servlet.blog
      ▼ servlet.game
      ▼ servlet.payment

TestArticleDAO.java
36 e.printStackTrace();
37 }
38 articleDAO = new ArticleService(db);
39 userDao = new UserService(db);
40
41 userDao.insertUser("Autore", "Password", "autore@gmail.com");
42 autore = userDao.getUserByUsername("Autore");
43 System.out.println("Autore ha ID: " + autore.getId());
44
45
46 articoloEsistente = new Article("Benvenuto su Shodan", "Buona navigazione", "html", autore);
47 articoloNonEsistente = new Article(-1, "inesistente", "inesistente", "html", autore);
48
49 articleDAO.addArticle(articoloEsistente);
50 articoloEsistente = articleDAO.findArticle(articoloEsistente.getTitle(), articoloEsistente.getShortTitle(), articoloEsistente.getHtml());
51
52 }
53
54 @Test
55 public void TestGetArticleExisting () {
56     assertNotNull(articleDAO.getArticle(articoloEsistente.getId()));
57 }
58
59 @Test
60 public void TestGetArticleNotExisting () {
61     assertNull(articleDAO.getArticle(articoloNonEsistente.getId()));
62 }
63
64 @Test
65 public void TestGetAllArticles () {
66     assertNotNull(articleDAO.getAllArticles(100));
67 }
68
69 @Test
70 public void TestAddArticle () {
71     Article nuovo = new Article("nuovo", "nuovo", "nuovo", autore);
72     assertEquals(true, articleDAO.addArticle(nuovo));
73     nuovo = articleDAO.findArticle("nuovo", "nuovo", "nuovo");
74 }

PROBLEMI OUTPUT CONSOLE DI DEBUG TERMINALE
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_loginFailed2 > Utente non esistente
```

Test Has Cart



```
TestHasCartDAO.java - shodan-workspace (Area di lavoro) - Visual Studio Code

TEST
Filtro, ad esempio test, testcode, @tag
▼ shodan_maven 8.3s
  ▼ dao 7.2s
    ▼ TestArticleDAO 1.1s
    ▼ TestCardDAO 95ms
    ▼ TestGameDAO 73ms
    ▼ TestHasCardDAO 1.8s
    ▼ TestHasGameDAO 87ms
    ▼ TestTransactionDAO 1.2s
    ▼ TestUserDAO 36ms
    ▼ seleniumtesting
    ▼ servlet.account 1.1s
    ▼ servlet.blog
    ▼ servlet.game
    ▼ servlet.payment

TestHasCartDAO.java
51 gamer = userDao.getUserByUsername(gamer.getName());
52
53 normie = new User(-1, "Normie", "password", "normie@gmail.com", 5000, "session", new ArrayList<Role>());
54 userDao.insertUser(normie.getName(), normie.getPassword(), normie.getEmail());
55 normie = userDao.getUserByUsername(normie.getName());
56
57 inCart = new Game(-1, 50, "Nel carrello", "Descrizione", "image", new Date(1000L), "landscape");
58 gameDAO.addGame(inCart.getName(), inCart.getImage(), inCart.getPrice(), inCart.getRelease(), inCart.getDescription(), inCart.getId());
59 inCart = gameDAO.getGameByName(inCart.getName());
60
61 notInCart = new Game(-1, 50, "Non nel carrello", "Descrizione", "image", new Date(1000L), "landscape");
62 gameDAO.addGame(notInCart.getName(), notInCart.getImage(), notInCart.getPrice(), notInCart.getRelease(), notInCart.getDescription(), notInCart.getId());
63 notInCart = gameDAO.getGameByName(notInCart.getName());
64
65 unexistingUser = new User(-1, "unexisting", "unexisting", "unexisting@gmail.com", 0, "session", null);
66
67 cartElement = new HasCart(gamer.getId(), inCart.getId());
68 hascartDAO.addItem(cartElement);
69
70 }
71
72 @Test
73 public void TestAddCartUnbinded () {
74     HasCart temp = new HasCart(gamer.getId(), notInCart.getId());
75     assertEquals(true, hascartDAO.addItem(temp));
76     assertEquals(true, hascartDAO.removeItem(temp));
77 }
78
79 @Test
80 public void TestAddCartBinded () {
81     assertEquals(false, hascartDAO.addItem(cartElement));
82 }
83
84 @Test
85 public void TestRemoveItemUnbinded () {
86     assertEquals(false, hascartDAO.removeItem(new HasCart(normie.getId(), notInCart.getId())));
87 }
88 }

PROBLEMI OUTPUT CONSOLE DI DEBUG TERMINALE
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_loginFailed2 > Utente non esistente
```

Test Card

The screenshot shows the Visual Studio Code editor with the `TestCardDAO.java` file open. The file contains the following code:

```
1 // ...
2
3 cardDAO = new CardService(db);
4 userDao = new UserService(db);
5
6 OwnerUser = new User(-3, "Owner", "password", "owner@gmail.com", 5000, "session", new ArrayList<Role>());
7 userDao.insertUser(OwnerUser.getName(), OwnerUser.getPassword(), OwnerUser.getEmail());
8 OwnerUser = userDao.getUserByUsername(OwnerUser.getName());
9
10 poorUser = new User(-4, "Poor", "password", "poor@gmail.com", 0, "session", new ArrayList<Role>());
11 userDao.insertUser(poorUser.getName(), poorUser.getPassword(), poorUser.getEmail());
12 poorUser = userDao.getUserByUsername(poorUser.getName());
13
14 cartaEsistente = new Card(-3, "VISA", 5333171065934522L, "Owner", new Date(1000L), OwnerUser);
15 cartaNonEsistente = new Card(-4, "VISA", 1234567891234567L, "unexisting", new Date(1000L), OwnerUser);
16 cardDAO.insertCard(cartaEsistente);
17 cartaEsistente = cardDAO.getCardByNumber(cartaEsistente.getCard_number());
18
19 System.out.println("Carta esistente ha id: " + cartaEsistente.getCard_id());
20 System.out.println("Owner ha id: " + OwnerUser.getId());
21 System.out.println("Poor ha id: " + poorUser.getId());
22
23
24 @Test
25 public void TestGetCardExisting () {
26     assertNotNull(cardDAO.getCard(cartaEsistente.getCard_id()));
27 }
28
29 @Test
30 public void TestGetCardNotExisting () {
31     assertNull(cardDAO.getCard(cartaNonEsistente.getCard_id()));
32 }
33
34 @Test
35 public void TestGetCardByNumberExisting () {
36     assertNotNull(cardDAO.getCardByNumber(cartaEsistente.getCard_number()));
37 }
38
39 @Test
40 public void TestGetCardByNumberNotExisting () {
41     assertNull(cardDAO.getCardByNumber(cartaNonEsistente.getCard_number()));
42 }
43
44 // ...
```

The test results are shown in the bottom panel, indicating that all tests passed:

```
PROBLEMI 0 OUTPUT CONSOLE DI DEBUG TERMINALE
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_LoginFailed2 > Utente non esistente
```

Test Transaction

The screenshot shows the Visual Studio Code editor with the `TestTransactionDAO.java` file open. The file contains the following code:

```
1 // ...
2
3 userDao.insertUser(gamer.getName(), gamer.getPassword(), gamer.getEmail());
4 gamer = userDao.getUserByUsername(gamer.getName());
5
6 normie = new User(-1, "Normie", "password", "normie@gmail.com", 5000, "session", new ArrayList<Role>());
7 userDao.insertUser(normie.getName(), normie.getPassword(), normie.getEmail());
8 normie = userDao.getUserByUsername(normie.getName());
9
10 bought = new Game(-1, 50, "Nel carrello", "Descrizione", "image", new Date(1000L), "landscape");
11 gameDAO.addGame(bought.getName(), bought.getImage(), bought.getPrice(), bought.getDescription(), bought.getLand);
12 bought = gameDAO.getGameByName(bought.getName());
13
14 notBought = new Game(-1, 50, "Non nel carrello", "Descrizione", "image", new Date(1000L), "landscape");
15 gameDAO.addGame(notBought.getName(), notBought.getImage(), notBought.getPrice(), notBought.getDescription(), notBought.getLand);
16 notBought = gameDAO.getGameByName(notBought.getName());
17
18 // ...
19 unexistingUser = new User(-1, "unexisting", "unexisting", "unexisting@gmail.com", 0, "session", null);
20
21 transaction = new Transaction(gamer, bought, new Date(1000L), bought.getPrice());
22 transactionDAO.insertTransaction(transaction);
23
24
25 @Test
26 public void TestGetTransactionsByUser () {
27     assertEquals(true, transactionDAO.getTransactionsByUser(gamer).size() > 0);
28 }
29
30 @Test
31 public void TestGetTransactionByUserNotExisting () {
32     assertEquals(true, transactionDAO.getTransactionsByUser(unexistingUser).size() == 0);
33 }
34
35 @Test
36 public void TestGetTransactionByUserNoTransactions () {
37     assertEquals(true, transactionDAO.getTransactionsByUser(normie).size() == 0);
38 }
39
40 @Test
41 public void TestGetTransactions () {
42     // ...
43 }
44
45 // ...
```

The test results are shown in the bottom panel, indicating that all tests passed:

```
PROBLEMI 0 OUTPUT CONSOLE DI DEBUG TERMINALE
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_LoginFailed2 > Utente non esistente
```

Test Game

The screenshot shows the Visual Studio Code editor with the file `TestGameDAO.java` open. The editor is displaying a series of unit tests for the `TestGameDAO` class. The tests are as follows:

```
44 @Test
45 public void TestGetGameExisting() {
46     assertNotNull(gameDAO.getGame(giocoEsistente.getId()));
47 }
48
49 @Test
50 public void TestGetGameNotExisting() {
51     assertNull(gameDAO.getGame(giocoNonEsistente.getId()));
52 }
53
54 @Test
55 public void TestGetGameByNameExisting() {
56     assertNotNull(gameDAO.getGameByName(giocoEsistente.getName()));
57 }
58
59 @Test
60 public void TestGetGameByNameNotExisting() {
61     assertNull(gameDAO.getGameByName(giocoNonEsistente.getName()));
62 }
63
64 @Test
65 public void TestGetAllGamesByUserHasGames() {
66     assertNotNull(gameDAO.getAllGamesByUser(1));
67 }
68
69 @Test
70 public void TestGetAllGamesByUserHasNoGames() {
71     assertNull(gameDAO.getAllGamesByUser(2));
72 }
73
74 @Test
75 public void TestSearchGamesExisting() {
76     assertNotNull(gameDAO.searchGames(giocoEsistente.getName()));
77 }
78
79 @Test
80 public void TestSearchGamesNotExisting() {
81     assertNull(gameDAO.searchGames(giocoNonEsistente.getName()));
82 }
83 }
```

The terminal window at the bottom shows the output of the tests, indicating that all tests passed successfully. The output includes the following lines:

```
# UserService > Inserisco l'utente SampleUserName
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
L'utente esistente ha ora id: 98
# UserService > Query > SELECT * FROM users WHERE user_email = ?
Sono entrato nella tearDown
# UserService > Query > DELETE FROM users WHERE user_id = ?
```

Test Has Game

The screenshot shows the Visual Studio Code editor with the file `TestHasGameDAO.java` open. The editor is displaying a series of unit tests for the `TestHasGameDAO` class. The tests are as follows:

```
66 @Test
67 public void TestAddGameUnbinded () {
68     assertEquals(true, hasGameDAO.addGame(normieUser, sampleGame));
69     assertEquals(true, hasGameDAO.removeItem(normieUser, sampleGame));
70 }
71
72 @Test
73 public void TestAddGameBinded () {
74     assertEquals(false, hasGameDAO.addGame(gamerUser, sampleGame));
75 }
76
77 @Test
78 public void TestHasGameUnbinded () {
79     assertEquals(false, hasGameDAO.hasGame(normieUser, sampleGame));
80 }
81 }
```

The terminal window at the bottom shows the output of the tests, indicating that all tests passed successfully. The output includes the following lines:

```
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_LoginFailed2 > Utente non esistente
```

Test User

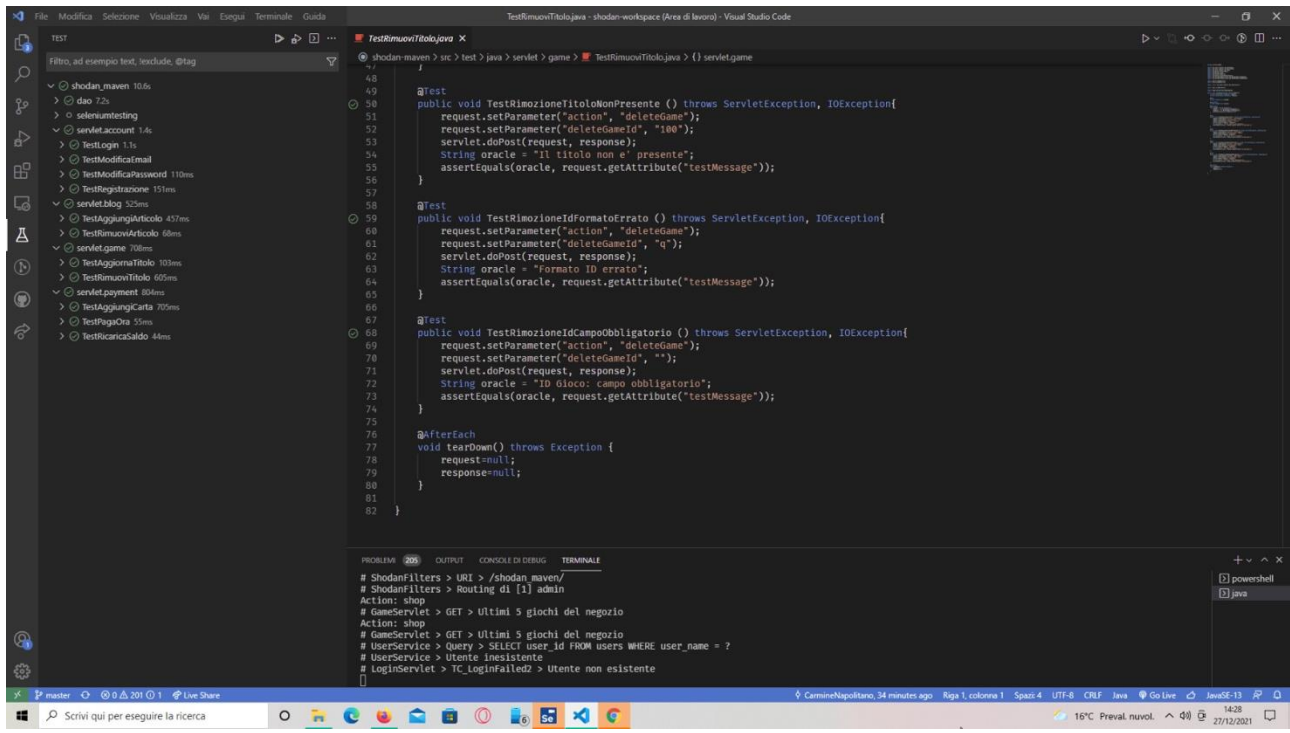
```
38 public void setUp() {
39     try {
40         db = DBConnectionPool.getConnection();
41     } catch (SQLException e) {
42         e.printStackTrace();
43     }
44     userDao = new UserService(db);
45     utenteEsistente = new User(-1, "SampleUsername", "SamplePassword", "sample@gmail.com", 500, "session", new ArrayList<Role>());
46     utenteNonEsistente = new User(-2, "unexisting", "unexisting", "unexisting@gmail.com", 500, "session", new ArrayList<Role>());
47
48     userDao.insertUser(utenteEsistente.getName(), utenteEsistente.getPassword(), utenteEsistente.getEmail());
49     utenteEsistente.setId(userDAO.getIdByUsername(utenteEsistente.getName()));
50     System.out.println("l'utente esistente ha ora id: " + utenteEsistente.getId());
51 }
52
53 @Test
54 public void testGetIdByUsernameExisting () {
55     assertEquals(-1, userDao.getIdByUsername(utenteEsistente.getName()));
56 }
57
58 @Test
59 public void testGetIdByUsernameNotExisting () {
60     assertEquals(-1, userDao.getIdByUsername(utenteNonEsistente.getName()));
61 }
62
63 @Test
64 public void testGetUserExisting () {
65     assertNotNull(userDAO.getUser(utenteEsistente.getId()));
66 }
67
68 @Test
69 public void testGetUserNotExisting () {
70     assertNull(userDAO.getUser(utenteNonEsistente.getId()));
71 }
72
73 @Test
74 public void testFindUserByEmailExisting () {
75     assertEquals(true, userDao.findUserByEmail(utenteEsistente.getEmail()));
76 }
77 }
```

PROBLEMI OUTPUT CONSOLE DI DEBUG TERMINALE

```
# ShodanFilters > URL > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_loginFailed > Utente non esistente
```


Testing Servlet

Test Rimuovi titolo

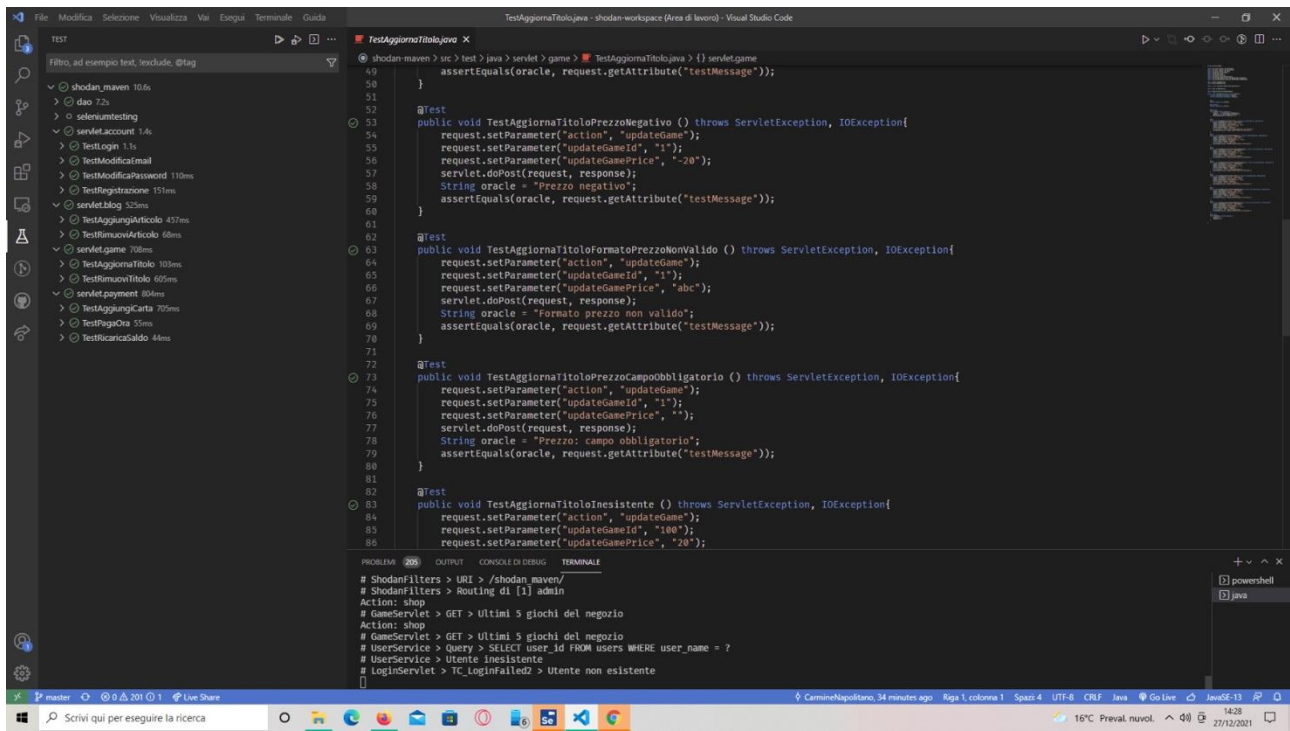


```
TEST
Filtro, ad esempio test, testSuite, @tag
▼
  shodan_maven 10.6s
  dao 7.2s
  seleniumtesting
  servlet.account 1.4s
  TestLogin 1.1s
  TestModificaEmail
  TestModificaPassword 110ms
  TestRegistrazione 151ms
  servlet.blog 525ms
  TestAggiungiArticolo 457ms
  TestRimuoviArticolo 68ms
  servlet.game 708ms
  TestAggiornaTitolo 103ms
  TestRimuoviTitolo 605ms
  servlet.payment 804ms
  TestAggiungiCarta 705ms
  TestRagione 55ms
  TestRicaricaSaldo 44ms

TestRimuoviTitolo.java
shodan-maven > src > test > java > servlet > game > TestRimuoviTitolo.java > {} servlet.game
47
48
49
50 @Test
51 public void TestRimozioneTitoloNonPresente () throws ServletException, IOException{
52     request.setParameter("action", "deleteGame");
53     request.setParameter("deleteGameId", "100");
54     servlet.doPost(request, response);
55     String oracle = "Il titolo non e' presente";
56     assertEquals(oracle, request.getAttribute("testMessage"));
57 }
58
59 @Test
60 public void TestRimozioneIdFormatoErrato () throws ServletException, IOException{
61     request.setParameter("action", "deleteGame");
62     request.setParameter("deleteGameId", "q");
63     servlet.doPost(request, response);
64     String oracle = "Formato ID errato";
65     assertEquals(oracle, request.getAttribute("testMessage"));
66 }
67
68 @Test
69 public void TestRimozioneIdCampoObbligatorio () throws ServletException, IOException{
70     request.setParameter("action", "deleteGame");
71     request.setParameter("deleteGameId", "");
72     servlet.doPost(request, response);
73     String oracle = "ID gioco: campo obbligatorio";
74     assertEquals(oracle, request.getAttribute("testMessage"));
75 }
76
77 @AfterEach
78 void tearDown() throws Exception {
79     request=null;
80     response=null;
81 }
82 }

PROBLEMI OUTPUT CONSOLE DI ERRORE TERMINALE
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > Tc_loginFailed2 > Utente non esistente
```

Test Aggiorna titolo



```
TEST
Filtro, ad esempio test, testSuite, @tag
▼
  shodan_maven 10.6s
  dao 7.2s
  seleniumtesting
  servlet.account 1.4s
  TestLogin 1.1s
  TestModificaEmail
  TestModificaPassword 110ms
  TestRegistrazione 151ms
  servlet.blog 525ms
  TestAggiungiArticolo 457ms
  TestRimuoviArticolo 68ms
  servlet.game 708ms
  TestAggiornaTitolo 103ms
  TestRimuoviTitolo 605ms
  servlet.payment 804ms
  TestAggiungiCarta 705ms
  TestRagione 55ms
  TestRicaricaSaldo 44ms

TestAggiornaTitolo.java
shodan-maven > src > test > java > servlet > game > TestAggiornaTitolo.java > {} servlet.game
49
50 assertEquals(oracle, request.getAttribute("testMessage"));
51 }
52
53 @Test
54 public void TestAggiornaTitoloPrezzoNegativo () throws ServletException, IOException{
55     request.setParameter("action", "updateGame");
56     request.setParameter("updateGameId", "1");
57     request.setParameter("updateGamePrice", "-20");
58     servlet.doPost(request, response);
59     String oracle = "Prezzo negativo";
60     assertEquals(oracle, request.getAttribute("testMessage"));
61 }
62
63 @Test
64 public void TestAggiornaTitoloFormatoPrezzoNonValido () throws ServletException, IOException{
65     request.setParameter("action", "updateGame");
66     request.setParameter("updateGameId", "1");
67     request.setParameter("updateGamePrice", "abc");
68     servlet.doPost(request, response);
69     String oracle = "Formato prezzo non valido";
70     assertEquals(oracle, request.getAttribute("testMessage"));
71 }
72
73 @Test
74 public void TestAggiornaTitoloPrezzoCampoObbligatorio () throws ServletException, IOException{
75     request.setParameter("action", "updateGame");
76     request.setParameter("updateGameId", "1");
77     request.setParameter("updateGamePrice", "");
78     servlet.doPost(request, response);
79     String oracle = "Prezzo: campo obbligatorio";
80     assertEquals(oracle, request.getAttribute("testMessage"));
81 }
82
83 @Test
84 public void TestAggiornaTitoloInesistente () throws ServletException, IOException{
85     request.setParameter("action", "updateGame");
86     request.setParameter("updateGameId", "100");
87     request.setParameter("updateGamePrice", "20");
88 }
89 }

PROBLEMI OUTPUT CONSOLE DI ERRORE TERMINALE
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > Tc_loginFailed2 > Utente non esistente
```

Test Registrazione

The screenshot shows the Visual Studio Code editor with the file `TestRegistration.java` open. The left sidebar displays a project explorer with a list of test cases under the `TEST` folder, including `TestLogin`, `TestModificaPassword`, `TestRegistrazione`, and `TestRicercaSaldo`. The main editor area contains the source code for `TestRegistration.java`, which includes several JUnit tests for the registration process. The tests are as follows:

```
@Test
public void TestRegistrazioneUsernameMancante () throws ServletException, IOException {
    request.setParameter("username", "");
    request.setParameter("password", "millionidiscala1967");
    request.setParameter("password2", "millionidiscala1967");
    request.setParameter("email", "emontale@gmail.com");
    servlet.doPost(request, response);
    String oracle = "Username campo obbligatorio";
    assertEquals(oracle, request.getAttribute("testMessage"));
}

@Test
public void TestRegistrazioneUsernameFormatoInvalido () throws ServletException, IOException {
    request.setParameter("username", "15eugenio");
    request.setParameter("password", "millionidiscala1967");
    request.setParameter("password2", "millionidiscala1967");
    request.setParameter("email", "emontale@gmail.com");
    servlet.doPost(request, response);
    String oracle = "Formato username non valido";
    assertEquals(oracle, request.getAttribute("testMessage"));
}

@Test
public void TestRegistrazioneUsernameGiaEsistente () throws ServletException, IOException {
    request.setParameter("username", "admin");
    request.setParameter("password", "millionidiscala1967");
    request.setParameter("password2", "millionidiscala1967");
    request.setParameter("email", "emontale@gmail.com");
    servlet.doPost(request, response);
    String oracle = "Username gia esistente";
    assertEquals(oracle, request.getAttribute("testMessage"));
}

@Test
public void TestRegistrazionePasswordMancante () throws ServletException, IOException {
    request.setParameter("username", "eugenioemontale");
    request.setParameter("password", "");
    request.setParameter("password2", "millionidiscala1967");
}
```

The bottom of the screen shows the `TERMINALE` (Terminal) output, which displays the results of the tests and the application's response to the registration attempts.

Test Login

The screenshot shows the Visual Studio Code editor with the file `TestLogin.java` open. The left sidebar displays a project explorer with a list of test cases under the `TEST` folder, including `TestLogin`, `TestModificaPassword`, `TestRegistrazione`, and `TestRicercaSaldo`. The main editor area contains the source code for `TestLogin.java`, which includes several JUnit tests for the login process. The tests are as follows:

```
@BeforeEach
void setUp() throws Exception {
    request = new MockHttpServletRequest();
    response = new MockHttpServletResponse();
    MockitoAnnotations.initMocks(this);
}

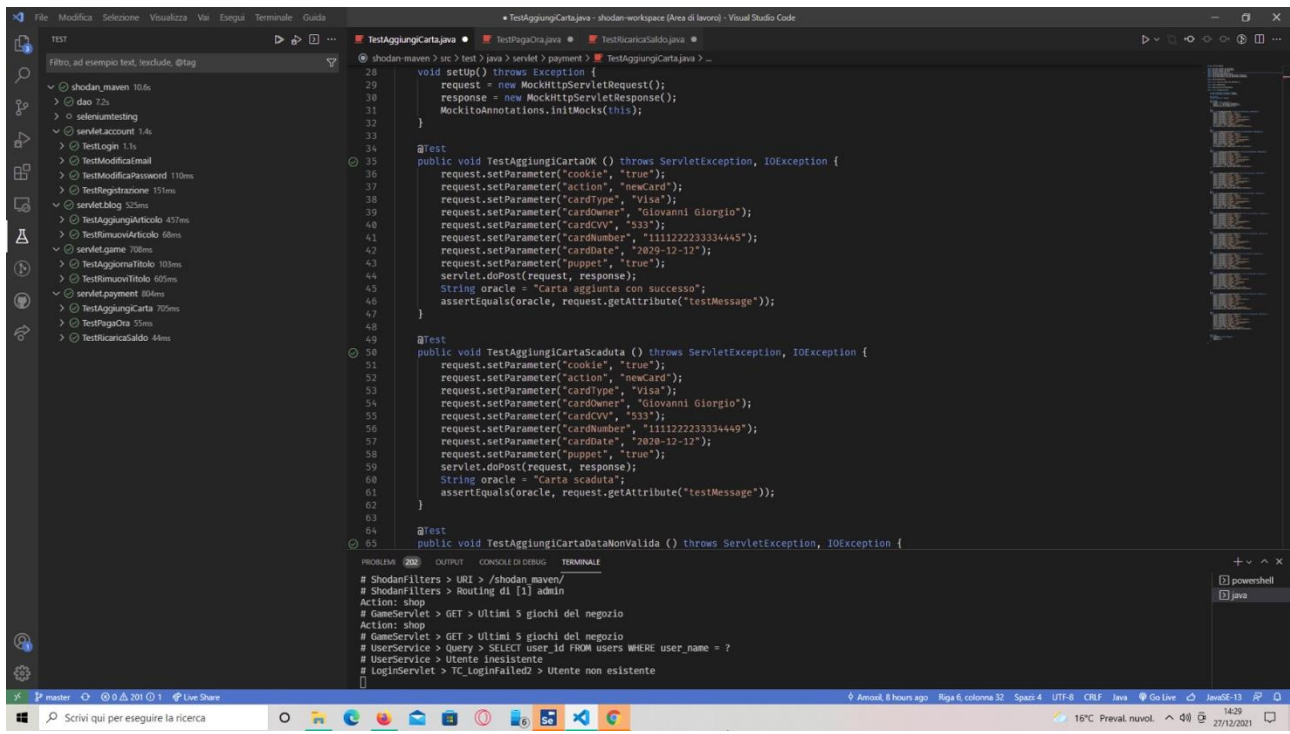
@Test
public void TestLoginCorretto () throws ServletException, IOException {
    request.setParameter("username", "admin");
    request.setParameter("password", "123");
    request.setParameter("cookie", "true");
    servlet.doPost(request, response);
    String oracle = "Login riuscito";
    assertEquals(oracle, request.getAttribute("testMessage"));
}

@Test
public void TestLoginPasswordErrata () throws ServletException, IOException {
    request.setParameter("username", "admin");
    request.setParameter("password", "124");
    request.setParameter("cookie", "true");
    servlet.doPost(request, response);
    String oracle = "Password errata";
    assertEquals(oracle, request.getAttribute("testMessage"));
}

@Test
public void TestLoginUsernameInesistente () throws ServletException, IOException {
    request.setParameter("username", "beethoven");
    request.setParameter("password", "perelisa");
    request.setParameter("cookie", "true");
    servlet.doPost(request, response);
    String oracle = "Utente non esistente";
    assertEquals(oracle, request.getAttribute("testMessage"));
}
```

The bottom of the screen shows the `TERMINALE` (Terminal) output, which displays the results of the tests and the application's response to the login attempts.

Test Aggiungi carta

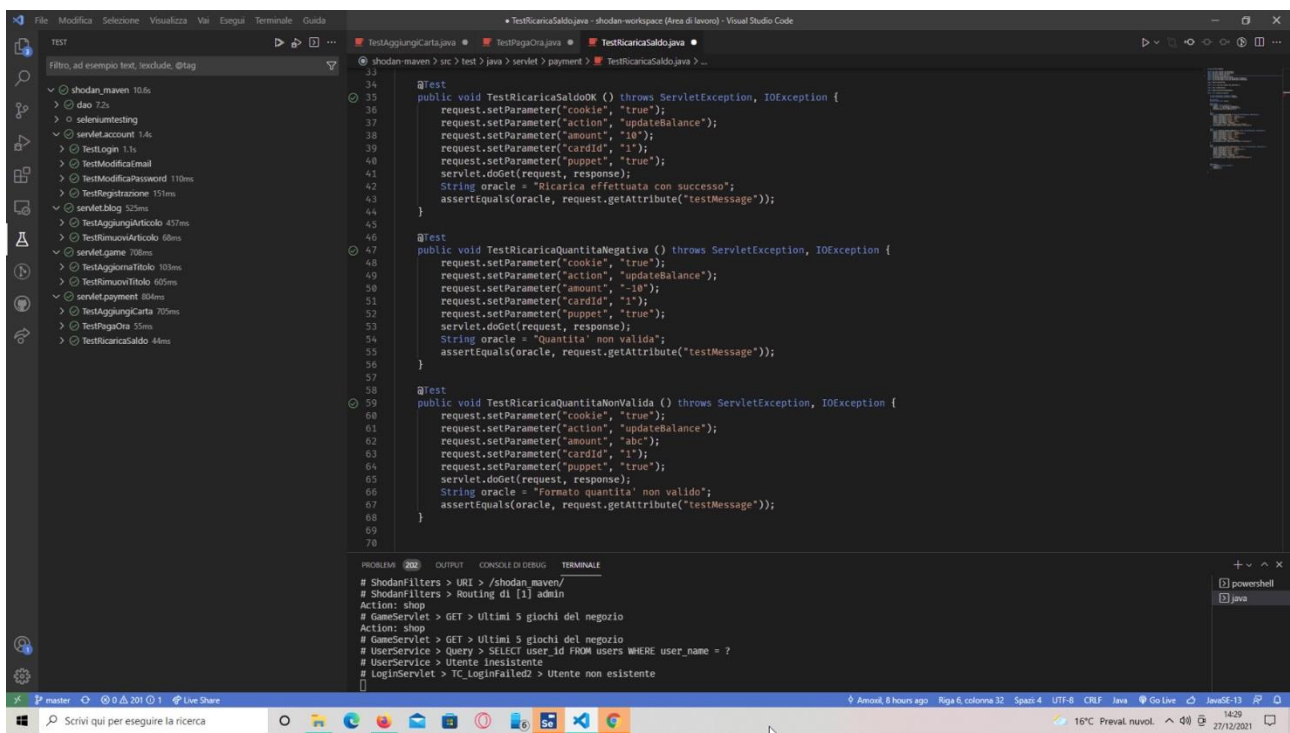


```
28 void setUp() throws Exception {
29     request = new MockHttpServletRequest();
30     response = new MockHttpServletResponse();
31     MockitoAnnotations.initMocks(this);
32 }
33
34 @Test
35 public void TestAggiungiCartaOK () throws ServletException, IOException {
36     request.setParameter("cookie", "true");
37     request.setParameter("action", "newCard");
38     request.setParameter("cardType", "Visa");
39     request.setParameter("cardOwner", "Giovanni Giorgio");
40     request.setParameter("cardCVV", "533");
41     request.setParameter("cardNumber", "11112222333344449");
42     request.setParameter("cardDate", "2020-12-12");
43     request.setParameter("puppet", "true");
44     servlet.doPost(request, response);
45     String oracle = "Carta aggiunta con successo";
46     assertEquals(oracle, request.getAttribute("testMessage"));
47 }
48
49 @Test
50 public void TestAggiungiCartaScaduta () throws ServletException, IOException {
51     request.setParameter("cookie", "true");
52     request.setParameter("action", "newCard");
53     request.setParameter("cardType", "Visa");
54     request.setParameter("cardOwner", "Giovanni Giorgio");
55     request.setParameter("cardCVV", "533");
56     request.setParameter("cardNumber", "1111222233334449");
57     request.setParameter("cardDate", "2020-12-12");
58     request.setParameter("puppet", "true");
59     servlet.doPost(request, response);
60     String oracle = "Carta scaduta";
61     assertEquals(oracle, request.getAttribute("testMessage"));
62 }
63
64 @Test
65 public void TestAggiungiCartaDataNonValida () throws ServletException, IOException {
66     request.setParameter("cookie", "true");
67     request.setParameter("action", "newCard");
68     request.setParameter("cardType", "Visa");
69     request.setParameter("cardOwner", "Giovanni Giorgio");
70     request.setParameter("cardCVV", "533");
71     request.setParameter("cardNumber", "1111222233334449");
72     request.setParameter("cardDate", "2020-12-12");
73     request.setParameter("puppet", "true");
74     servlet.doPost(request, response);
75     String oracle = "Formato data non valido";
76     assertEquals(oracle, request.getAttribute("testMessage"));
77 }
78 }
```

PROBLEMI OUTPUT CONSOLE DI DEBUG TERMINALE

```
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_LoginFailed > Utente non esistente
```

Test Ricarica saldo



```
34 @Test
35 public void TestRicaricaSaldoOK () throws ServletException, IOException {
36     request.setParameter("cookie", "true");
37     request.setParameter("action", "updateBalance");
38     request.setParameter("amount", "10");
39     request.setParameter("cardId", "1");
40     request.setParameter("puppet", "true");
41     servlet.doGet(request, response);
42     String oracle = "Ricarica effettuata con successo";
43     assertEquals(oracle, request.getAttribute("testMessage"));
44 }
45
46 @Test
47 public void TestRicaricaQuantitaNegativa () throws ServletException, IOException {
48     request.setParameter("cookie", "true");
49     request.setParameter("action", "updateBalance");
50     request.setParameter("amount", "-10");
51     request.setParameter("cardId", "1");
52     request.setParameter("puppet", "true");
53     servlet.doGet(request, response);
54     String oracle = "Quantita' non valida";
55     assertEquals(oracle, request.getAttribute("testMessage"));
56 }
57
58 @Test
59 public void TestRicaricaQuantitaNonValida () throws ServletException, IOException {
60     request.setParameter("cookie", "true");
61     request.setParameter("action", "updateBalance");
62     request.setParameter("amount", "abc");
63     request.setParameter("cardId", "1");
64     request.setParameter("puppet", "true");
65     servlet.doGet(request, response);
66     String oracle = "Formato quantita' non valido";
67     assertEquals(oracle, request.getAttribute("testMessage"));
68 }
69
70 }
```

PROBLEMI OUTPUT CONSOLE DI DEBUG TERMINALE

```
# ShodanFilters > URI > /shodan_maven/
# ShodanFilters > Routing di [1] admin
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
Action: shop
# GameServlet > GET > Ultimi 5 giochi del negozio
# UserService > Query > SELECT user_id FROM users WHERE user_name = ?
# UserService > Utente inesistente
# LoginServlet > TC_LoginFailed > Utente non esistente
```

Testing Selenium

Selenium IDE - shodan-unit-testing*

Project: shodan-unit-testing*

Test suites +

Search tests...

http://localhost:8080/shodan_maven/

	Command	Target	Value
1	open	http://localhost:8080/shodan_maven/	
2	set window size	1898x1018	
3	click	id=login-username	
4	type	id=login-username	admin
5	click	id=login-password	
6	type	id=login-password	123

Command: open

Target: http://localhost:8080/shodan_maven/

Value:

Description:

Log Reference

6. sendKeys on id=login-password with value \${KEY_ENTER} OK	14:17:00
7. waitForElementPresent on id=settings-link with value 30000 OK	14:17:00
8. click on id=settings-link OK	14:17:02
9. waitForElementPresent on id=add-card with value 30000 OK	14:17:03
10. click on id=add-card OK	14:17:03
11. click on css=.credit-card-add OK	14:17:04