

Università degli studi di Salerno

Corso di Laurea in Informatica

Ingegneria del Software

FreshFood

Studenti:

Anna Fulgione

0512105572

Martina Casalnuovo

0512105110

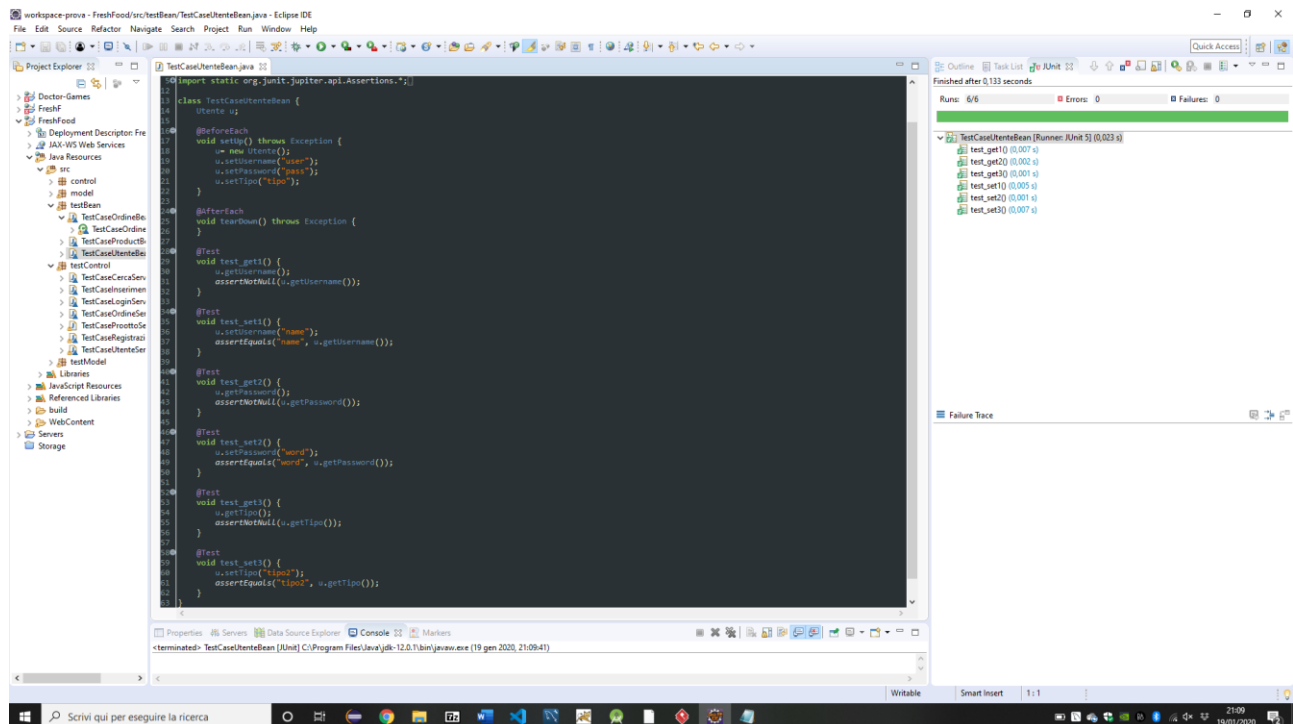
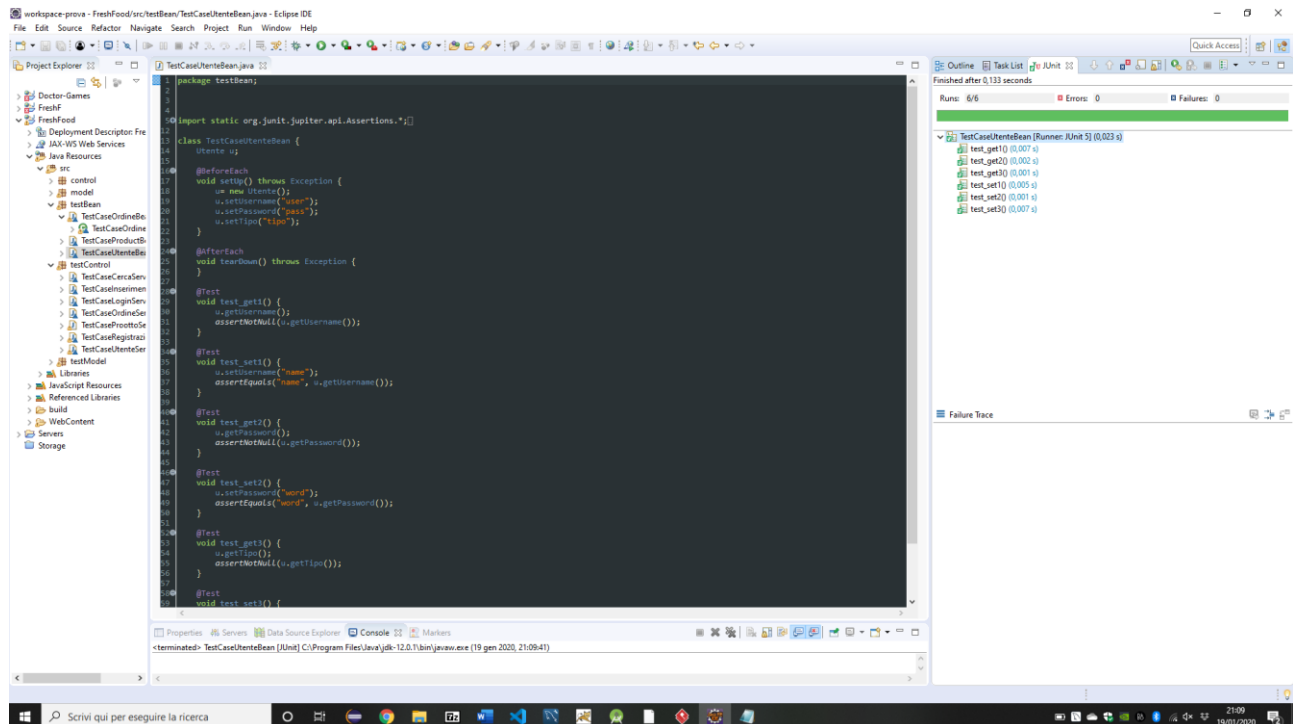
Anno Accademico: 2019/20

Sommario

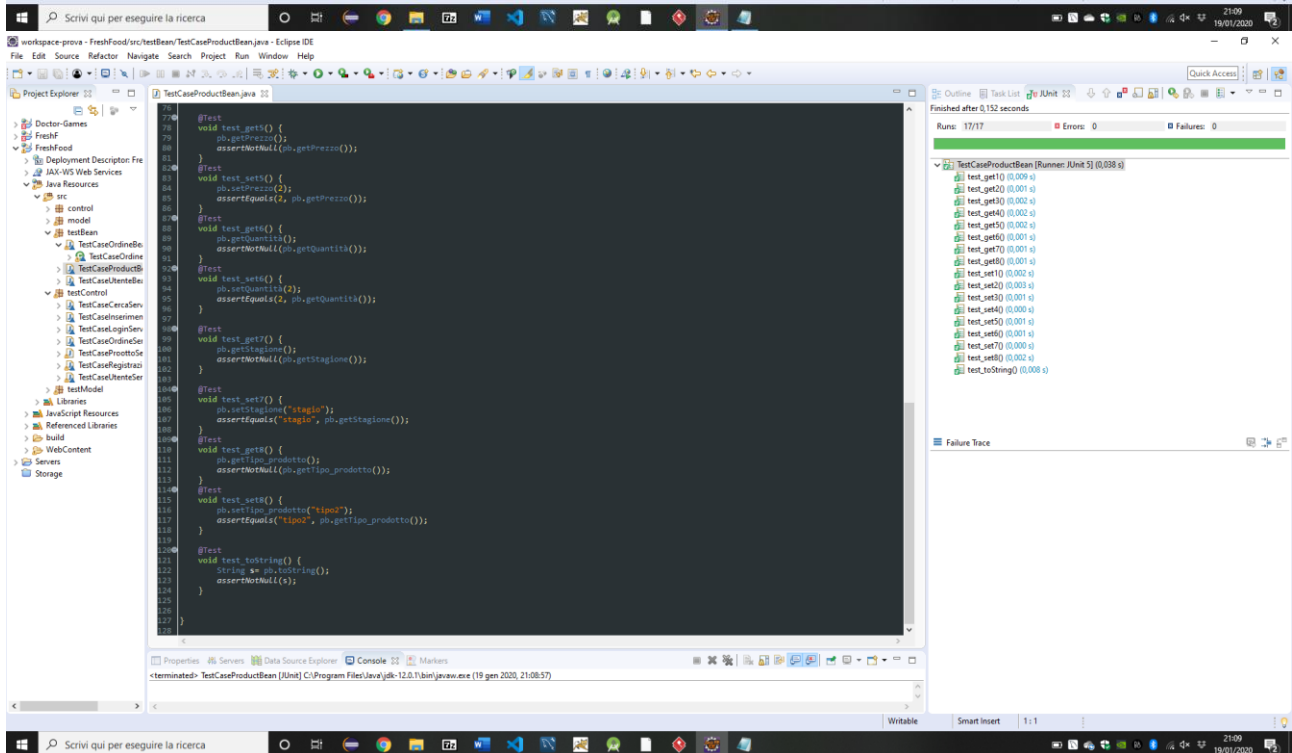
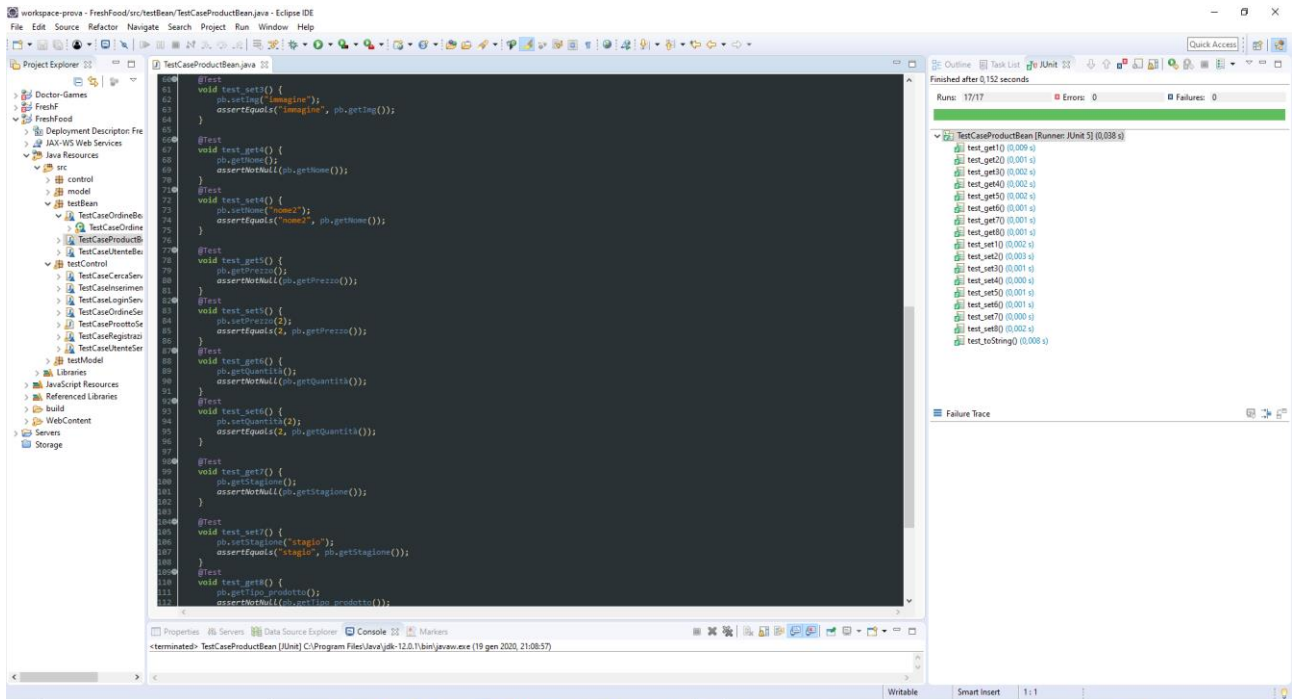
- 1. Test Bean3
 - 1.1 Test Utente3
 - 1.2 Prodotto4
 - 1.3 Ordine.....6
- 2. Test Model7
 - 2.1 Utente7
 - 2.2 Prodotto8
 - 2.3 Carrello10
 - 2.4 Ordine.....11
- 3. Test Control12
 - 3.1 Utente12
 - 3.2 Prodotto13
 - 3.3 Login14
 - 3.4 Cerca.....15
 - 3.5 Ordine.....15
 - 3.6 Inserimento16
 - 3.7 Registrazione16

1. Test Bean

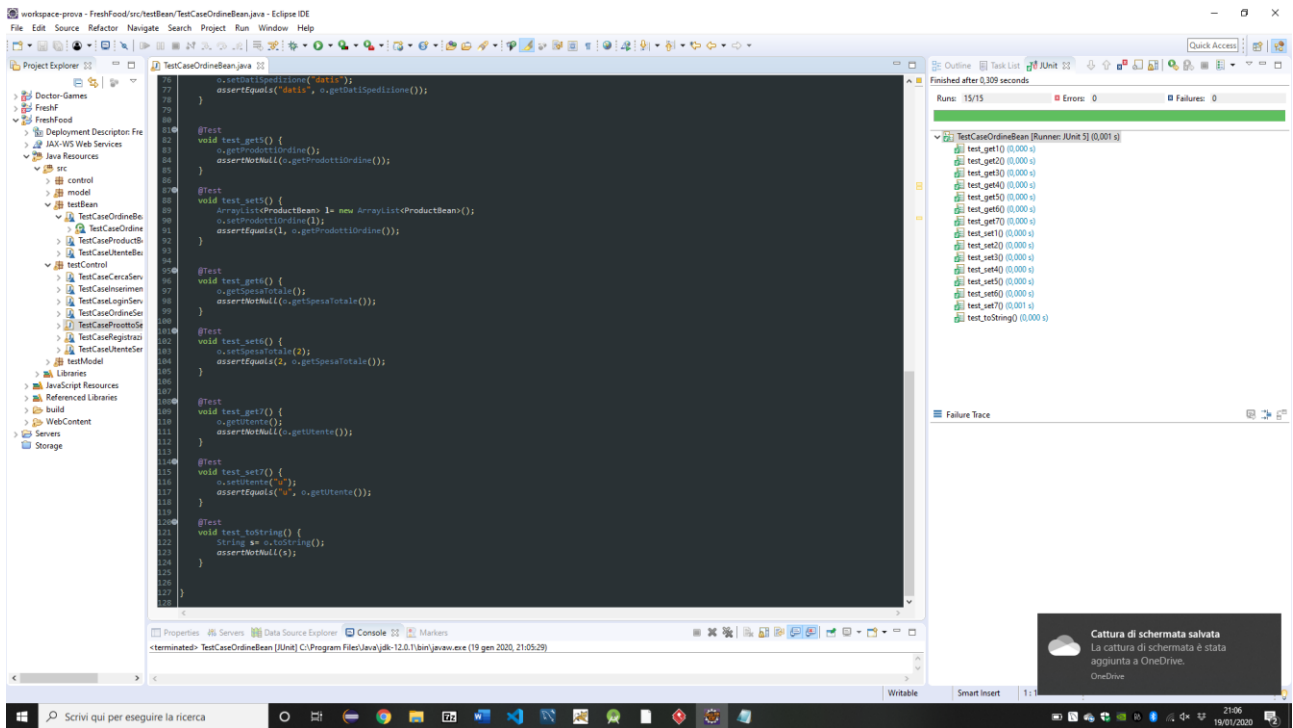
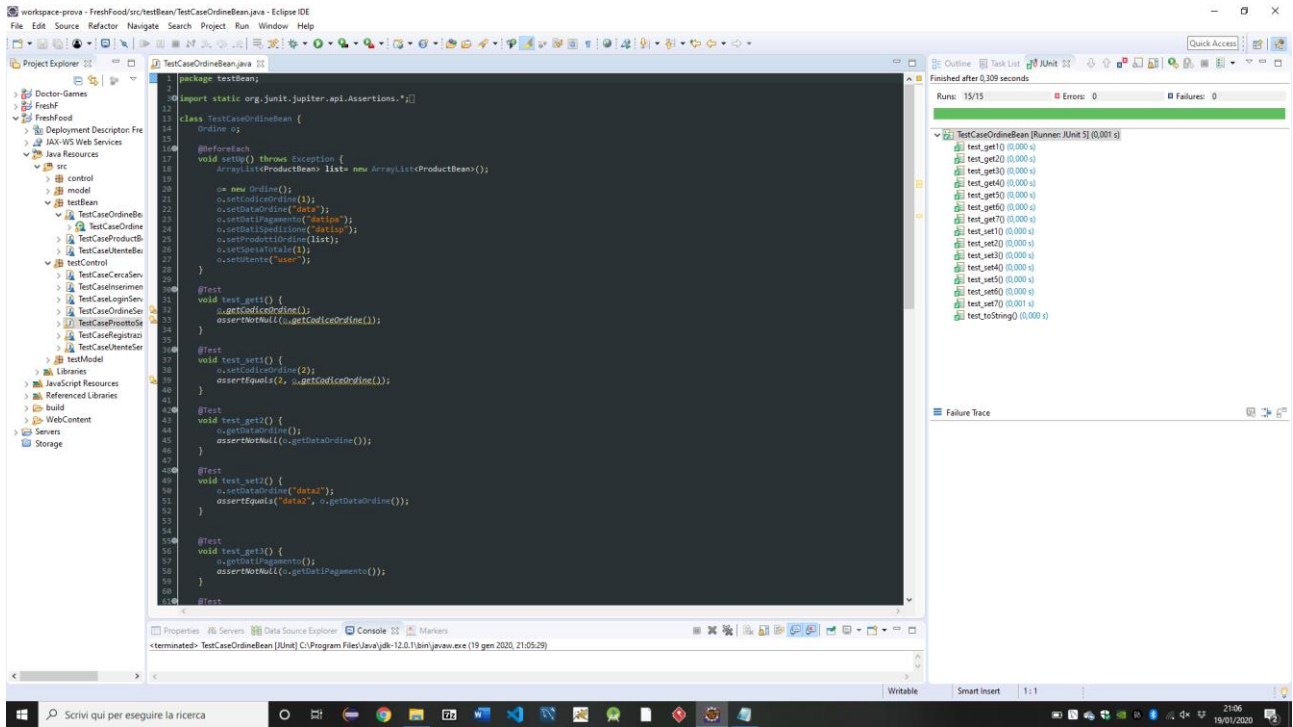
1.1 Test Utente



[illegible]



1.3 Ordine



2. Test Model

2.1 Utente

The image displays two screenshots of the Eclipse IDE, showing the development and testing of a Java class named `TestModelUtente.java`.

Top Screenshot: The code editor shows the initial implementation of `TestModelUtente`. It includes a `@Before` method for setup, an `@After` method for cleanup, and three `@Test` methods: `test1` (deleting an account), `test2` (deleting an account and verifying details), and `test3` (showing accounts). The JUnit runner on the right shows the test suite completed successfully after 0.579 seconds, with 5/5 runs, 0 errors, and 0 failures.

Bottom Screenshot: The code editor shows the same `TestModelUtente` class, but with additional test methods added: `test4` (login) and `test5` (checking username). The JUnit runner on the right shows the test suite completed successfully after 0.579 seconds, with 5/5 runs, 0 errors, and 0 failures.

The screenshot shows an IDE window titled "workspace - FreshFood - src\testModel\TestCaseProdotta.java". The main editor displays the source code for `TestCaseProdotta`, which includes methods for deleting products by ID or name and retrieving products by type or ordering.

```

54 void test2() throws SQLException {
55     boolean prod = prodotta.delete(prod_bean.getCode());
56     assertEquals(true, prod);
57 }
58
59 @Test
60 void test3() throws SQLException {
61     // public synchronized Collection<ProductBean> doRetrieveByProd(String tipo) throws SQLException
62     prodotta.doRetrieveByProd(prod_bean.getTipo_prodotta());
63     assertEquals("tipo prod", prod_bean.getTipo_prodotta());
64 }
65
66 @Test
67 void test4() throws SQLException {
68     //public synchronized ArrayList<ProductBean> doRetrieveAll() throws SQLException
69     prodotta.doRetrieveAll();
70     ArrayList<ProductBean> list= prodotta.doRetrieveAll();
71     assertNotNull(list);
72 }
73
74 @Test
75 void test5() throws SQLException {
76     // public synchronized ArrayList<ProductBean> cercaPerTipo(String tipo) throws SQLException
77     ArrayList<ProductBean> list= prodotta.cercaPerTipo(prod_bean.getTipo_prodotta());
78     list.add(prod_bean);
79     assertNotNull(list);
80 }
81
82
83 @Test
84 void test6() throws SQLException {
85     // public synchronized boolean delete(int code) throws SQLException
86     boolean b;
87     b= prodotta.delete(prod_bean.getCode());
88     assertEquals(prod_bean.getCode(), null);
89     assert(true(b));
90 }
91
92
93 @Test
94 void test7() throws SQLException {
95     // public synchronized Collection<ProductBean> doRetrieveAll(String order) throws SQLException
96     String ordine="";
97     Collection<ProductBean> list= prodotta.doRetrieveAll(ordine);
98     assertEquals("", list);
99 }
100
101 @Test
102 void test8() throws SQLException {
103     // public synchronized int getNumCode()
104     int n= prodotta.getNumCode();
    
```

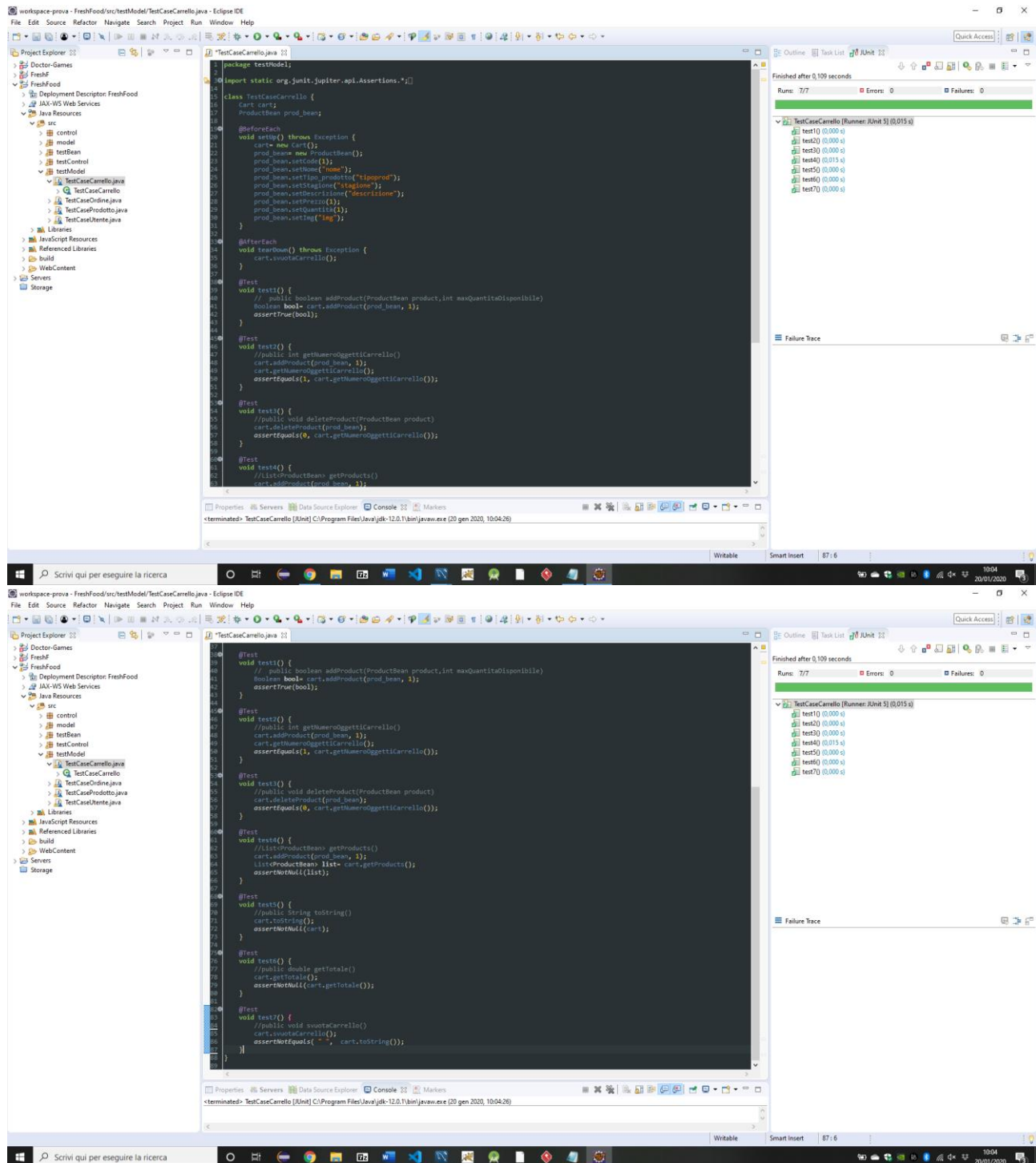
The right-hand pane shows the "Outline" view with a tree structure of test cases:

- `testCaseProdotta (Runner:JUnit5 [0.486 s])`
 - `test10 (0.333 s)`
 - `test20 (0.000 s)`
 - `test30 (0.031 s)`
 - `test40 (0.016 s)`
 - `test50 (0.022 s)`
 - `test60 (0.000 s)`
 - `test70 (0.073 s)`
 - `test80 (0.000 s)`
 - `test90 (0.047 s)`

Below the Outline view, the "Failure Trace" section is visible but currently empty.

The bottom status bar indicates the console output: "`<terminated> testCaseProdotta [JUnit] C:\Program Files\Java\jdk-12.0.1\bin\java.exe (20 gen 2020, 11:14:42)`".

2.3 Carrello



2.4 Ordine

The screenshot displays the Eclipse IDE interface with the following components:

- Project Explorer:** Shows the project structure with packages like `control`, `model`, and `src`. The `src` package contains `TestModel`, which includes `TestModelOrdine.java`.
- Editor:** Displays the `TestModelOrdine.java` file. The code defines a `TestCaseOrdine` class with the following methods:
 - `setup()`: Initializes the test environment, including creating a `Cart`, `ProdBean`, `Utente`, and `Ordine` objects.
 - `@Before`: A method that runs before each test, calling `setup()`.
 - `@Test`: Three test methods:
 - `test1()`: Tests the `placiatOrdine` method.
 - `test2()`: Tests the `caricaCarrello` method.
 - `test3()`: Tests the `aggiornaCarrello` method.
 - `@After`: A method that runs after each test, calling `tearDown()`.
 - `tearDown()`: Cleans up the test environment.
- JUnit Runner:** Shows the test results. The test suite `TestModelOrdine` ran successfully, with 4/4 tests passed in 0.358 seconds. The test results are listed as:
 - `test1()` (0.128 s)
 - `test2()` (0.016 s)
 - `test3()` (0.016 s)
 - `test4()` (0.016 s)
 - `test5()` (0.000 s)

3. Test Control

3.1 Utente

The screenshot shows the Eclipse IDE workspace. The Project Explorer on the left lists the project structure, including the 'testControl' package. The main editor displays the 'TestControl.java' file, which contains the following code:

```
package testControl;

import static org.junit.jupiter.api.Assertions.*;

class TestControlServlet extends Mockito {

    HttpServletRequest request;
    HttpServletResponse response;
    HttpSession session;
    Cart cart;
    ServletContext ctx;

    UtenteControl servlet = new UtenteControl();
    Utente utente;

    @BeforeEach
    void setUp() throws Exception {
        request = (HttpServletRequest) Mockito.mock(HttpServletRequest.class);
        response = (HttpServletResponse) Mockito.mock(HttpServletResponse.class);
        session = (HttpSession) Mockito.mock(HttpSession.class);
        ctx = (ServletContext) Mockito.mock(ServletContext.class);

        servlet = new UtenteControl();
        utente = new Utente();
        utente.setUsername("annafulgione@alice.it");
        utente.setPassword("anna");
        utente.setTipo("cliente");

        cart = new Cart();
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test() throws ServletException, IOException {
        servlet = new UtenteControl();
        String action = "deleteUser";
        String username = "annafulgione@alice.it";
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("XANBELLIO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        when(request.getParameter("username")).thenReturn(username);
        servlet.doGet(request, response);
        assertEquals("utente", null);
    }
}
```

The right-hand side of the IDE shows the Test Runner with a list of tests and their execution times:

Test	Time
test0	0.951 s
test10	0.028 s
test20	0.009 s

The screenshot shows the Eclipse IDE workspace after a screenshot capture. The Project Explorer on the left lists the project structure, including the 'testControl' package. The main editor displays the 'TestControl.java' file, which contains the following code:

```
package testControl;

import static org.junit.jupiter.api.Assertions.*;

class TestControlServlet extends Mockito {

    HttpServletRequest request;
    HttpServletResponse response;
    HttpSession session;
    Cart cart;
    ServletContext ctx;

    UtenteControl servlet = new UtenteControl();
    Utente utente;

    @BeforeEach
    void setUp() throws Exception {
        request = (HttpServletRequest) Mockito.mock(HttpServletRequest.class);
        response = (HttpServletResponse) Mockito.mock(HttpServletResponse.class);
        session = (HttpSession) Mockito.mock(HttpSession.class);
        ctx = (ServletContext) Mockito.mock(ServletContext.class);

        servlet = new UtenteControl();
        utente = new Utente();
        utente.setUsername("annafulgione@alice.it");
        utente.setPassword("anna");
        utente.setTipo("cliente");

        cart = new Cart();
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test() throws ServletException, IOException {
        servlet = new UtenteControl();
        String action = "deleteUser";
        String username = "annafulgione@alice.it";
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("XANBELLIO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        when(request.getParameter("username")).thenReturn(username);
        servlet.doGet(request, response);
        assertEquals("utente", null);
    }

    @Test
    void test1() throws ServletException, IOException {
        String action = "cancelUtente";
        String username = "annafulgione@alice.it";
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("XANBELLIO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        servlet.doGet(request, response);
        assertEquals("utente", null);
    }

    @Test
    void test2() throws ServletException, IOException {
        String action = "log-out";
        String username = "annafulgione@alice.it";
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("XANBELLIO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        servlet.doGet(request, response);
        assertEquals("utente", null);
    }
}
```

The right-hand side of the IDE shows the Test Runner with a list of tests and their execution times:

Test	Time
test0	0.951 s
test10	0.028 s
test20	0.009 s

A notification in the bottom right corner states: "Cattura di schermata salvata. La cattura di schermata è stata aggiunta a OneDrive." (Screenshot saved. The screenshot capture has been added to OneDrive.)

3.2 Prodotto

The image displays two screenshots of the Eclipse IDE, showing the development and testing of a Java application. The top screenshot shows the initial state of the project, with the `TestCercaServizioServlet` class selected. The bottom screenshot shows the `TestCercaServizioServlet` class with multiple test methods, including `test1()`, `test2()`, and `test3()`. The test results panel on the right indicates that the tests passed successfully.

Top Screenshot: Initial State

The `TestCercaServizioServlet` class is shown with the following code:

```
package testControl;

import static org.junit.jupiter.api.Assertions.*;

class TestCercaServizioServlet extends Mockito {
    HttpServletRequest request;
    HttpServletResponse response;
    HttpSession session;

    ProductControl servlet = new ProductControl();
    ProductBean prodotto = new ProductBean();
    Cart cart;
    Utente u;

    @BeforeEach
    void setUp() throws Exception {
        request = (HttpServletRequest) Mockito.mock(HttpServletRequest.class);
        response = (HttpServletResponse) Mockito.mock(HttpServletResponse.class);
        session = (HttpSession) Mockito.mock(HttpSession.class);

        cart = new Cart();
        prodotto.setCode(0);
        prodotto.setNome("mela");
        prodotto.setStagione("Inverno");
        prodotto.setTipo_prodotto("Frutta");
        prodotto.setQuantita(10);
        prodotto.setDescrizione("la mela è rossa");
        prodotto.setPrezzo(1.5);
        prodotto.setIscritto("");

        u = new Utente();
        u.setUsername("annafulgione@alice.it");
        servlet = new ProductControl();
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test1() throws ServletException, IOException {
        String action = "delete";
        String id = prodotto.getCode() + "";
        String username = getUsername();
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("CANELLO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        when(request.getParameter("id")).thenReturn(id);
        when(session.getAttribute("nome")).thenReturn(username);
        servlet.doGet(request, response);
        assertEquals(0, prodotto.getCode());
    }
}
```

The test results panel shows the following output:

```
Finished after 32,108 seconds
Runs: 2/2 Errors: 0 Failures: 0
TestCercaServizioServlet (Runner: JUnit 5) (31,949 s)
  test1() (0,898 s)
  test2() (31,051 s)
```

Bottom Screenshot: Test Results

The `TestCercaServizioServlet` class is shown with the following code:

```
package testControl;

import static org.junit.jupiter.api.Assertions.*;

class TestCercaServizioServlet extends Mockito {
    HttpServletRequest request;
    HttpServletResponse response;
    HttpSession session;

    ProductControl servlet = new ProductControl();
    ProductBean prodotto = new ProductBean();
    Cart cart;
    Utente u;

    @BeforeEach
    void setUp() throws Exception {
        request = (HttpServletRequest) Mockito.mock(HttpServletRequest.class);
        response = (HttpServletResponse) Mockito.mock(HttpServletResponse.class);
        session = (HttpSession) Mockito.mock(HttpSession.class);

        cart = new Cart();
        prodotto.setCode(0);
        prodotto.setNome("mela");
        prodotto.setStagione("Inverno");
        prodotto.setTipo_prodotto("Frutta");
        prodotto.setQuantita(10);
        prodotto.setDescrizione("la mela è rossa");
        prodotto.setPrezzo(1.5);
        prodotto.setIscritto("");

        u = new Utente();
        u.setUsername("annafulgione@alice.it");
        servlet = new ProductControl();
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test1() throws ServletException, IOException {
        String action = "delete";
        String id = prodotto.getCode() + "";
        String username = getUsername();
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("CANELLO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        when(request.getParameter("id")).thenReturn(id);
        when(session.getAttribute("nome")).thenReturn(username);
        servlet.doGet(request, response);
        assertEquals(0, prodotto.getCode());
    }

    @Test
    void test2() throws ServletException, IOException {
        String action = "add";
        String id = prodotto.getCode() + "";
        String username = getUsername();
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("CANELLO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        when(request.getParameter("id")).thenReturn(id);
        when(session.getAttribute("nome")).thenReturn(username);
        servlet.doGet(request, response);
        assertEquals(1, prodotto.getCode());
    }

    @Test
    void test3() throws ServletException, IOException {
        String action = "add";
        String id = prodotto.getCode() + "";
        String username = getUsername();
        when(request.getSession()).thenReturn(session);
        when(request.getAttribute("CANELLO")).thenReturn(cart);
        when(request.getParameter("action")).thenReturn(action);
        when(request.getParameter("id")).thenReturn(id);
        when(session.getAttribute("nome")).thenReturn(username);
        servlet.doGet(request, response);
        assertEquals(2, prodotto.getCode());
    }
}
```

The test results panel shows the following output:

```
Finished after 32,108 seconds
Runs: 2/2 Errors: 0 Failures: 0
TestCercaServizioServlet (Runner: JUnit 5) (31,949 s)
  test1() (0,898 s)
  test2() (31,051 s)
  test3() (31,051 s)
```

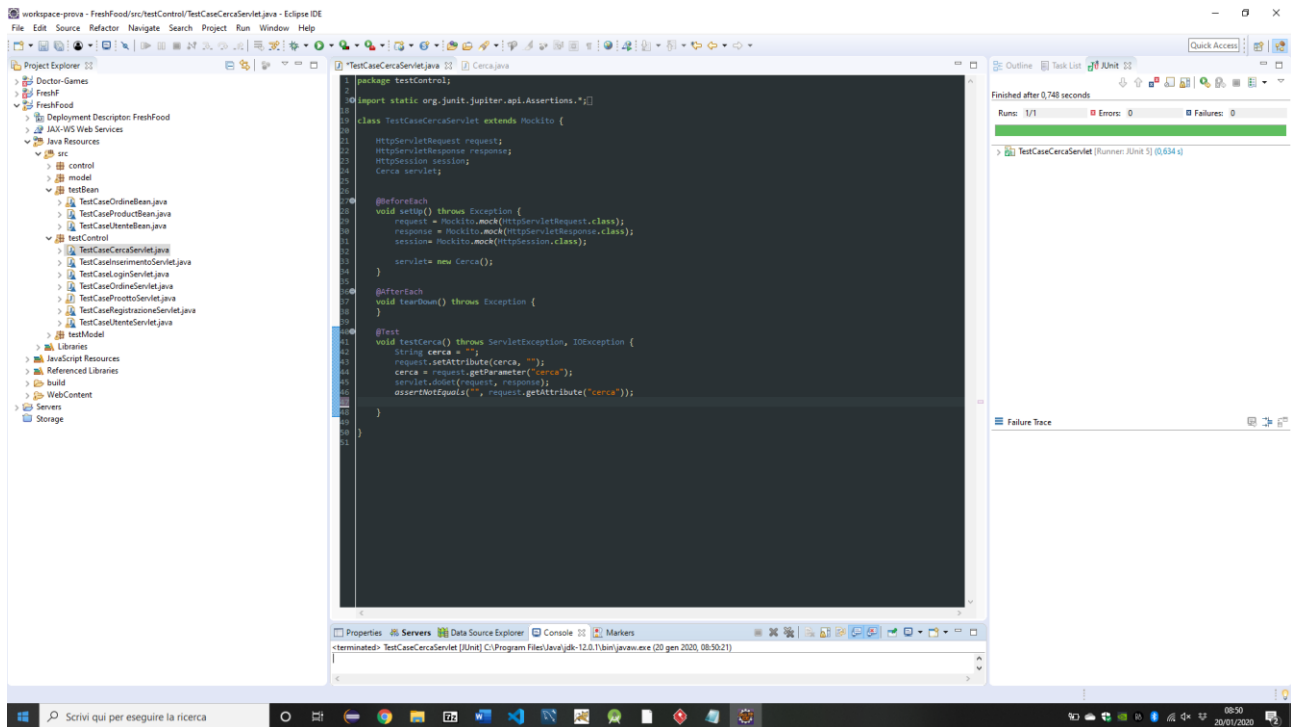
3.3 Login

The image displays two screenshots of the Eclipse IDE, showing the development and testing of a login service using Mockito and JUnit.

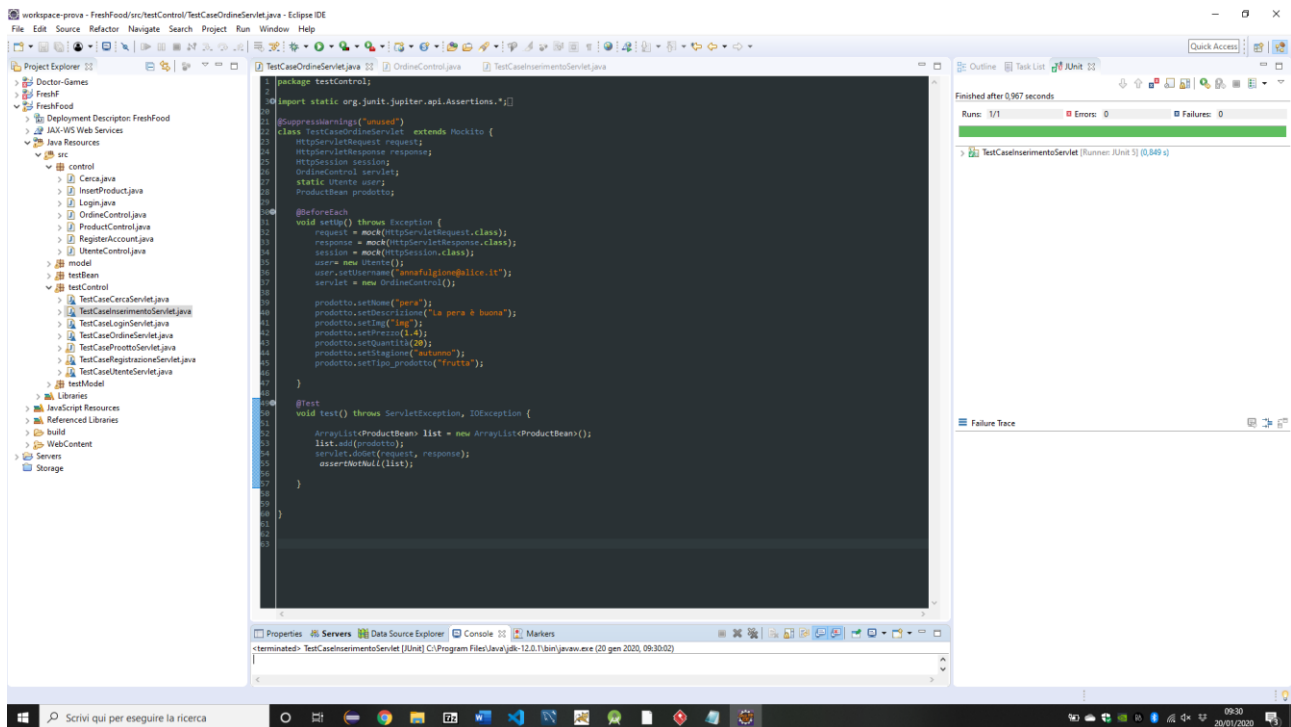
Top Screenshot: The main editor shows the `TestCasLoginServlet.java` file. The code includes imports for `org.junit.jupiter.api.Assertions`, `org.junit.jupiter.api.BeforeEach`, `org.junit.jupiter.api.Test`, `org.mockito.Mockito`, and `org.mockito.Mockito.mock`. The `TestCasLoginServlet` class extends `Mockito` and implements `HttpServletRequest`, `HttpServletResponse`, and `HttpSession`. The `@Test` method `testLogin()` is defined, which calls `loginServlet.login()` and asserts that the user is not null.

Bottom Screenshot: The same file is shown, but with additional code added. The `@Before` method `setUp()` is added, which sets up the mock objects for `HttpServletRequest`, `HttpServletResponse`, and `HttpSession`. The `@After` method `tearDown()` is also added, which calls `loginServlet.logout()` and asserts that the user is not null. The `@Test` method `testLogin()` is updated to include assertions for the session and the user.

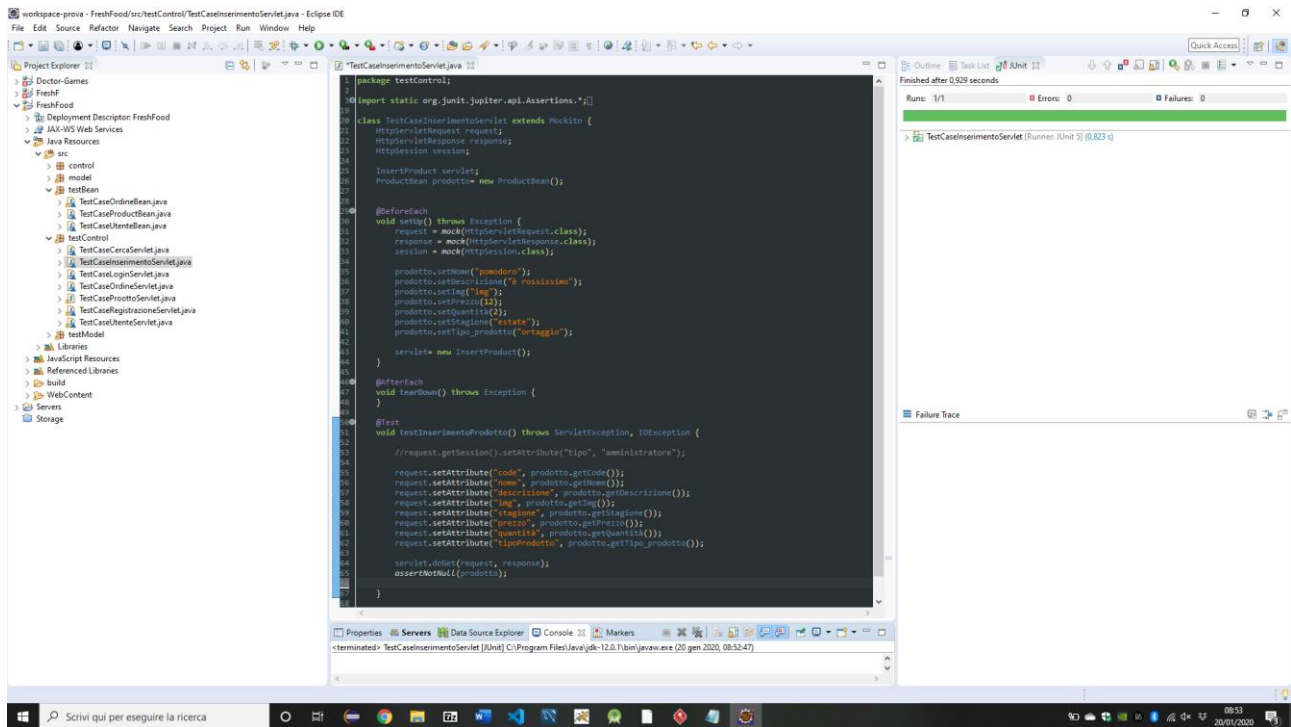
3.4 Cerca



3.5 Ordine



3.6 Inserimento



3.7 Registrazione

