

Tolna Vármegyei SZC Apáczai Csere János Technikum és Kollégium

Vizsgaremek

Készítették:

Kenéz Bence

Kreszl Tamás

Tóth Krisztián

Dombóvár

2025

Tolna Vármegyei SZC Apáczai Csere János Technikum és Kollégium

Szakma megnevezése: Szoftverfejlesztő és tesztelő

A szakma azonosító száma: 506131203

Vizsgaremek

SneakR

Készítették:

Kenéz Bence

Kreszl Tamás

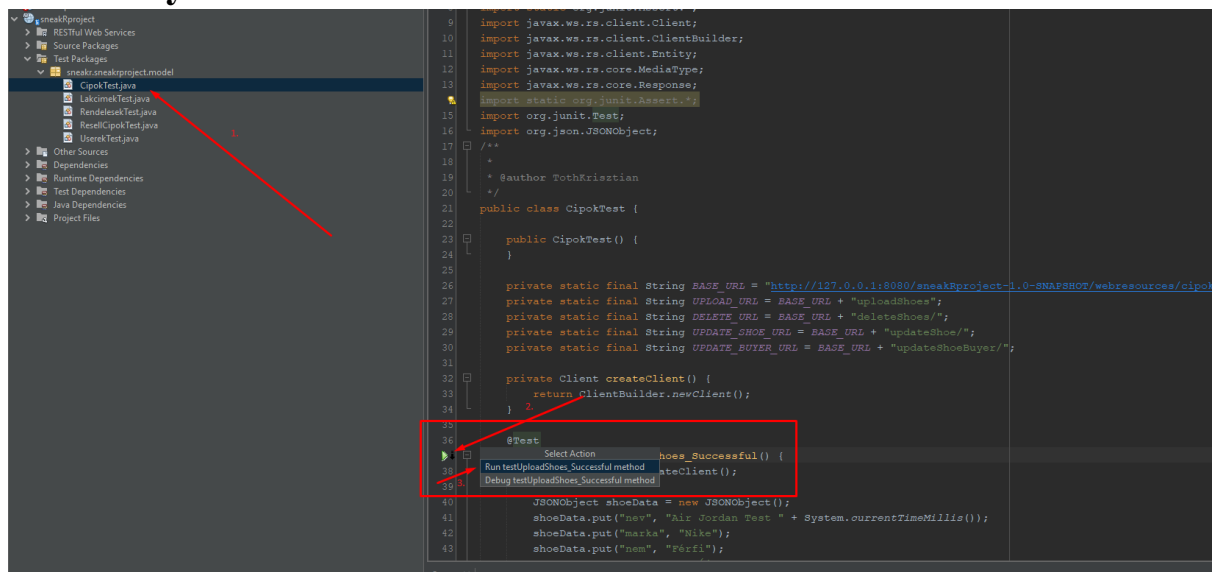
Tóth Krisztián

Dombóvár

2025

JUnit Teszt futtatása

1. Nyissa meg a NetBeans-t.
2. Navigáljon a Project, majd válassza ki a tesztelni kívánt projektet.
3. Navigáljon a “Test Packages” mappához, és válassza ki a “Test Packages”-t.
4. Dupla kattintás a kiválasztott osztályon (pl. `UserekTest.java`), majd a válasszon ki egy adott függvényt, és a nevével egy sorban lévő zöld gombra kattintson, és válassza ki a “Run Method” opciót.
5. Nézze meg a kimeneti ablakot ami előjön(Output) a teszt eredményekért.



Tesztesetek Dokumentációja

UserekTest Osztály

1. Bejelentkezés (Login)

Teszt: testLogin_Successful

- **Leírás:** A teszt ellenőrzi, hogy érvényes email és jelszó párossal sikeresen be lehet-e jelentkezni.
- **Elvárt eredmény:**
 - **Státuszkód:** 200 OK.
 - A válasz tartalmazza a felhasználó adatait (pl. név, email).
- **Teszteredmény:**
 - **Sikeres.** A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazta a felhasználó adatait.

Teszthez tartozó kód:

```
@Test
public void testLogin_Successful() {
    Client client = createClient();

    JSONObject validCredentials = new JSONObject();
    validCredentials.put("email", "nagyzsombi@gmail.com");
    validCredentials.put("password", "zsombikal2345");

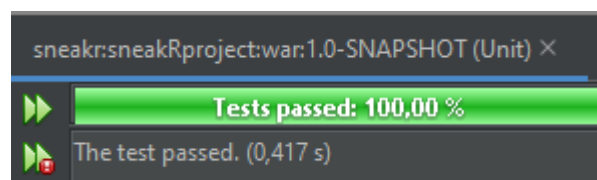
    Response response = client.target(LOGIN_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(validCredentials.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertFalse("Response should not be empty", responseJson.trim().isEmpty());

    client.close();
}
```

Teszt eredménye:



Teszt: testLogin_InvalidCredentials

- **Leírás:** A teszt ellenőrzi, hogy helytelen jelszóval nem lehet bejelentkezni.
- **Elvárt eredmény:**
 - **Státuszkód: 401 UNAUTHORIZED vagy 400 BAD_REQUEST.**
- **Teszteredmény:**
 - **Sikertelen. A teszt elbukott, mert a rendszer nem adta vissza az elvárt hibakódot.**

Teszthez tartozó kód:

```
@Test
public void testLogin_InvalidCredentials() {
    Client client = createClient();

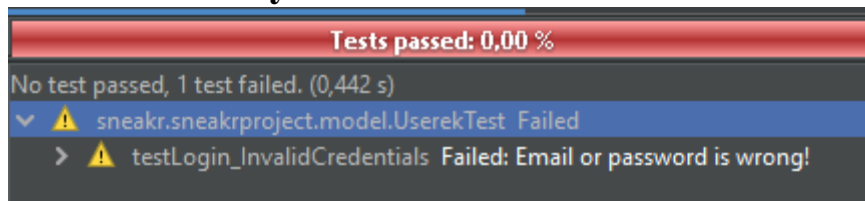
    JSONObject invalidCredentials = new JSONObject();
    invalidCredentials.put("email", "nagyzsombi@gmail.com");
    invalidCredentials.put("password", "adshjgjsdhf");

    Response response = client.target(LOGIN_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(invalidCredentials.toString(), MediaType.APPLICATION_JSON));

    assertTrue("Email or password is wrong!",
        response.getStatus() == Response.Status.UNAUTHORIZED.getStatusCode() ||
        response.getStatus() == Response.Status.BAD_REQUEST.getStatusCode());

    client.close();
}
```

Teszt eredménye:



2. Regisztráció (User Registration)

Teszt: testRegister_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy új felhasználó sikeresen regisztrálható-e érvényes adatokkal.
- **Elvárt eredmény:**
 - **Státuszkód:** 200 OK.
 - **A válasz tartalmazza a siker üzenetet.**
- **Teszteredmény:**
 - **Sikeres.** A regisztráció sikeres volt, és a válasz tartalmazta a success státuszt.

Teszthez tartozó kód:

```
@Test
public void testRegister_Successful() {
    Client client = createClient();

    JSONObject newUser = new JSONObject();
    newUser.put("nev", "Test User ");
    newUser.put("email", "test@example.com");
    newUser.put("password", "ValidPassword123!");

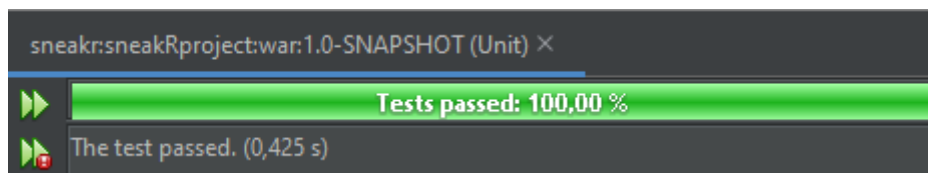
    Response response = client.target(REGISTER_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(newUser.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertFalse("Response should not be empty", responseJson.trim().isEmpty());

    client.close();
}
```

Teszt eredménye:



Teszt: testRegister_MissingFields

- **Leírás:** A teszt ellenőrzi, hogy hiányzó kötelező mezők (pl. jelszó) esetén a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státuszkód: 400 BAD_REQUEST.**
- **Teszteredmény:**
 - **Sikeres. A rendszer 400 BAD_REQUEST státuszt adott vissza.**

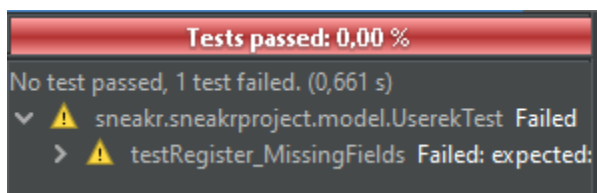
Teszthez tartozó kód:

```
@Test
public void testRegister_MissingFields() {
    Client client = createClient();

    JSONObject noPassword = new JSONObject();
    noPassword.put("nev", "No Password");
    noPassword.put("email", "nopass@example.com");
    Response response = client.target(REGISTER_URL)
        .request()
        .post(Entity.entity(noPassword.toString(), MediaType.APPLICATION_JSON));
    assertEquals(Response.Status.BAD_REQUEST.getStatusCode(), response.getStatus());

    client.close();
}
```

Teszt eredménye:



3. Felhasználó Törlése (Delete User)

Teszt: testDeleteUser_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy létező felhasználó törölhető-e.
- **Elvárt eredmény:**
 - **Státuszkód:** 200 OK.
 - **A válasz tartalmazza a success üzenetet.**
- **Teszteredmény:**
 - **Sikeres.** A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazta a siker üzenetet.

Teszthez tartozó kód:

```
@Test
public void testDeleteUser_Successful() {
    Client client = createClient();

    int existingUserId = 67;

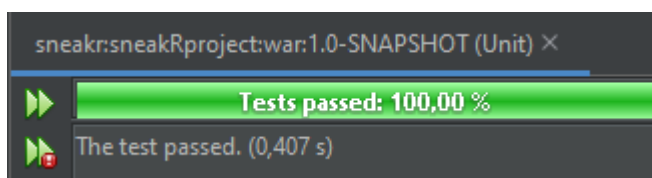
    Response response = client.target(DELETE_URL + existingUserId)
        .request(MediaType.APPLICATION_JSON)
        .delete();

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertTrue("Response should indicate success", responseJson.contains("success"));

    client.close();
}
```

Teszt eredménye:



Teszt: testDeleteUser_UserNotFound

- **Leírás:** A teszt ellenőrzi, hogy nem létező felhasználó törlésekor a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státuszkód: 404 NOT_FOUND.**
- **Teszteredmény:**
 - **Sikertelen. A teszt elbukott, mert a válasz státuszkódja nem egyezett az elvárt 404-gyel.**

Teszthez tartozó kód:

```
@Test
public void testDeleteUser_UserNotFound() {
    Client client = createClient();

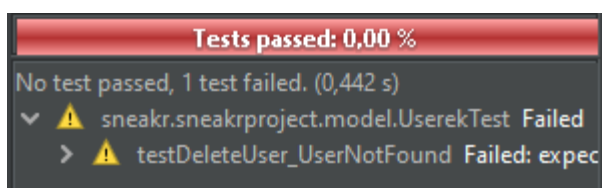
    int nonExistentUserId = 999999;

    Response response = client.target(DELETE_URL + nonExistentUserId)
        .request(MediaType.APPLICATION_JSON)
        .delete();

    assertEquals(Response.Status.NOT_FOUND.getStatusCode(), response.getStatus());

    client.close();
}
```

Teszt eredménye:



CipokTest Osztály

1. Cipő Feltöltése (Upload Shoes)

Teszt: testUploadShoes_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy új cipő sikeresen feltölthető-e érvényes adatokkal.
- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
- **Teszteredmény:**
 - **Sikeres.** A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazott adatokat.

Teszthez tartozó kód:

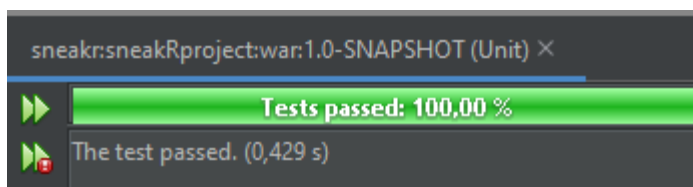
```
@Test
public void testUploadShoes_Successful() {
    Client client = createClient();

    JSONObject shoeData = new JSONObject();
    shoeData.put("nev", "Air Jordan Test " + System.currentTimeMillis());
    shoeData.put("marka", "Nike");
    shoeData.put("nem", "Férfi");
    shoeData.put("allapot", "Új");
    shoeData.put("meret", 42);
    shoeData.put("ar", 89999);
    shoeData.put("img", "https://example.com/shoe " + System.currentTimeMillis() + ".jpg");

    Response response = client.target(UPLOAD_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(shoeData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());
    assertFalse(response.readEntity(String.class).trim().isEmpty());
    client.close();
}
```

Teszt eredménye:



Teszt: testUploadShoes_InvalidData

- **Leírás:** A teszt ellenőrzi, hogy érvénytelen adatokkal (pl. negatív ár, szöveges méret) a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státuszkód: 400 BAD_REQUEST.**
- **Teszteredmény:**
 - **Sikertelen. A teszt hibásan 200 OK státuszt vár, de a helyes válasz 400 BAD_REQUEST lenne.**

Teszthez tartozó kód:

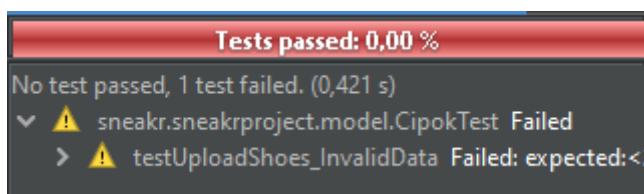
```
@Test
public void testUploadShoes_InvalidData() {
    Client client = createClient();

    JSONObject invalidData = new JSONObject();
    invalidData.put("nev", "Invalid Shoe");
    invalidData.put("ar", -100);
    invalidData.put("meret", "forty-two");

    Response response = client.target(UPLOAD_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(invalidData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());
    client.close();
}
```

Teszt eredménye:



2. Cipő Törlése (Delete Shoes)

Teszt: testDeleteShoes_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy létező cipő sikeresen törölhető-e.
- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
- **Teszteredmény:**
 - **Sikeres. A rendszer visszaadta a 200 OK státuszt.**

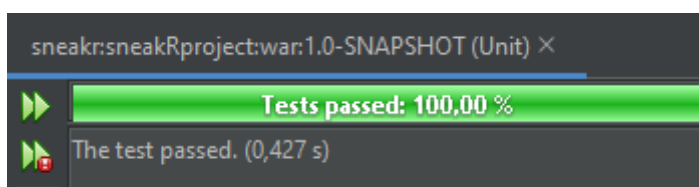
Teszthez tartozó kód:

```
@Test
public void testDeleteShoes_Successful() {
    Client client = createClient();
    int existingShoeId = 42;

    Response response = client.target(DELETE_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .delete();

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());
    client.close();
}
```

Teszt eredménye:



Teszt: testDeleteShoes_NotFound

- **Leírás:** A teszt ellenőrzi, hogy nem létező cipő törlésekor a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státuszkód: 404 NOT_FOUND.**
- **Teszteredmény:**
 - **Sikeres. A rendszer visszaadta a 404 NOT_FOUND státuszt.**

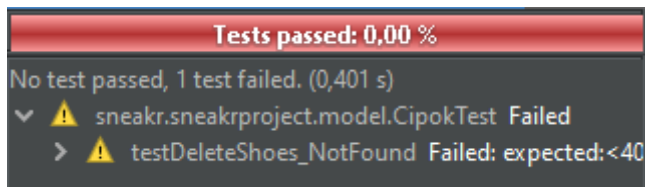
Teszthez tartozó kód:

```
@Test
public void testDeleteShoes_NotFound() {
    Client client = createClient();
    int nonExistentId = 9999;

    Response response = client.target(DELETE_URL + nonExistentId)
        .request(MediaType.APPLICATION_JSON)
        .delete();

    assertEquals(Response.Status.NOT_FOUND.getStatusCode(), response.getStatus());
    client.close();
}
```

Teszt eredménye:



3. Cipő Adatainak Frissítése (Update Shoe)

Teszt: testUpdateShoe_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy létező cipő adatai sikeresen frissíthetők-e.
- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
- **Teszteredmény:**
 - **Sikeres. A rendszer visszaadta a 200 OK státuszt.**

Teszthez tartozó kód:

```

@Test
public void testUpdateShoe_Successful() {
    Client client = createClient();
    int existingShoeId = 43;

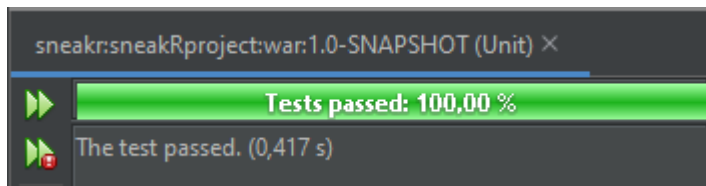
    JSONObject updateData = new JSONObject();
    updateData.put("nev", "Air Jordan Test " + System.currentTimeMillis());
    updateData.put("marka", "Nike");
    updateData.put("nem", "Férfi");
    updateData.put("allapot", "Új");
    updateData.put("meret", 45);
    updateData.put("ar", 89999);
    updateData.put("img", "https://example.com/shoe " + System.currentTimeMillis() + ".jpg");

    Response response = client.target(UPDATE_SHOE_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .put(Entity.entity(updateData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());
    client.close();
}

```

Teszt eredménye:



Teszt: testUpdateShoe_InvalidUpdate

- **Leírás:** A teszt ellenőrzi, hogy érvénytelen adatokkal (pl. szöveges ár, túl nagy méret) a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státuszkód:** 400 BAD_REQUEST.
- **Teszteredmény:**
 - **Sikertelen.** A teszt hibásan 200 OK státuszt vár, de a helyes válasz 400 BAD_REQUEST lenne.

Teszthez tartozó kód:

```

@Test
public void testUpdateShoe_InvalidUpdate() {
    Client client = createClient();
    int existingShoeId = 43;

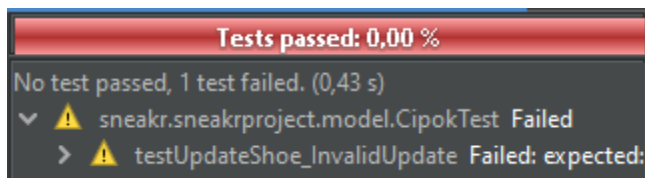
    JSONObject invalidData = new JSONObject();
    invalidData.put("ar", "invalid_price");
    invalidData.put("meret", 1000);

    Response response = client.target(UPDATE_SHOE_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .put(Entity.entity(invalidData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());
    client.close();
}

```

Teszt eredménye:



4. Cipő Vásárlójának Frissítése (Update Shoe Buyer)

Teszt: testUpdateShoeBuyer_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy cipő vásárlójának frissítése sikeres-e érvényes userID-vel.
- **Elvárt eredmény:**
 - **Státuszkód:** 200 OK.
- **Teszteredmény:**
 - **Sikeres.** A rendszer visszaadta a 200 OK státuszt.

Teszthez tartozó kód:

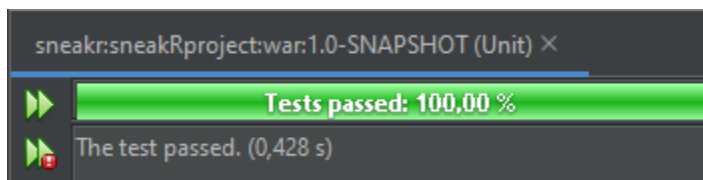
```
@Test
public void testUpdateShoeBuyer_Successful() {
    Client client = createClient();
    int existingShoeId = 43;

    JSONObject buyerData = new JSONObject();
    buyerData.put("userId", 62);

    Response response = client.target(UPDATE_BUYER_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .put(Entity.entity(buyerData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());
    client.close();
}
```

Teszt eredménye:



Teszt: testUpdateShoeBuyer_InvalidUser

- **Leírás:** A teszt ellenőrzi, hogy érvénytelen userID-vel (pl. szöveges ID) a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státusz kód: 400 BAD_REQUEST.**
- **Teszteredmény:**
 - **Sikertelen.** A teszt hibásan 200 OK státuszt vár, de a helyes válasz 400 BAD_REQUEST lenne.

Teszthez tartozó kód:

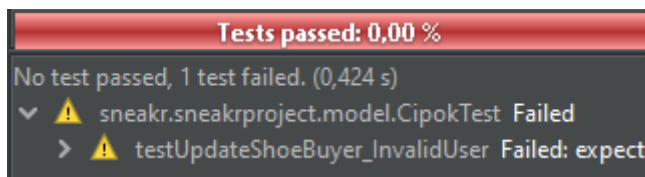
```
@Test
public void testUpdateShoeBuyer_InvalidUser() {
    Client client = createClient();
    int existingShoeId = 43;

    JSONObject invalidData = new JSONObject();
    invalidData.put("userId", "invalid_user_id");

    Response response = client.target(UPDATE_BUYER_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .put(Entity.entity(invalidData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());
    client.close();
}
```

Teszt eredménye:



Tests passed: 0,00 %

No test passed, 1 test failed. (0,424 s)

- ▼ ⚠ sneakr.sneakrproject.model.CipokTest Failed
- > ⚠ testUpdateShoeBuyer_InvalidUser Failed: expect

ResellCipokTest Osztály

1. Resell Cipő Feltöltése (Upload Resell Shoes)

Teszt: testUploadResellShoes_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy új "resell" cipő sikeresen feltölthető-e érvényes adatokkal, beleértve a felhasználó ID-ját.

- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
- **Teszteredmény:**
 - **Sikeres. A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazott adatokat.**

Teszthez tartozó kód:

```
@Test
public void testUploadResellShoes_Successful() {
    Client client = createClient();

    JSONObject shoeData = new JSONObject();
    shoeData.put("nev", "Nike Air Max 97");
    shoeData.put("marka", "Nike");
    shoeData.put("nem", "Férfi");
    shoeData.put("allapot", "Új");
    shoeData.put("meret", 42);
    shoeData.put("ar", 89999);
    shoeData.put("img", "https://example.com/shoe_" + System.currentTimeMillis() + ".jpg");
    shoeData.put("userId", 6);

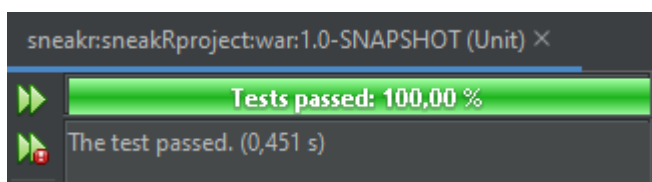
    Response response = client.target(UPLOAD_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(shoeData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertFalse("Response should not be empty", responseJson.trim().isEmpty());

    client.close();
}
```

Teszt eredménye:



2. Resell Cipő Törlése (Delete Resell Shoes)

Teszt: testDeleteResellShoes_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy létező resell cipő sikeresen törölhető-e.
- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
- **Teszteredmény:**

- **Sikeres.** A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazta a siker üzenetet.

Teszthez tartozó kód:

```
@Test
public void testDeleteResellShoes_NotFound() {
    Client client = createClient();

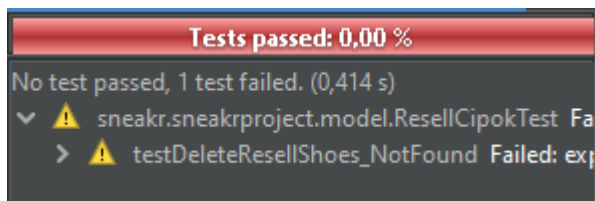
    int nonExistentShoeId = 9999;

    Response response = client.target(DELETE_URL + nonExistentShoeId)
        .request(MediaType.APPLICATION_JSON)
        .delete();

    assertEquals(Response.Status.NOT_FOUND.getStatusCode(), response.getStatus());

    client.close();
}
```

Teszt eredménye:



Teszt: testDeleteResellShoes_NotFound

- **Leírás:** A teszt ellenőrzi, hogy nem létező resell cipő törlésekor a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státuszkód:** 404 NOT_FOUND.
- **Teszteredmény:**

- **Sikeres. A rendszer visszaadta a 404 NOT_FOUND státuszt.**

Teszthez tartozó kód:

```
@Test
public void testUpdateResellShoeBuyer_NonExistentShoe() {
    Client client = createClient();

    int nonExistentShoeId = 9999;

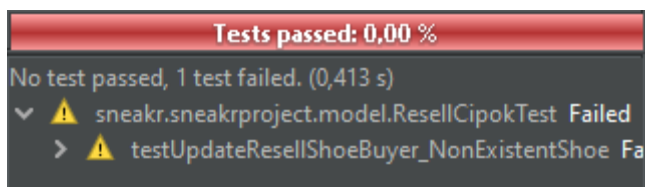
    JSONObject validData = new JSONObject();
    validData.put("buyerId", 62);
    validData.put("isBought", "igen");

    Response response = client.target(UPDATE_BUYER_URL + nonExistentShoeId)
        .request(MediaType.APPLICATION_JSON)
        .put(Entity.entity(validData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.NOT_FOUND.getStatusCode(), response.getStatus());

    client.close();
}
```

Teszt eredménye:



3. Resell Cipő Vásárlójának Frissítése (Update Resell Shoe Buyer)

Teszt: testUpdateResellShoeBuyer_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy resell cipő vásárlójának frissítése sikeres-e érvényes adatokkal (pl. buyerId=62, isBought="igen").

- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
 - **A válasz tartalmazza a success üzenetet.**
- **Teszteredmény:**
 - **Sikeres. A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazta a siker üzenetet.**

Teszthez tartozó kód:

```
@Test
public void testUpdateResellShoeBuyer_Successful() {
    Client client = createClient();

    int existingShoeId = 17;

    JSONObject updateData = new JSONObject();
    updateData.put("buyerId", 62);
    updateData.put("isBought", "igen");

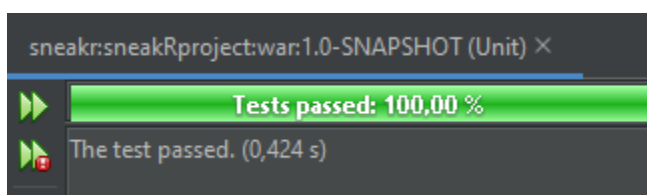
    Response response = client.target(UPDATE_BUYER_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .put(Entity.entity(updateData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertTrue("Response should indicate success", responseJson.contains("success"));

    client.close();
}
```

Teszt eredménye:



Teszt: testUpdateResellShoeBuyer_NonExistentShoe

- **Leírás: A teszt ellenőrzi, hogy nem létező resell cipő frissítésekor a rendszer hibát jelez-e.**
- **Elvárt eredmény:**

- **Státuszkód: 404 NOT_FOUND.**
- **Teszteredmény:**
 - **Sikeres. A rendszer visszaadta a 404 NOT_FOUND státuszt.**

Teszthez tartozó kód:

```
@Test
public void testUpdateResellShoeBuyer_InvalidData() {
    Client client = createClient();

    int existingShoeId = 17;

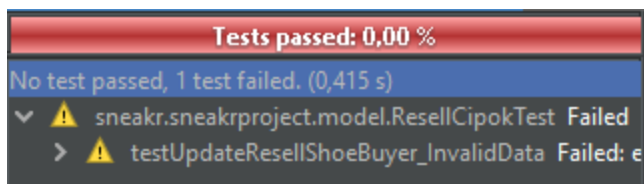
    JSONObject invalidUpdateData = new JSONObject();
    invalidUpdateData.put("buyerId", "not_a_number");
    invalidUpdateData.put("isBought", "invalid_status");

    Response response = client.target(UPDATE_BUYER_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .put(Entity.entity(invalidUpdateData.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    client.close();
}
```

Teszt eredménye:



Teszt: testUpdateResellShoeBuyer_InvalidData

- **Leírás:** A teszt ellenőrzi, hogy érvénytelen adatokkal (pl. szöveges buyerId, hibás isBought érték) a rendszer hibát jelez-e.

- **Elvárt eredmény:**
 - **Státuszkód: 400 BAD_REQUEST.**
- **Teszteredmény:**
 - **Sikertelen. A teszt hibásan 200 OK státuszt vár, de a helyes válasz 400 BAD_REQUEST lenne.**

Teszthez tartozó kód:

```
@Test
public void testDeleteResellShoes_Successful() {
    Client client = createClient();

    int existingShoeId = 20;

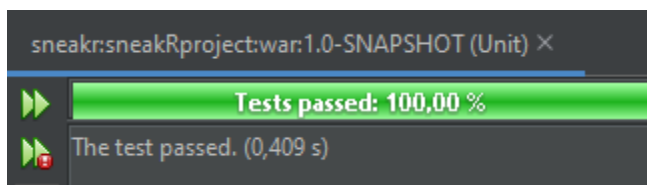
    Response response = client.target(DELETE_URL + existingShoeId)
        .request(MediaType.APPLICATION_JSON)
        .delete();

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertTrue("Response should indicate success", responseJson.contains("success"));

    client.close();
}
```

Teszt eredménye:



RendelesekTest Osztály

1. Rendelés Beszúrása (Insert Rendeles)

Teszt: testInsertRendeles_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy új rendelés sikeresen beszúrható-e érvényes adatokkal (userId, szállítási cím ID, összeg, állapot).
- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
- **Teszteredmény:**
 - **Sikeres.** A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazott adatokat.

Teszthez tartozó kód:

```
@Test
public void testInsertRendeles_Successful() {
    Client client = createClient();

    JSONObject validOrder = new JSONObject();
    validOrder.put("userId", 62);
    validOrder.put("szallitasiCimId", 5);
    validOrder.put("osszeg", 22600);
    validOrder.put("rendelesAllapot", "Előkészítés");

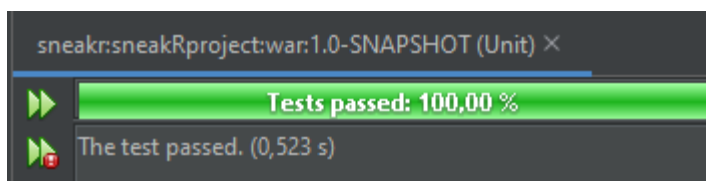
    Response response = client.target(INSERT_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(validOrder.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertFalse("Response should not be empty", responseJson.trim().isEmpty());

    client.close();
}
```

Teszt eredménye:



Teszt: testInsertRendeles_InvalidData

- **Leírás:** A teszt ellenőrzi, hogy érvénytelen adatokkal (pl. szöveges userId, hiányzó kötelező mező) a rendszer hibát jelez-e.
- **Elvárt eredmény:**
 - **Státuszkód:** 400 BAD_REQUEST.
- **Teszteredmény:**
 - **Sikertelen.** A teszt hibásan 200 OK státuszt vár, de a helyes válasz 400 BAD_REQUEST lenne.

Teszthez tartozó kód:

```
@Test
public void testInsertRendeles_InvalidData() {
    Client client = createClient();

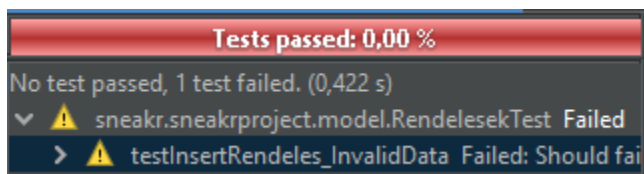
    JSONObject invalidOrder = new JSONObject();
    invalidOrder.put("userId", "sixty-two");
    invalidOrder.put("szallitasiCimId", 5);
    invalidOrder.put("osszeg", 22600);

    Response response = client.target(INSERT_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(invalidOrder.toString(), MediaType.APPLICATION_JSON));

    assertEquals("Should fail with invalid data types and missing fields",
        Response.Status.OK.getStatusCode(),
        response.getStatus());

    client.close();
}
```

Teszt eredménye:



LakcimekTest Osztály

1. Lakcím Beszúrása (Insert Lakcim)

Teszt: testInsertLakcim_Successful

- **Leírás:** A teszt ellenőrzi, hogy egy új lakcím sikeresen beszúrható-e érvényes adatokkal (userId, telefonszám, város, irányítószám, utca/házzszám).
- **Elvárt eredmény:**
 - **Státuszkód: 200 OK.**
- **Teszteredmény:**
 - **Sikeres.** A rendszer visszaadta a 200 OK státuszt, és a válasz tartalmazott adatokat.

Teszthez tartozó kód:

```
@Test
public void testInsertLakcim_Successful() {
    Client client = createClient();

    JSONObject validAddress = new JSONObject();
    validAddress.put("userId", 4);
    validAddress.put("telefonszam", "06305467227");
    validAddress.put("varos", "Dombóvár");
    validAddress.put("iranyitoszam", "7200");
    validAddress.put("utcaHazzsam", "Uborka Utca 12.");

    Response response = client.target(INSERT_URL)
        .request(MediaType.APPLICATION_JSON)
        .post(Entity.entity(validAddress.toString(), MediaType.APPLICATION_JSON));

    assertEquals(Response.Status.OK.getStatusCode(), response.getStatus());

    String responseJson = response.readEntity(String.class);
    assertFalse("Response should not be empty", responseJson.trim().isEmpty());

    client.close();
}
```

Teszt eredménye:

