

LABORATORIO #7 SOFTWARE TESTING

Valentina Bueno Valles
Valentina Viafara Esteban

- I. Descripción y soporte visual de las pruebas de unidad diseñadas, implementadas y ejecutadas.

Consideración: Para el desarrollo de este laboratorio se hizo uso de la aplicación que el docente dejó lista en el Laboratorio 4 debido a que no se logró completar con éxito todo el desarrollo de la aplicación como se había planteado en el Laboratorio 2. Adicionalmente, solo se hicieron pruebas sobre uno de los métodos presentes que es *sendMoney()* ya que *createUser()* se ejecuta directamente en la base de datos y al ser una prueba local no se tiene acceso a ella.

Para realizar la prueba del método *sendMoney()* se hizo necesario crear en la clase *UserController* un método auxiliar llamado *sendMoneyMod()* que no tuviera conexión con las otras capas y así no genere errores en la compilación y ejecución de las pruebas. Simplemente se busca probar que el método realice la actualización de los saldos (resta el valor a enviar de la cuenta origen y suma el valor enviado a la cuenta destino) en caso de poder llevarse a cabo la transacción o no actualizar los saldos en caso de no ser posible.

```
//MÉTODO SENDMONEY MODIFICADO PARA PODER PROBAR LA FUNCIONALIDAD
public static boolean sendMoneyMod(User sourceUser, User targetUser, double value) {

    //userRepository = new UserRepository(context);

    //final User sourceUser = userRepository.getUserById(sourceId);
    System.out.println("Source User - ID: " + sourceUser.getId() +
        ", Name: " + sourceUser.getName() +
        ", Balance: " + sourceUser.getBalance());

    if (sourceUser.getBalance() >= value) {

        //final User targetUser = userRepository.getUserById(targetId);
        System.out.println("Target User - ID: " + targetUser.getId() +
            ", Name: " + targetUser.getName() +
            ", Balance: " + targetUser.getBalance());

        sourceUser.setBalance(sourceUser.getBalance() - value);
        targetUser.setBalance(targetUser.getBalance() + value);
        //userRepository.updateUser(sourceUser);
        //userRepository.updateUser(targetUser);

        //final User updatedSourceUser = userRepository.getUserById(sourceId);
        System.out.println("Source User (updated) - ID: " + sourceUser.getId() +
            ", Name: " + sourceUser.getName() +
            ", Balance: " + sourceUser.getBalance());

        //final User updatedTargetUser = userRepository.getUserById(targetId);
        System.out.println("Target User (updated) - ID: " + targetUser.getId() +
            ", Name: " + targetUser.getName() +
            ", Balance: " + targetUser.getBalance());

        return true;
    } else {
        return false;
    }
}
```

Por otra parte se crea una clase en la cual se llevarán a cabo las pruebas del método. Para hacer estas pruebas se sigue el siguiente principio: Se crean 2 usuarios (origen y destino) con unos datos predeterminados por las desarrolladoras, se sabe cuál va a ser el resultado por lo que se crea una variable booleana cargada con el dato que se sabe que es verdadero según el caso y se llama al método *assertThat()* para comparar el dato cargado y el valor que *sendMoneyMod()* enviará. En caso de que ambos datos sean correctos la ejecución termina satisfactoriamente.

Se crearon 2 casos de prueba:

1. Se crea un usuario origen que tiene saldo de \$40.000 y un usuario destino que tiene un saldo de \$10.000 y se intenta enviar \$10.000, como la transacción si es posible, se carga la variable booleana con True para hacer la comparación.

```
UserController.java x UserControllerTest.java x
package co.edu.unal.se1.businessLogic.controller;

import org.junit.Test;

import co.edu.unal.se1.dataAccess.model.User;

import static org.junit.Assert.*;

public class UserControllerTest {

    @Test
    public void sendMoneySiPasa() {

        User sourceUser = new User();
        sourceUser.id = 1234;
        sourceUser.name = "Valentina V";
        sourceUser.balance = 40000;
        //assertEquals(user1.id,1234);

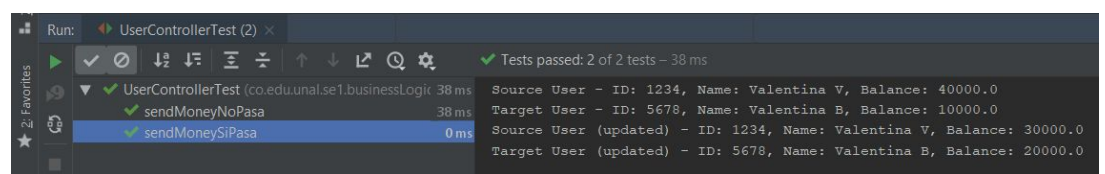
        User targetUser = new User();
        targetUser.id = 5678;
        targetUser.name = "Valentina B";
        targetUser.balance = 10000;
        //assertEquals(user2.id,5678);

        double value = 10000;

        //YO CONOZCO QUE EL VALOR DEBE SER VERDADERO
        boolean result=true;

        //COMPARACION DE LOS METODOS
        assertEquals(result, UserController.sendMoneyMod(sourceUser,targetUser,value));
    }
}
```

Como ambas variables coinciden, se termina la ejecución correctamente como se muestra a continuación:



Run: UserControllerTest (2) x

Tests passed: 2 of 2 tests - 38 ms

Test	Duration
UserControllerTest (co.edu.unal.se1.businessLogic)	38 ms
sendMoneyNoPasa	38 ms
sendMoneySiPasa	0 ms

Source User - ID: 1234, Name: Valentina V, Balance: 40000.0
Target User - ID: 5678, Name: Valentina B, Balance: 10000.0
Source User (updated) - ID: 1234, Name: Valentina V, Balance: 30000.0
Target User (updated) - ID: 5678, Name: Valentina B, Balance: 20000.0

2. Se crea un usuario origen que tiene saldo de \$4.000 y un usuario destino que tiene un saldo de \$10.000 y se intenta enviar \$10.000, como la transacción no es posible, se carga la variable booleana con False para hacer la comparación.

```
@Test
public void sendMoneyNoPasa() {

    User sourceUser = new User();
    sourceUser.id = 1234;
    sourceUser.name = "Valentina V";
    sourceUser.balance = 4000;
    //assertEquals(user1.id,1234);

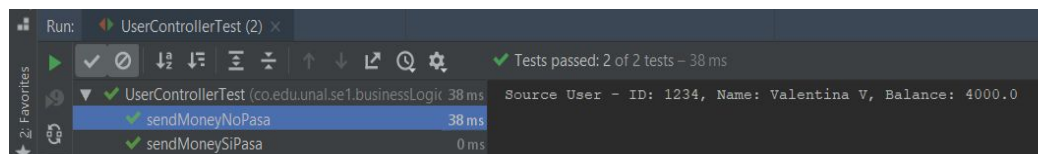
    User targetUser = new User();
    targetUser.id = 5678;
    targetUser.name = "Valentina B";
    targetUser.balance = 10000;
    //assertEquals(user2.id,5678);

    double value = 10000;

    //YO CONOZCO QUE EL VALOR DEBE SER FALSO
    boolean result=false;

    //COMPARACION DE LOS METODOS
    assertEquals(result, UserController.sendMoneyMod(sourceUser,targetUser,value));
}
```

Como ambas variables coinciden, se termina la ejecución correctamente como se muestra a continuación:



II. *Enlace al repositorio público creado y en el cual se desarrolló el laboratorio.*

<https://github.com/un-sei2019ii-labs-valealcuadrado/bank.git>