

UNIVERSIDAD DE LAS FUERZAS ARMADAS



Object-oriented programming

NRC: 9642

U2 Workshop N° 34

Topic: Project Revision

ING. Jorge Edison Lascano.

Rubric:

1. GitHub	/100
2. Project	/100
1. Requirements	10/10 (features)
2. Use Case	10/10 (features and users are modeled)
3. Class Diagram	/10 (MVC)
4. Mockup	/10 (tool/hand-made) vs CD
5. MongoDB	/10 (vs CD)
6. Test Cases	/10 (JUnit, at least 40)
7. OOP	/10 (Fundamentals: Polymorphism, Inheritance, Abstraction, Interfaces, Modularity, ...)
8. Clean GUI	/10 (No business rules or CRUD or validation code in buttons)
9. Clean Code	/20 (Clean Code)
3. Execution	/100
1. 4 business rules	40
2. Login	10
3. others	50

Evidences

1. GitHub	7:53 - 7:58
2. Project	
1. Requirements	7:58 - 8:03
2. Use Case	8:03 - 8:10
3. Class Diagram	8:10 - 8:15
4. Mockup	8:15 - 8:20
5. MongoDB	8:23 - 8:28
6. Test Cases	8:28 - 8:33
7. OOP	8:33 - 8:38
8. Clean GUI	8:38 - 8:43
9. Clean Code	8:43 - 8:48
3. Execution	
1. 4 business rules	
2. Login	
3. others	

T2

Team Name and Project: KillChain - EVSU Store

Inspector Team (Leader): Yeshua Chilikuinga

GitHub URL: <https://github.com/JuDav-093/ESPE23-KillChainTeam>

MongoDB Compass URL:

<mongodb+srv://jcobena:jcobena@cluster0.mhpieao.mongodb.net/>

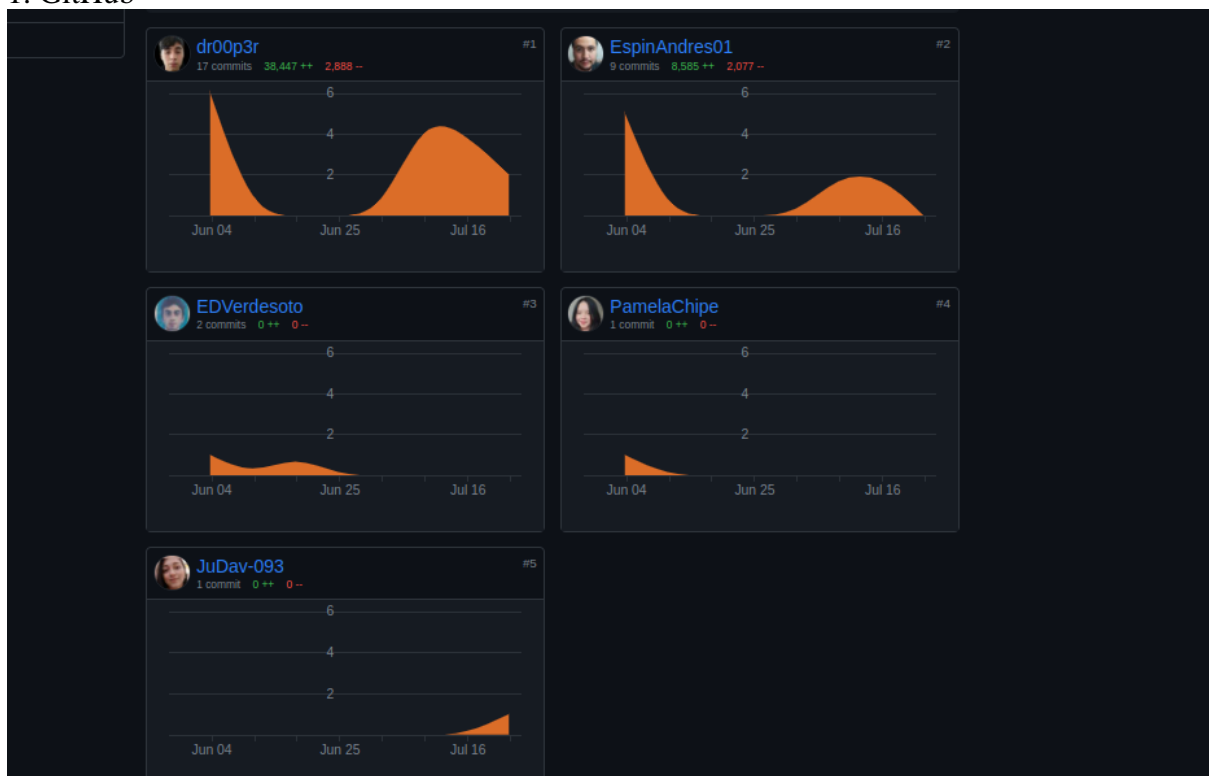
2. COBEÑA ZAMBRANO JOAN OSWALDO	80
3. DAVILA MOLINA ANABEL	25
4. ESPIN ANDRADE ANDRES ALEXANDER	65

Rubric:

1. GitHub	/100	
2. Project	74/100	
10. Requirements	10/10 (features)	
11. Use Case	0/10 (features and users are modeled)	
12. Class Diagram	8/10 (MVC)	
13. Mockup	10/10 (tool/hand-made) vs CD	
14. MongoDB	0/10 (vs CD)	
15. Test Cases	8/10 (JUnit, at least 40)	
16. OOP	10/10 (Fundamentals: Polymorphism, Inheritance, Abstraction, Interfaces, Modularity, ...)	
17. Clean GUI	10/10 (No business rules or CRUD or validation code in buttons)	
18. Clean Code	18/20 (Clean Code)	
3. Execution	100/100	
4. 4 business rules	40/40	
5. Login	10/10	
6. others	50/50	

Evidences

1. GitHub



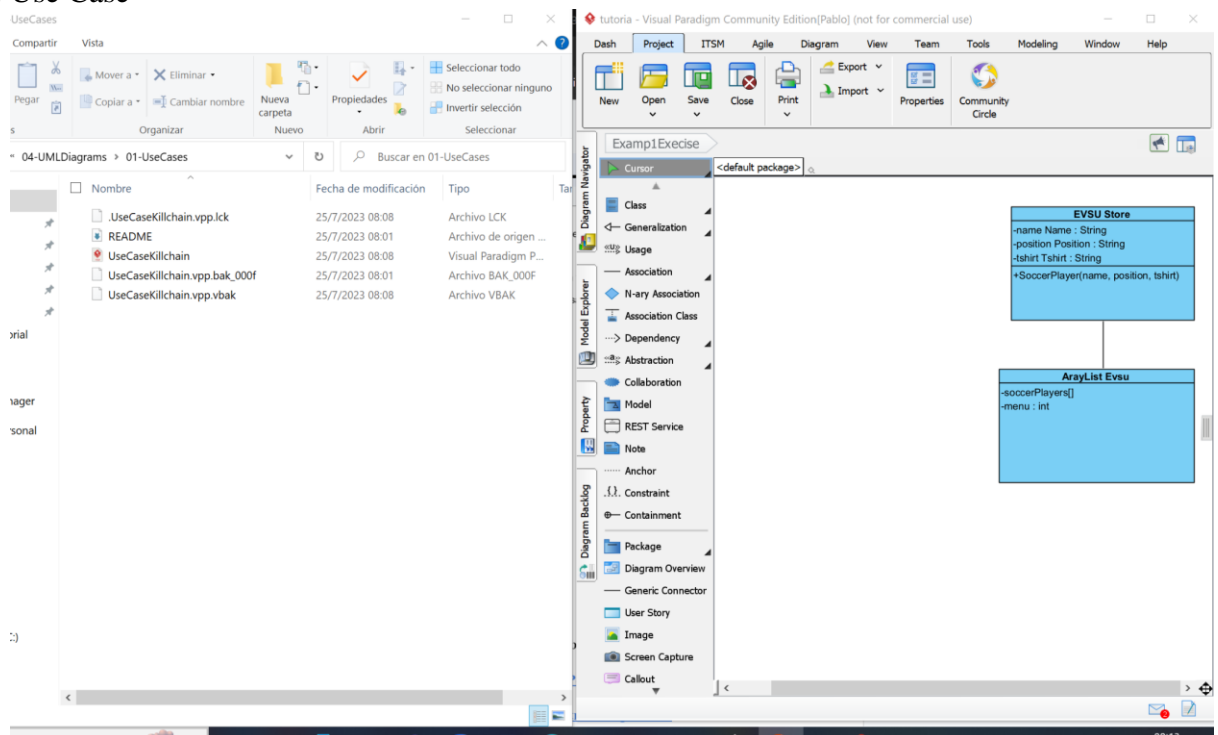
2. Project

10. Requirements

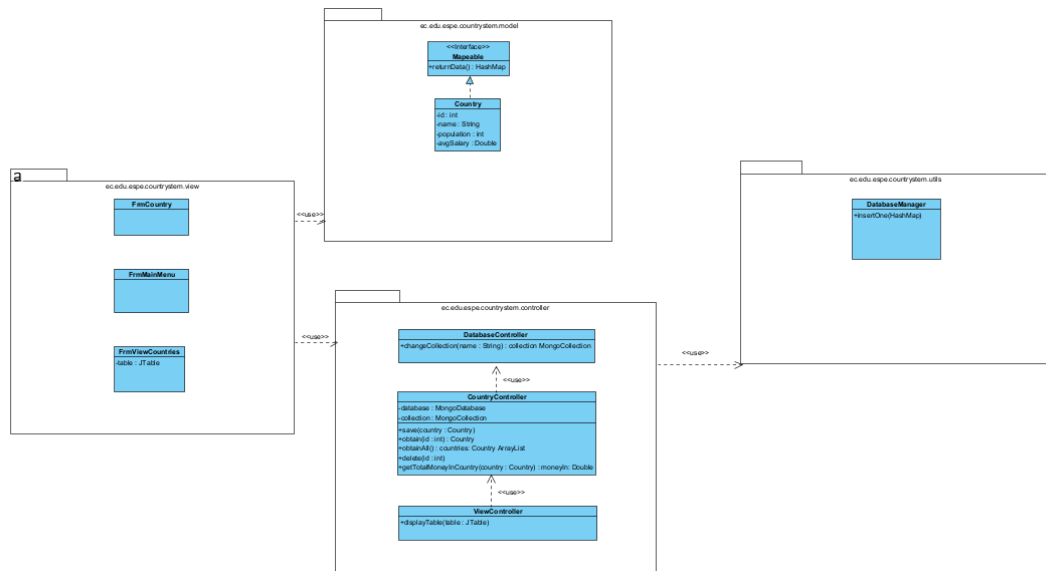
3.2 Functions

- Inventory management: add, remove and update products.
- Sales Tracking: record sales and generate reports.
- Customer Information Management: Store and update customer information.
- Vendor Recommendation: Suggest vendors based on component compatibility and quality.
- Algorithm to generate compatibility according to the components you buy.
- Implement a payment system.
- Implement a text generator to automate the creation of reviews and thus improve the business.
- You should handle different classes that contain various unique objects, since managing a store requires you to specify which item is for sale.

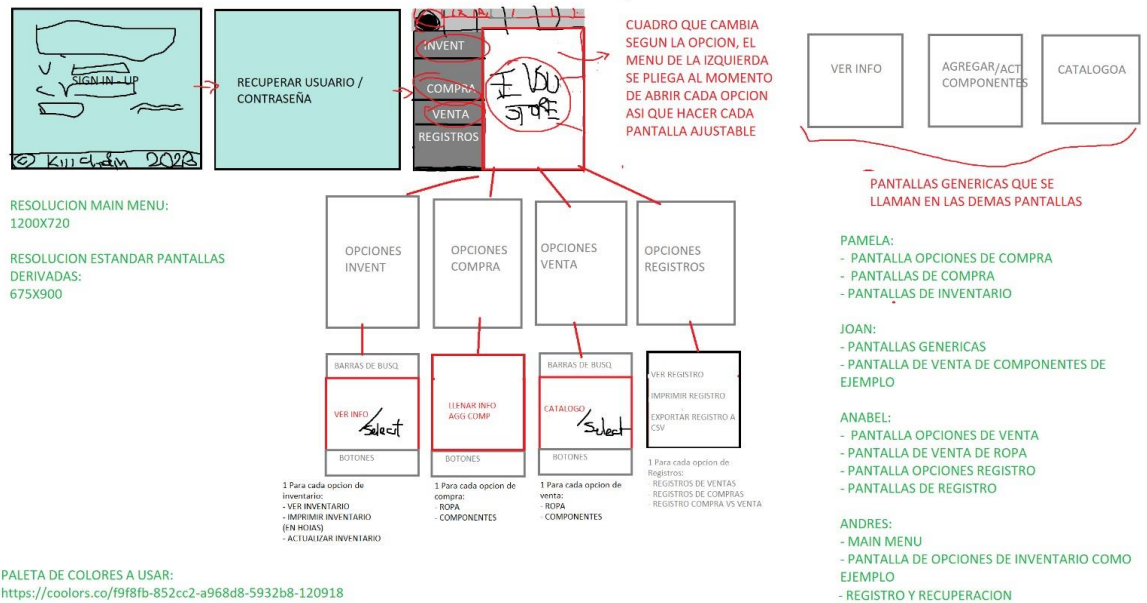
11. Use Case



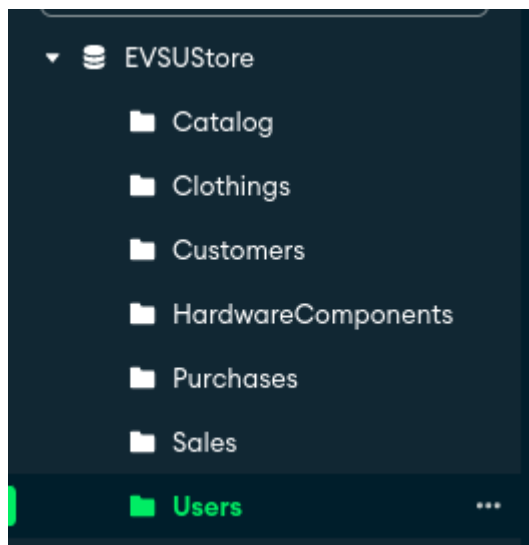
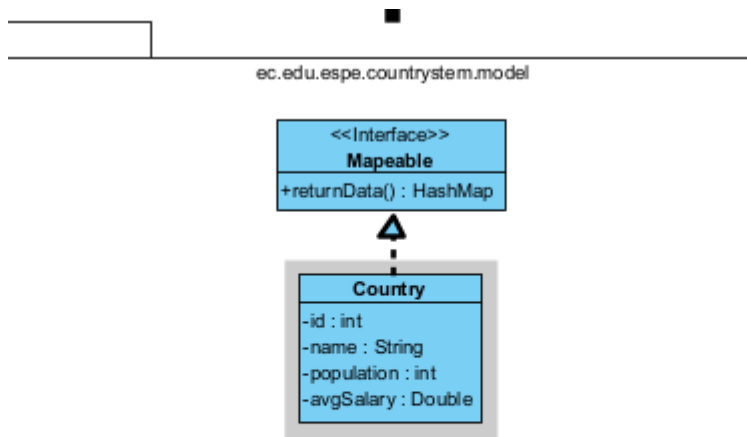
12. Class Diagram



13. Mockup



14. MongoDB



15. Test Cases

The screenshot shows an IDE window with a Java source file and an output console. The source file, located at `ec.edu.espe.evsustore.controller`, contains the following code:

```
1 package ec.edu.espe.evsustore.controller;
2
3 import ec.edu.espe.evsustore.model.Catalog;
4 import java.util.HashMap;
5 import org.junit.jupiter.api.AfterEach;
6 import org.junit.jupiter.api.AfterAll;
7 import org.junit.jupiter.api.BeforeEach;
8 import org.junit.jupiter.api.BeforeAll;
9 import org.junit.jupiter.api.Test;
10 import static org.junit.jupiter.api.Assertions.*;
11
12
13 /**
14  *
15  * @author Joan Cobeña, KillChain, DCC0-ESPE
16  */
17 public class CatalogControllerTest {
18
19     public CatalogControllerTest() {
```

The output console, titled "Output x", shows the results of running the test class `CatalogControllerTest` using the `EVSUStore` runner. The output is as follows:

```
Run (EVSUStore) x Test (CatalogControllerTest) x
Results:
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
```

Source History

```
1 package ec.edu.espe.evsustore.controller;
2
3 import ec.edu.espe.evsustore.model.Catalog;
4 import java.util.HashMap;
5 import org.junit.jupiter.api.AfterEach;
6 import org.junit.jupiter.api.AfterAll;
7 import org.junit.jupiter.api.BeforeEach;
8 import org.junit.jupiter.api.BeforeAll;
9 import org.junit.jupiter.api.Test;
10 import static org.junit.jupiter.api.Assertions.*;
11
12 /**
13  *
14  * @author Joan Cobeña, KillChain, DCCO-ESPE
15  */
16 public class CatalogControllerTest {
17     public CatalogControllerTest() {
```

Output Test Results x

ec.edu.espe:EVSUStore:jar:1.0-SNAPSHOT (Unit) x

Tests passed: 100.00 % getData

The test passed. (0.085 s)

Source History

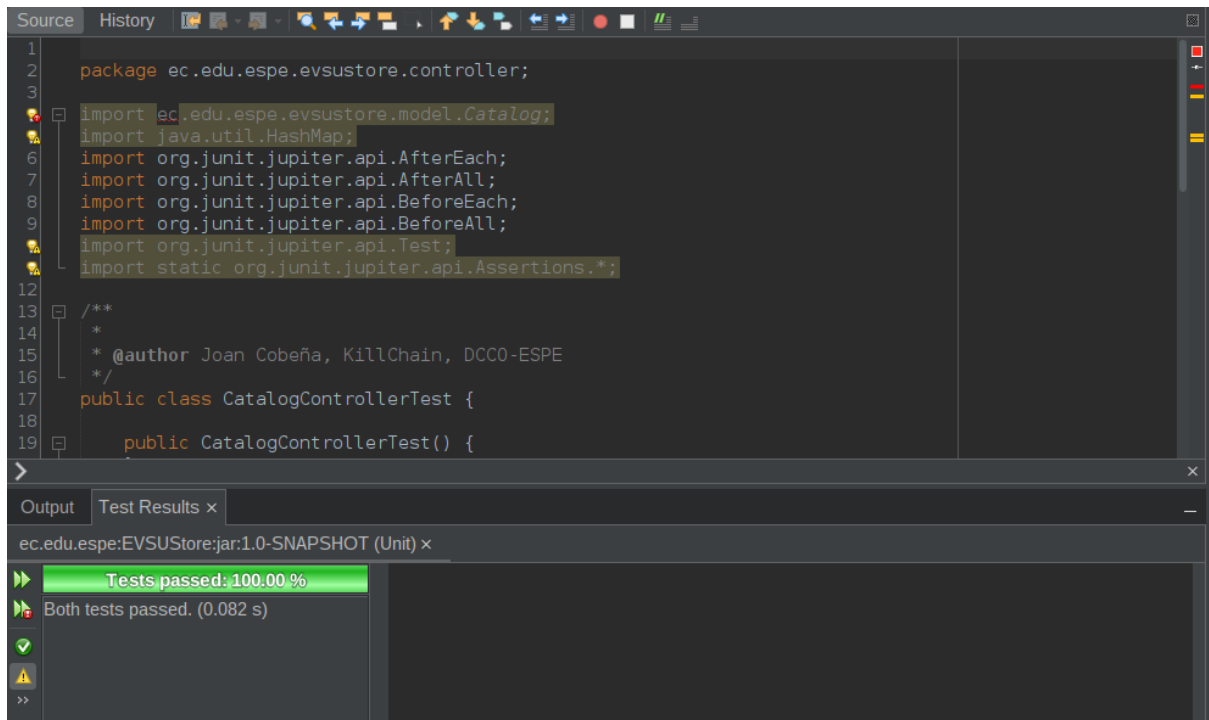
```
1 package ec.edu.espe.evsustore.controller;
2
3 import ec.edu.espe.evsustore.model.Catalog;
4 import java.util.HashMap;
5 import org.junit.jupiter.api.AfterEach;
6 import org.junit.jupiter.api.AfterAll;
7 import org.junit.jupiter.api.BeforeEach;
8 import org.junit.jupiter.api.BeforeAll;
9 import org.junit.jupiter.api.Test;
10 import static org.junit.jupiter.api.Assertions.*;
11
12 /**
13  *
14  * @author Joan Cobeña, KillChain, DCCO-ESPE
15  */
16 public class CatalogControllerTest {
17     public CatalogControllerTest() {
```

Output Test Results x

ec.edu.espe:EVSUStore:jar:1.0-SNAPSHOT (Unit) x

Tests passed: 100.00 % roundToTwoTenths

The test passed. (0.096 s)



Source History

```
50     String[] result = componentModel.split(regex: " ");
51     assertEquals(expected, result[0]);
52 }
53
54 public int getIdOfComponentModel(String componentModel){
55     int id;
56
57
58     String[] partsOfModel = componentModel.split(regex: " ");
59     id = Integer.parseInt(partsOfModel[1]);
60
61     return id;
62 }
63
64 }
65
```

ec.edu.espe.evsustore.view.PnelViewInfoTest > getIdOfComponentModel >

Output Test Results x

ec.edu.espe:EVSUStore:jar:1.0-SNAPSHOT (Unit) x

Tests passed: 50.00 %

1 test passed, 1 test caused an error. (0.107 s)

ec.edu.espe.evsustore.view.PnelViewInfoTest > testGetIdOfComponentModel c

Source History

```
37 * Test of roundToTwoTenths method, of class DecimalsControl.
38 */
39 @Test
40 public void testRoundToTwoTenths() {
41     System.out.println(x: "roundToTwoTenths");
42     Double amountToRound = 125.24768;
43     Double expectedResult = 125.0;
44     Double result = DecimalsControl.roundToTwoTenths(amountToRound);
45     assertEquals(expected: expectedResult, actual: result);
46
47 }
48
49 }
50
```

ec.edu.espe.evsustore.utils.TaxTest > testRoundToTwoTenths > expectedResult >

Output Test Results x

ec.edu.espe:EVSUStore:jar:1.0-SNAPSHOT (Unit) x

Tests passed: 0.00 % roundToTwoTenths

No test passed, 1 test failed. (0.107 s)

ec.edu.espe.evsustore.utils.TaxTest > testRoundToTwoTenths Failed

The screenshot shows an IDE with a Java file named `HardwareComponentTest`. The code defines a `@Test` method `testGetData()` that creates a `HardwareComponent` instance, initializes a `HashMap` with product details, and calls `instance.getData()`. The test fails because the returned `HashMap` is empty. The output window shows the test results: "Tests passed: 0.00 %", "No test passed, 1 test failed. (0.088 s)", and a detailed failure message for `testGetData` stating "Failed: expected: <HashMap> but was: <HashMap>".

```
42  */
43  @Test
44  public void testGetData() {
45      System.out.println(x: "getData");
46      HardwareComponent instance = new HardwareComponent(1, 3, 123, 334, "sdsa", "dsds");
47      HashMap<Object, Object> xdd = new HashMap<>();
48      xdd.put(key: "quantity", value: 2);
49      xdd.put(key: "cost", value: 123.0);
50      xdd.put(key: "price", value: 334.0);
51      xdd.put(key: "name", value: "sdsa");
52      xdd.put(key: "model", value: "dsds");
53      xdd.put(key: "id", value: 1);
54
55      HashMap<Object, Object> expResult = xdd;
56      HashMap<Object, Object> result = instance.getData();
57      assertEquals(expected:expResult, actual: result);
58
59  }
```

ec.edu.espe.evsustore.model.HardwareComponentTest > testGetData > instance >

Output Test Results x

ec.edu.espe:EVSUStore:jar:1.0-SNAPSHOT (Unit) x

Tests passed: 0.00 % getData

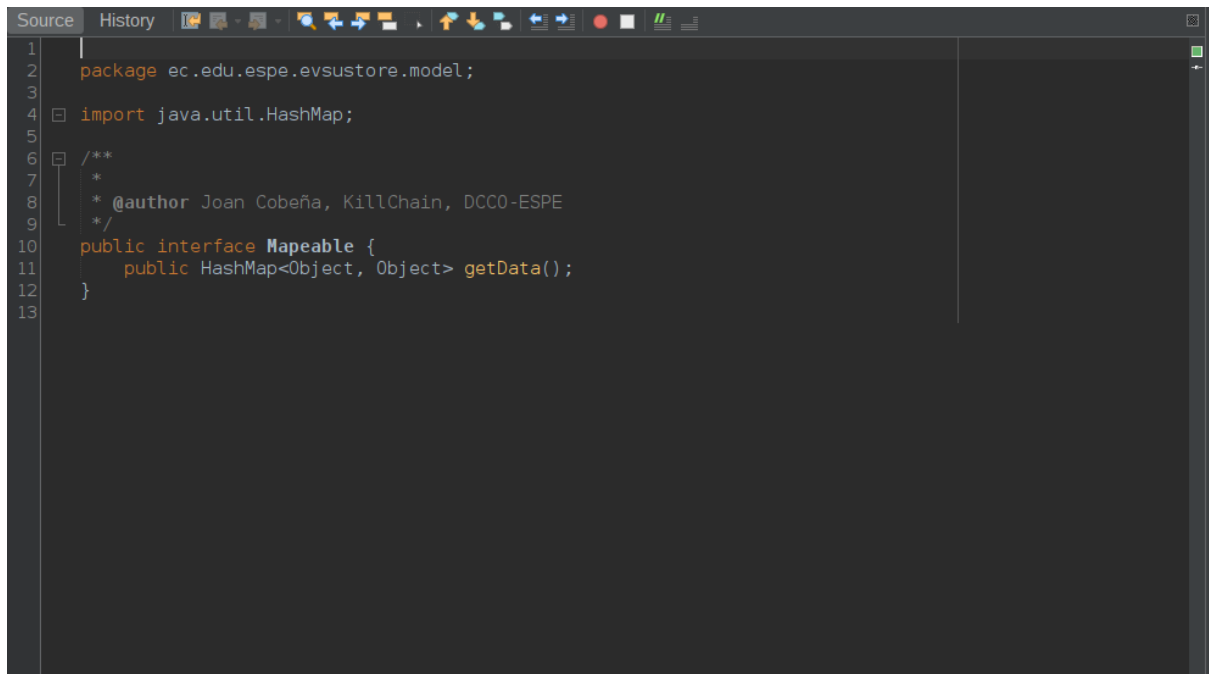
No test passed, 1 test failed. (0.088 s)

ec.edu.espe.evsustore.model.HardwareComponentTest > testGetData Failed: expected: <HashMap> but was: <HashMap>

16. OOP

The screenshot shows an IDE with a Java file named `Catalog`. The code defines a `Catalog` class that implements the `Mapeable` interface. It has attributes for `productDescription`, `id`, `quantity`, `price`, and a `HashMap` for `data`. The `toString()` method returns a string representation of the object. The constructor initializes the `data` map with the object's attributes.

```
4  import java.util.HashMap;
5
6  /**
7   *
8   * @author Joan Cobeña, KillChain, DCC0-ESPE
9   */
10 public class Catalog implements Mapeable{
11     String productDescription;
12     int id;
13     int quantity;
14     Double price;
15     HashMap<Object, Object> data;
16
17     @Override
18     public String toString() {
19         return productDescription + " Cantidad" + quantity + " Precio" + price + "\t";
20     }
21
22
23
24     public Catalog(int id, String productDescription, int quantity, Double price) {
25         data = new HashMap<>();
26         data.put(key: "id", value: id);
27         data.put(key: "productDescription", value: productDescription);
28         data.put(key: "quantity", value: quantity);
29         data.put(key: "price", value: price);
30
31         this.id = id;
32         this.productDescription = productDescription;
33     }
34 }
```



```
1 |
2 | package ec.edu.espe.evsustore.model;
3 |
4 | import java.util.HashMap;
5 |
6 | /**
7 |  *
8 |  * @author Joan Cobeña, KillChain, DCC0-ESPE
9 |  */
10 | public interface Mapeable {
11 |     public HashMap<Object, Object> getData();
12 | }
13 |
```

17. Clean GUI

```

Source Design History
298
299 private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
300
301     String username = txtUserName.getText().trim();
302     char[] passwordChars = txtPassword.getPassword();
303     String password = String.valueOf(data.passwordChars).trim();
304
305     if (username.isEmpty() || password.isEmpty()) {
306         JOptionPane.showMessageDialog(parentComponent: this, message: "Algún campo está vacío, intent
307     } else {
308         sessionController.migratePasswordsToBCrypt();
309         if (sessionController.checkCredentials(username, password)) {
310             showMessageWithoutButton(parentFrame: this, message: "Inicio de sesión exitoso", title: "E
311         } else {
312             JOptionPane.showMessageDialog(parentComponent: this, message: "Credenciales inválidas", t
313         }
314     }
315 }
316
Source Design History
167
168
169 private void btnEnviarActionPerformed(java.awt.event.ActionEvent evt) {
170     String username = txtUserName.getText();
171     String recipient = txtEmail.getText();
172     if (TextFieldValidator.isTextFieldEmpty(textField: txtUserName) || TextFieldValidator.isTextFi
173         JOptionPane.showMessageDialog(parentComponent: null, message: "Por favor, completa todos los
174     return;
175 }
176 String temporaryPassword = sessionController.generateTemporaryPassword();
177 boolean passwordUpdated = sessionController.updatePassword(username, newPassword: temporaryPass
178
179 if (!passwordUpdated) {
180     JOptionPane.showMessageDialog(parentComponent: null, message: "No se encontró ningún usuario
181     return;
182 }
183
184 String subject = "Recuperación de contraseña";
185 String body = "Hola " + username + ", tu contraseña temporal es: " + temporaryPassword;
186
187 try {
188     EmailUtils.sendRecoveryEmail(recipient, subject, body);
189     JOptionPane.showMessageDialog(parentComponent: null, message: "El correo electrónico ha sido
190 } catch (MessagingException e) {
191     JOptionPane.showMessageDialog(parentComponent: null, "Error al enviar el correo electrónico
192 }
193
194 }
195
196 private void txtUserNameActionPerformed(java.awt.event.ActionEvent evt) {
197     // regenerated by the Form Editor:
198     /**
199     *
200     */
201     @SuppressWarnings("unchecked")
202     // Generated Code
203
204
205
206
207
208
209
210 private void btnCreteUserActionPerformed(java.awt.event.ActionEvent evt) {
211     String name = txtName.getText().trim();
212     String lastName = txtLastName.getText().trim();
213     String username = txtUserName.getText().trim();
214     String password = new String(value: txtPassword.getPassword()).trim();
215
216     boolean createUserSuccess = sessionController.createUser(name, lastName, username, password);
217     if (createUserSuccess) {
218         JOptionPane.showMessageDialog(parentComponent: this, message: "Usuario creado exitosamente");
219         FrmLogin frmLogin = new FrmLogin();
220         frmLogin.setVisible(b: true);
221         dispose();
222     } else {
223         JOptionPane.showMessageDialog(parentComponent: this, message: "Error al crear el usuario. Nombre
224     }
225 }
226
227 }
228
229 private void btnCancelUserActionPerformed(java.awt.event.ActionEvent evt) {
230     FrmLogin frmLogin = new FrmLogin();
231     frmLogin.setVisible(b: true);
232     dispose();
233 }
234

```

18. Clean Code

```
Source History
1 package ec.edu.espe.evsustore.controller;
2
3 import javax.swing.JTable;
4
5 /**
6  *
7  * @author Joan Cobeña, KillChain, DCC0-ESPE
8  */
9 public class ListenersController {
10
11 }
12
```

```
Source History
1 package ec.edu.espe.evsustore.model;
2
3
4 /**
5  *
6  * @author Joan Cobeña, KillChain, DCC0-ESPE
7  */
8 public class Manager {
9
10 }
11
```

```

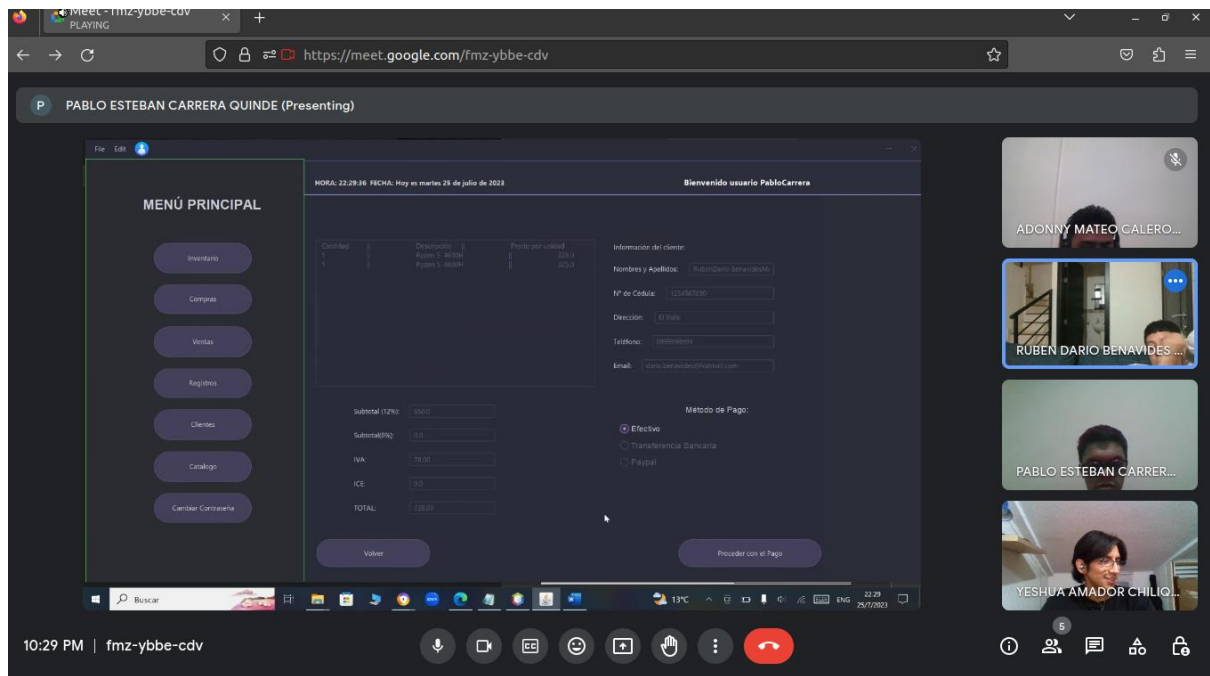
Source History
1 package ec.edu.espe.evsustore.model;
2
3
4 import java.time.LocalDate;
5 import java.util.ArrayList;
6 import java.util.Date;
7 import java.util.HashMap;
8
9 /**
10  *
11  * @author Joan Cobeña, KillChain, DCC0-ESPE
12  */
13 public class Sale implements Mapeable{
14     private int id;
15     private ArrayList<Catalog> soldComponents;
16     private int saleId;
17     private Customer customer;
18     private double salePrice;
19     private LocalDate date;
20
21     private HashMap<Object, Object> data;
22
23     public Sale(int id, ArrayList<Catalog> soldComponents, Customer customer, double salePrice, LocalDate date) {
24         data = new HashMap<>();
25         data.put(key: "id", value: id);
26         data.put(key: "soldComponents", value: soldComponents.toString());
27         data.put(key: "customer", value: customer.toString());
28         data.put(key: "salePrice", value: salePrice);
29         data.put(key: "date", value: date);
30     }
31 }

```

3. Execution

4. 4 business rules

1. Rule: Buy Components



3. Rule: Compute IVA
4. Rule: Compute ICE

5. Login

PABLO ESTEBAN CARRERA QUINDE (Presenting)

Nueva pestaña Privada

Buscar en Google o escribir una URL

Google YouTube WhatsApp Inicio - MIESPE Aula Virtual Calendar Bibizuru Online C++ Compl... Operadores C Recibidos (319) p... Classroom

EVSU Store

Usuario:

PabloCarrera

Contraseña:

pablocarreraj

¿Olvidó su contraseña?...Click aquí para recuperarla

Salir Registrarse Ingresar

9:28 PM | fmz-ybbe-cdv

https://meet.google.com/fmz-ybbe-cdv

PABLO ESTEBAN CARRERA QUINDE (Presenting)

MENÚ PRINCIPAL

- Inventario
- Compra
- Ventas
- Registros
- Cuentas
- Catalogo
- Cambiar Contraseña

HORA: 22:48:09 FECHA: Hoy es martes 25 de julio de 2023

Bienvenido usuario PabloCarrera

Cambiar Contraseña

Contraseña

Confirmar Contraseña

Confirmar

STORE EVSU

10:48 PM | fmz-ybbe-cdv

RUBEN DARIO BENAVIDES MACI...

PABLO ESTEBAN CARRERA QUI...

ADONNY MATEO CALERO ARG...

YESHUA AMADOR CHILIQUE...

6. others

meet - fmz-ybbe-cdv
PLAYING

https://meet.google.com/fmz-ybbe-cdv

PABLO ESTEBAN CARRERA QUINDE (Presenting)

The presentation slide displays the 'STORE EYS U' logo and a table with the following data:

ID	DESCRIPCION	CANTIDAD

Below the table, there is a 'Formulario de Pago' section with the following fields:

- Nombre completo del cliente
- Identificación
- Correo electrónico
- Teléfono
- Correo electrónico

Participants:

- ADONNY MATEO GALERO...
- RUBEN DARIO BENAVIDES...
- PABLO ESTEBAN CARRER...
- YESHUA AMADOR CHILIQ...

10:06 PM | fmz-ybbe-cdv

Controls: Microphone, Video, Chat, Screen Share, Hand, More Options, End Call

5

Info, Participants, Chat, Screen Share, Lock

Print Fature

Responder Responder a todos Reenviar Archivar

EVSU STORE FACTURA



kiboki1234@hotmail.com <kiboki1234@hotmail.com>

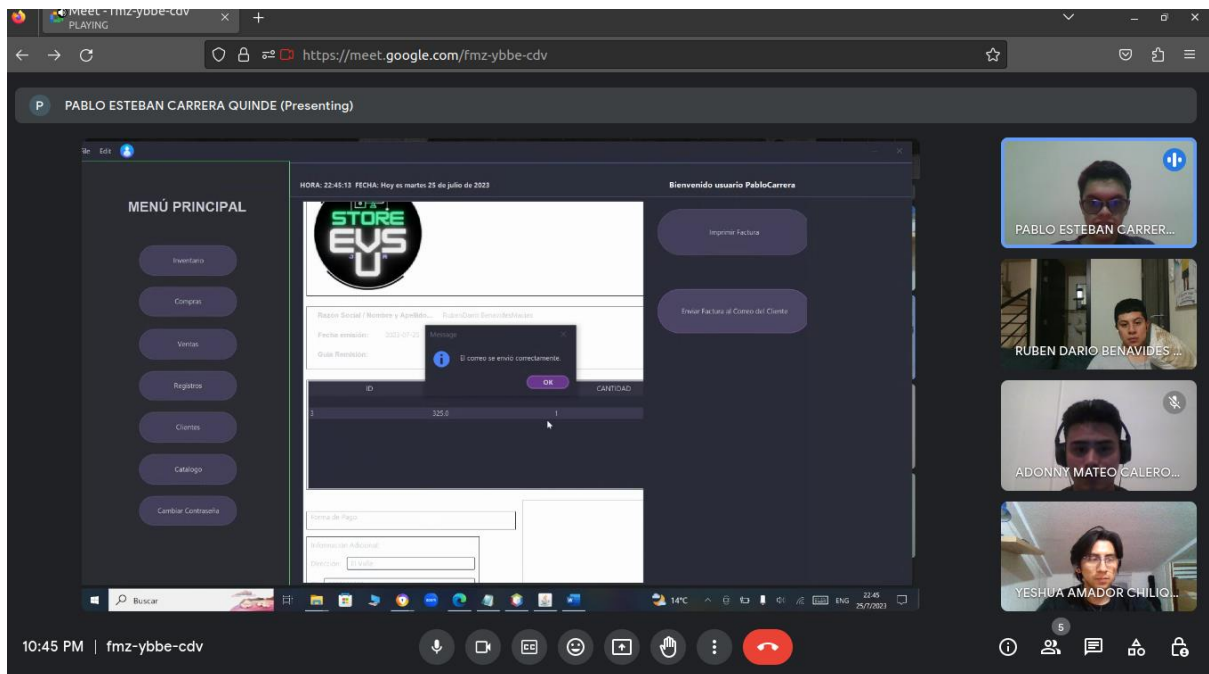
10:41:PM

Para: dario.benavides@hotmail.com



factura_20230725_224129.pdf
45.93 KB

Se adjunta la factura de su compra realizada, gracias por preferirnos.



Send to email