

UNIVERSIDAD DE LAS FUERZAS ARMADAS

ESPE



Object-oriented programming

NRC: 9642

U2 Workshop N° 40

Topic: Final Project Inspection

Ing. Jorge Edison Lascano. PhD

Rubric:	
Project GitHub	/25 (Individual Contribution)
Project Review (Inspection)	/25
Requirements and modeling (updated and design patterns)	/5
Principles,CleanCode, Pitfalls	/5
Functional Requirements user test (No CRUD operations)	/5
MongoDB	/5
Design patterns code	/5
Project Maintenance	/25 (Time effort)
new user / user type	/10
report screen	/10
print out	/5
Project Letter	/25 (Letter of acceptance or review)

Evidence:

Project GitHub

Project Review (Inspection)

 Requirements and modeling (updated and design patterns)

 Principles,CleanCode, Pitfalls

 Functional Requirements user test (No CRUD operations) **7:25**

 MongoDB

 Design patterns code

Project Maintenance

Project Letter

Team 1: Jsons Leader: Yeshua Chiliquinga

Inspector: KillChain Leader: Anabel Davila

Project: Odontoapp

GitHub: <https://github.com/YeshuaChiliquingaAmaya/Jsons>

MongoDB Compass:

"mongodb+srv://RBenavides:RBenavides@cluster0.js2ve9m.mongodb.net/"

Grade: PR: /25, PM: /25, Letter: /25 → 59.5/75

1. BENAVIDES MACIAS RUBEN DARIO
2. CALERO ARGUELLO ADONNY MATEO
3. CARRERA QUINDE PABLO ESTEBAN
4. CHILIQUINGA AMAYA YESHUA AMADOR

Rubric:

Project Review (Inspection)	17.5/25
Requirements and modeling (updated and design patterns)	5/5
Principles,CleanCode, Pitfalls	3/5
Functional Requirements user test (No CRUD operations)	3+/5
MongoDB	1.5+2.5= 4/5
Design patterns code	2.5/5
Project Maintenance	17/25 (Time effort)
new user / user type	7/10
report screen	10/10
print out	0/5
Project GitHub	/25 (Individual Contribution)
Project Letter	25/25 (Letter of acceptance or review)

Evidence:

Project GitHub

Project Review (Inspection)

Requirements and modeling (updated and design patterns)

- The system will have a login and password.
- **The system will be able to purchase services for a determined patient and show the corresponding bill of what he/she bought.**
- **The system will be able to print this information physically, and or save on a digital file e.g pdf.**
- The system will be able to schedule appointments for a determined patient.

Principles,CleanCode, Pitfalls

Functional Requirements user test (No CRUD operations)

OdontoApp.Password

Documents

Aggregations

Schema

Explain Plan

Indexes

Filter 

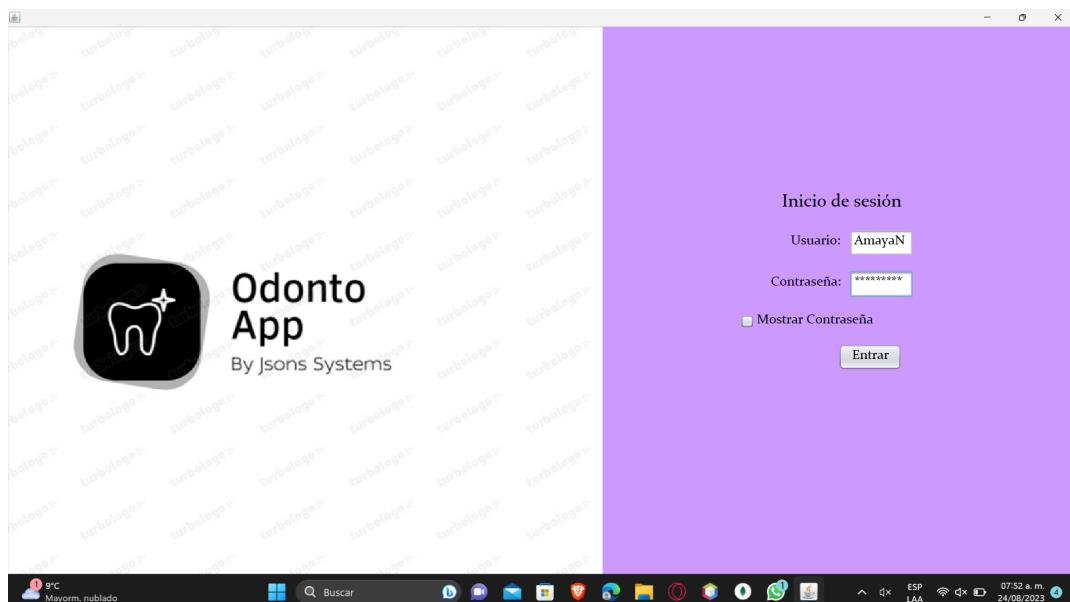


Type a query: { field: 'value' }

 ADD DATA

 EXPORT DATA

```
_id: ObjectId('64ac9dc3d68d2555dc755c55')
username: "DpdbcQ"
password: "RgrqwrDss"
```



[Login](#)

Servicio

- Profilaxis (limpieza dental): \$25
- Profilaxis + Fluorización (especialmente para niños): \$20
- Restauraciones simples: \$20
- Restauraciones complejas (Las restauraciones pueden ser con resina o amalgama): \$25
- Restauraciones para niños (empastes para niños): \$15
- Extracciones (extracción de un diente): \$20
- Extracciones complicadas: \$30**
- Tratamiento de un conducto (incisivos o dientes frontales): \$60
- Tratamiento de dos conductos (premolares): \$80
- Tratamiento de tres conductos (molares): \$100

Añadir al carrito

Eliminar del carrito

Comprar

Regresar

Factura

Fecha: 24/08/2023

Nombre del paciente: KillChainTeam

Compras Realizadas

Servicio	Costo
Extracciones complicadas	30.0

Precio Final: 30.0

Imprimir

Impresora

Nombre: Microsoft Print to PDF

Estado: Listo

Tipo: Microsoft Print To PDF

Ubicación: PORTPROMPT:

Comentario:

Intervalo de impresión

Todo

Páginas de: 1 a: 9999

Selección

Copias

Número de copias: 1

Intercalar

Imprimir a un archivo

Aceptar

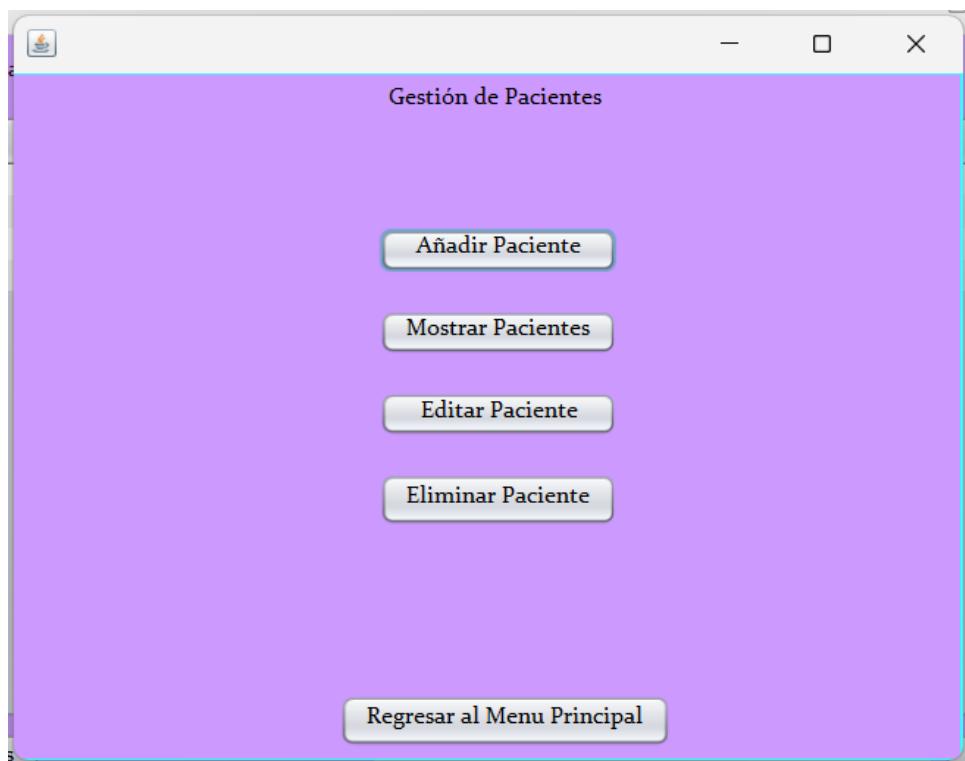
Cancelar

Total
30.0

Descuento
Ninguno

Regresar Al Menú Principal

Acquire Services with bill able to print



Id: Ingrese solo numeros enteros

Nombre:

Edad:

Altura en m (separe con puntos):

Peso en kg (separe con puntos):

Número de contacto:

Tratamiento del Paciente:

Sintomas:

Enfermedades Sistémicas:

Fecha de Inicio de Tratamiento:

Fecha de Finalización de Tratamiento:

```
_id: ObjectId('64e74e64a03e5f184a778477')
▼ clinicalHistory: Object
  id: 4
  name: "KillChainTeam"
  age: 20
  weight: 75
  height: 1.9
  diseaseSymptoms: "Si"
  cellphone: "0968727003"
  systemicDiseases: "No"
  patientTreatment: "Cuidado general"
  treatmentDateStart: "Aug 24, 2023, 12:00:00 AM"
  treatmentEndDate: "Aug 27, 2023, 7:33:26 AM"
  ▼ acquiredServices: Array (empty)
  ▼ appointments: Array (empty)
```

CRUD is working

MongoDB

```
_id: ObjectId('64e74e64a03e5f184a778477')
▼ clinicalHistory: Object
  id: 4
  name: "KillChainTeam"
  age: 20
  weight: 75
  height: 1.9
  diseaseSymptoms: "Si"
  cellphone: "0968727003"
  systemicDiseases: "No"
  patientTreatment: "Cuidado general"
  treatmentDateStart: "Aug 24, 2023, 12:00:00 AM"
  treatmentEndDate: "Aug 27, 2023, 7:33:26 AM"
  ▼ acquiredServices: Array (empty)
  ▼ appointments: Array (empty)
```

Schedule an appointment update the appointments in DB:

```

_id: ObjectId('64e74e64a03e5f184a778477')
clinicalHistory: Object
  id: 4
  name: "KillChainTeam"
  age: 20
  weight: 75
  height: 1.9
  diseaseSymptoms: "Si"
  cellphone: "0968727003"
  systemicDiseases: "No"
  patientTreatment: "Cuidado general"
  treatmentDateStart: "Aug 24, 2023, 12:00:00 AM"
  treatmentEndDate: "Aug 27, 2023, 7:33:26 AM"
acquiredServices: Array
appointments: Array
  0: Object
    date: "Aug 25, 2023, 12:00:00 AM"
    hour: "10:00 am"
    service: "Profilaxis (limpieza dental): $25"

```

The acquired service request is not updated in the database

```

_id: ObjectId('64e74e64a03e5f184a778477')
clinicalHistory: Object
acquiredServices: Array (empty)
appointments: Array (1)
  0: Object
    date: "Aug 25, 2023, 12:00:00 AM"
    hour: "10:00 am"
    service: "Profilaxis (limpieza dental): $25"

```

Design patterns code

```

private static LoginController uniqueInstance;

public static LoginController getInstance(JTextField txtUsername, JTextField txtPassword) {
    if(uniqueInstance == null){
        uniqueInstance = new LoginController(txtUsername, txtPassword);
    }
    return uniqueInstance;
}

private LoginController(JTextField txtUsername, JTextField txtPassword) {
    this.txtUsername = txtUsername;
    this.txtPassword = txtPassword;
}

```

Just 1 Design Pattern, Singleton in LoginController

Project Maintenance
Project Letter

Team 2: KillChainTeam Leader: Anabel Dávila.

Inspector: Leader: Carlos Jaya

Project: EVSUStore

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

MongoDB Compass: <mongodb+srv://jcobena:kcobena@cluster0.mhpiaao.mongodb.net/>

Rubric:

Project GitHub /25 (Individual Contribution)

Project Review (Inspection) 19.5/25

Requirements and modeling (updated and design patterns) 4/5

Principles, CleanCode, Pitfalls 4.5/5

Functional Requirements user test (No CRUD operations) 4.5/5

MongoDB 4/5

Design patterns code 2.5/5

Project Maintenance 10/25 (Time effort)

new user / user type 5/10

report screen 5/10

print out 0/5

Project Letter 0/25 (Letter of acceptance or review)

Evidence:

Project GitHub

Project Review (Inspection)

Requirements and modeling (updated and design patterns)

Principles, CleanCode, Pitfalls

Functional Requirements user test (No CRUD operations)

MongoDB

Design patterns code

Grade: PR: /25, PM: /25, Letter: /25 → 29.5 /75

5. CHIPE ZAMBRANO PAMELA NAOMI

6. COBEÑA ZAMBRANO JOAN OSWALDO

7. DAVILA MOLINA ANABEL

8. ESPIN ANDRADE ANDRES ALEXANDER

Evidence:

Project Review:

Requirements and modeling (updated and design patterns) 4/5

```
public synchronized static DatabaseController getInstance () {
    if (instance != null){
        }
    else {
        instance = new DatabaseController();
    }
    return instance;
}
```

```
public interface Mapeable {  
    public HashMap<Object, Object> getData();  
}
```

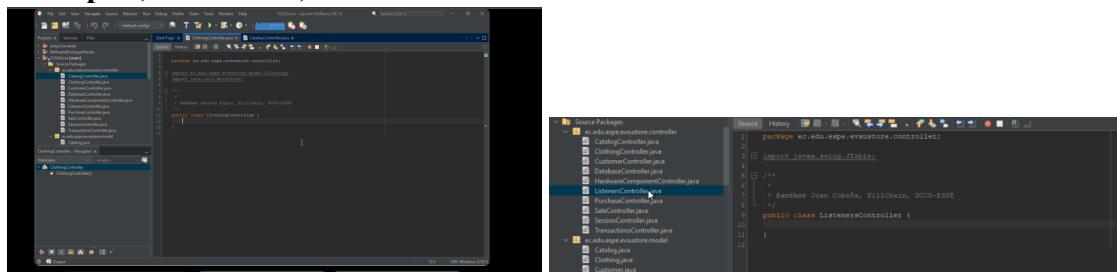


- Inventory management: add, remove and update products.
 - Sales Tracking: record sales and generate reports.
 - Customer Information Management: Store and update customer information.
 - Vendor Recommendation: Suggest vendors based on component compatibility and quality.
 - Algorithm to generate compatibility 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.
 - Different types of data are formed since the project tightened it in the same way, the encapsulation of functions will be essential to have a better control of the code.

* The UML Diagram is not update*

Principles,CleanCode, Pitfalls

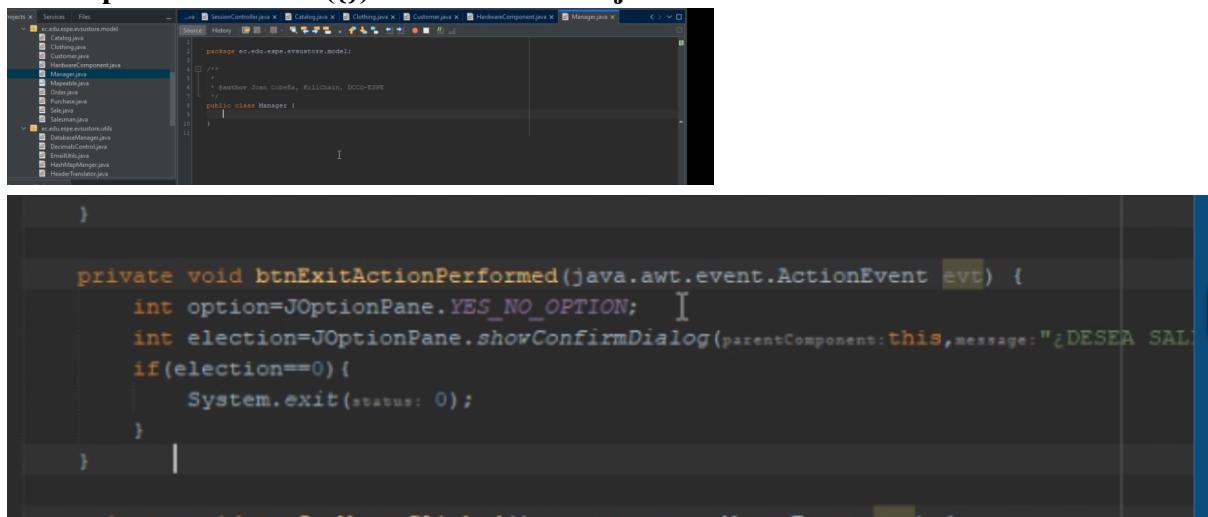
/5



```
private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
    int option=JOptionPane.YES_NO_OPTION;
    int election=JOptionPane.showConfirmDialog(parentComponent:this,message:"¿DESEA SALIR?",title: "SALIR",optionType: opt
    if(election==0){
        System.exit(status: 0);
    }
}
```

LazyClass

Bad implementation of ({}). CleanCode NOT just one line of code inside the if sentence



```
private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
    int option=JOptionPane.YES_NO_OPTION;
    int election=JOptionPane.showConfirmDialog(parentComponent:this,message:"¿DESEA SALIR?",title: "SALIR",optionType: opt
    if(election==0){
        System.exit(status: 0);
    }
}
```

Button contain code

```
private void btnCreateUserActionPerformed(java.awt.event.ActionEvent evt) {
    String name = txtName.getText().trim();
    String lastName = txtLastName.getText().trim();
    String username = txtUsername.getText().trim();
    String password = new String(txtPassword.getPassword()).trim();

    boolean createUserSuccess = sessionController.createUser(name, lastName, username, password);
    if (createUserSuccess) {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Usuario creado exitosamente");
        FrmLogin frmLogin = new FrmLogin();
        frmLogin.setVisible(b: true);
        dispose();
    } else {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Error al crear el usuario. Nombre de usuario en");
    }
}
```

```
private void btnCreateUserActionPerformed(java.awt.event.ActionEvent evt) {
    String name = txtName.getText().trim();
    String lastName = txtLastName.getText().trim();
    String username = txtUsername.getText().trim();
    String password = new String(txtPassword.getPassword()).trim();

    boolean createUserSuccess = sessionController.createUser(name, lastName, username, password);
    if (createUserSuccess) {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Usuario creado exitosamente");
        FrmLogin frmLogin = new FrmLogin();
        frmLogin.setVisible(b: true);
        dispose();
    } else {
        JOptionPane.showMessageDialog(parentComponent:this, message:"Error al crear el usuario. Nombre de usuario en");
    }
}
```

Default Name of Button

```

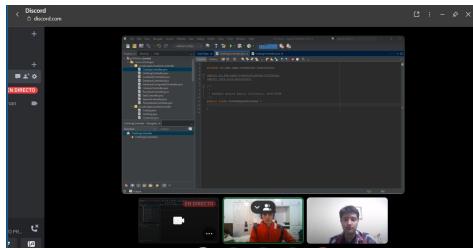
/*
@SuppressWarnings("unchecked")
Generated Code

private void jRadioButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jRadioButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

Evidence Meet



Functional Requirements user test (No CRUD operations) **4.5/5**

Functional Requirements

- 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

Evidence:

✉ kiboki1234@hotmail.com
Para: Usted
Jue 24/8/2023 7:26

factura_20230824_072631.pdf Descargado

Iniciar respuesta con: Recibido, gracias. Gracias por la factura. Muchas gracias.

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

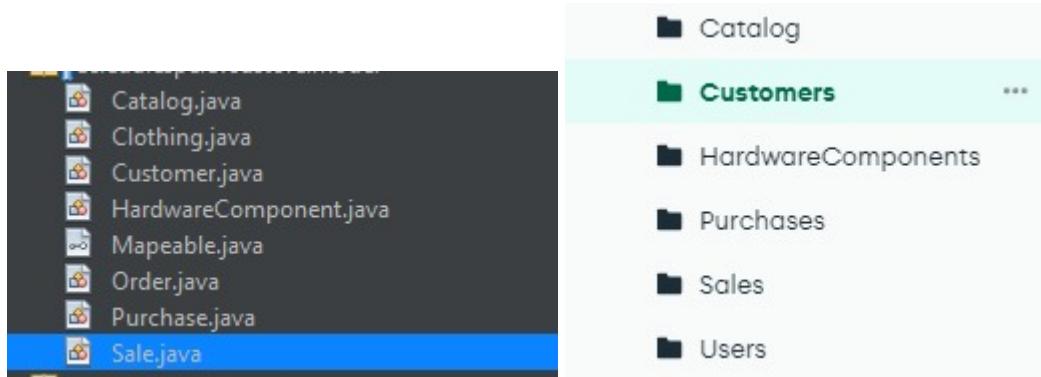
Responder Reenviar

```

_id: ObjectId('64e74c6c1ecbc6790faa15c3')
lastName: "FloresMaya"
phoneNumber: "0983372690"
name: "ALejandroCaetano"
idCardNumber: "1724621212"
location: "Quito"
id: 4
email: "caetano.flores@hotmail.com"

```

MongoDB **/5**



Clothing Class is not implemented in the MongoDB.

Design patterns code

/5

```
public synchronized static DatabaseController getInstance () {  
    if (instance != null) {  
  
    }  
    else {  
        instance = new DatabaseController();  
    }  
  
    return instance;  
}
```

Singleton Pattern in the Connection.

Team 3: Leader: Carlos Jaya

Inspector: Team 4:Code Crafters Leader: Josue Marin

Project:

GitHub: <https://github.com/cajaya1/MGOOP-SOFTWAREJUNIORS>

MongoDB Compass:

<mongodb+srv://FCaetano:FCaetano@cluster0.erktrrv.mongodb.net/>

Grade: PR: /25, PM: /25, Letter: /25 → 47 /75

9. FLORES MAYA ALEJANDRO CAETANO

10. GUAMAN ALCIVAR JORDAN ALEXANDER

11. JAYA HERRERA CARLOS ANDRES

12. JUMBO SALCEDO BRYAN ALEXANDER

Rubric:

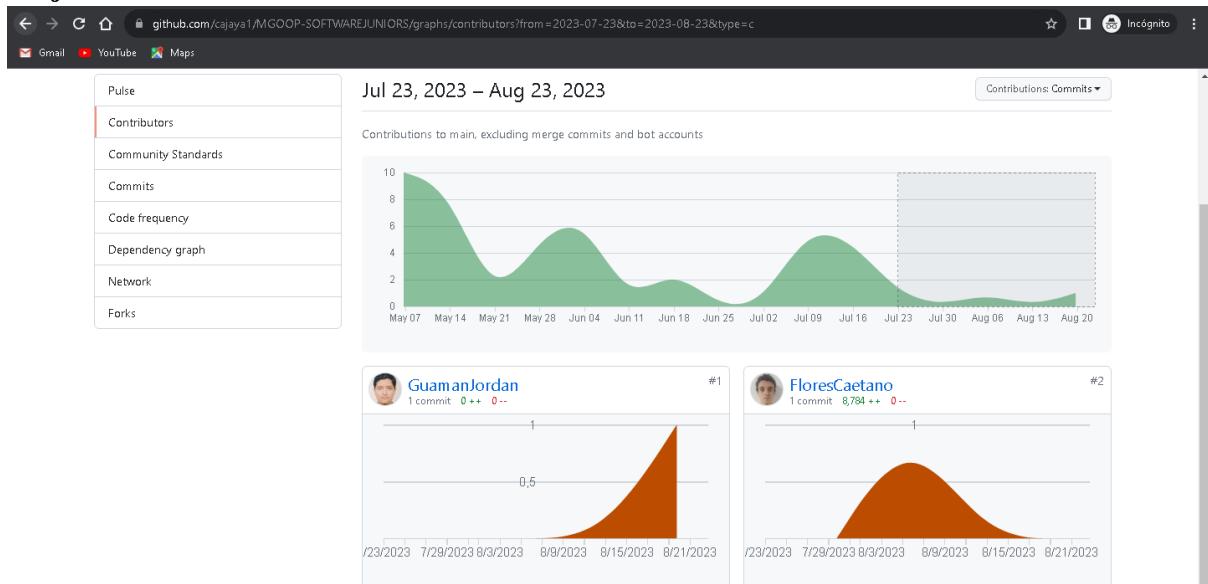
Project Review (Inspection)	19.5/25
Requirements and modeling (updated and design patterns)	3.5/5
Principles, CleanCode, Pitfalls	4/5
Functional Requirements user test (No CRUD operations)	4/5
MongoDB	4/5
Design patterns code	4/5
Project Maintenance	16/25 (Time effort)
new user / user type	1/10
report screen	10/10
print out	5/5

Project GitHub /25 (Individual
TOTAL:
Contribution)

13. FLORES MAYA ALEJANDRO CAETANO	25
14. GUAMAN ALCIVAR JORDAN ALEXANDER	25
15. JAYA HERRERA CARLOS ANDRES	0
16. JUMBO SALCEDO BRYAN ALEXANDER	0

Project Letter **12/25 (Letter of acceptance or review)**

Evidence: Project GitHub



© 2023 GitHub, Inc.

Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Project Review (Inspection) Requirements and modeling (updated and design patterns)

3.2. Functional Requirements

Product Registration: The system will allow you to register dresses (by specific object).

Product Elimination: The system will allow you to eliminate a product by searching for its name.

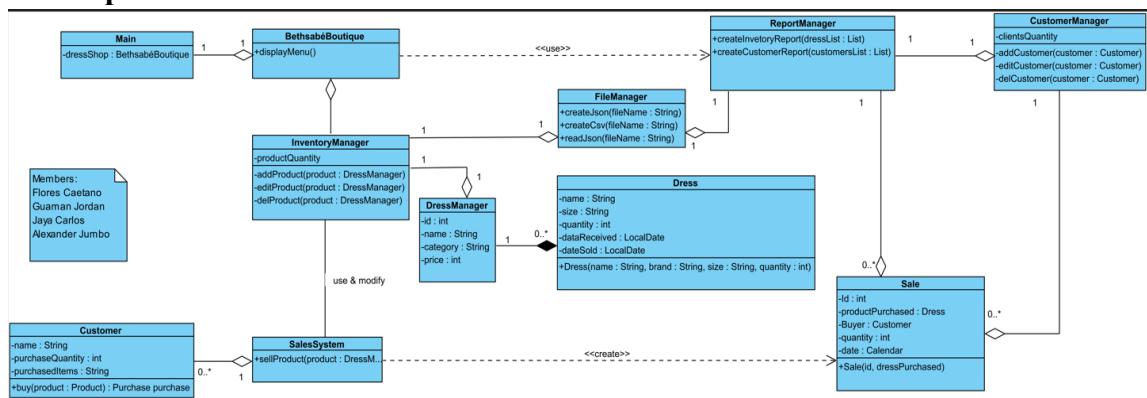
Product Edition: The system will allow you to edit the details of the dress. The system will allow manual registration of customers to the database.

Product printing(generate): Through a menu option it is possible to obtain a document external to the console in which the data entered by the console is located.

Product Sale: If at least one dress was previously added to inventory, it can be sold, with the current billing application for sale to the public.

Product printing(console): The existing dresses prior to the generation of the file are displayed in the console.

don't Update UML %



Principles,CleanCode, Pitfalls

```

    }

    public boolean checkProductAlreadyInCart(int id, MongoCollection collection) { //If is on cart adds 1 to quantity
        Document filter = new Document(key: "id", value: id);
        FindIterable<Document> documents = collection.find(bson: filter);
        int quantity = 0;
        for (Document document : documents) {
            quantity = document.getInteger(key: "quantity");
        }
        Document update = new Document(key: "$set", new Document(key: "quantity", quantity+1));
        if(quantity != 0) {
            collection.updateOne(bson: filter, bsonl: update);
            return true;
        }else {
            return false;
        }
    }

    public FindIterable<Document> retrieveDocuments(String collectionName) {
        collection = database.getCollection(string:collectionName);
        Document filter = new Document(); // Filtro vacío para obtener todos los documentos
        FindIterable<Document> documents = collection.find(bson: filter);
        return documents;
    }

    public float calculateTotal() {

```

Comments in Spanish and unnecessary

```

1 package ec.edu.espe.bethsabeboutique.view;
2
3 import com.mongodb.client.MongoClient;
4 import java.util.regex.Pattern;
5 import java.util.regex.Matcher;
6 import ec.edu.espe.bethsabeboutique.controller.DressController;
7 import ec.edu.espe.bethsabeboutique.controller.InventoryController;
8 import ec.edu.espe.bethsabeboutique.model.FormData;
9 import ec.edu.espe.bethsabeboutique.utils.RegularValidator;
10 import javax.swing.JOptionPane;
11
12 /**
13 * @author caeta
14 */
15 public class PnlInsertProduct extends javax.swing.JFrame {
16     /**
17      * Creates new form LoginPanel
18      */
19     public PnlInsertProduct() {
20         initComponents();
21     }

```

Import libraries unnecessary because not uses

Functional Requirements user test (No CRUD operations)

Product printing(console): The existing dresses prior to the generation of the file are displayed in the console.

The screenshot shows a Java application window titled "Reporte de inventario". An "Open" file dialog is displayed, showing a list of folders: Documentos, Escritorio, Este equipo, and Red. The "Folder name:" field is set to "C:\Users\Usuario\OneDrive\Documentos" and the "Files of type:" field is set to "All Files". Below the dialog, an Excel spreadsheet titled "ReporteInventario.csv - Excel" is open, showing a table of inventory items. The table has columns: id, precio, nombre, talla, Fecha de regis, and Cantidad. The data consists of multiple rows of pants (Pantalon) with various IDs, prices (e.g., 69, 70, 71, 72, 73, 74, 75, 76, 77), sizes (e.g., 10.0, 34), and quantities (e.g., 15).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	A1																	
2		id	precio	nombre	talla	Fecha de regis	Cantidad											
3		69	10.0	Pantalón		34	24/08/2023	15										
4		70	10.0	Pantalón		34	24/08/2023	15										
5		71	10.0	Pantalón		34	24/08/2023	15										
6		72	10.0	Pantalón		34	24/08/2023	15										
7		73	10.0	Pantalón		34	24/08/2023	15										
8		74	10.0	Pantalón		34	24/08/2023	15										
9		75	10.0	Pantalón		34	24/08/2023	15										
10		76	10.0	Pantalón		34	24/08/2023	15										
11		77	10.0	Pantalón		34	24/08/2023	15										

Generate report in excel

Product Sale: If at least one dress was previously added to inventory, it can be sold, with the current billing application for sale to the public.

The screenshot shows a MongoDB Compass interface. On the left, a PDF document titled "FACTURA" is displayed, containing fields for a client's name, address, and a table of items with columns for id, nombre, precio, and cantidad. The total price is listed as 28.00. On the right, the MongoDB database "BethsabeBoutique.Carts" is shown with two documents:

```

_id: ObjectId('64e752dda75b3214ee2ba076')
id: 80
name: "Pantalon"
price: 10
quantity: 1

_id: ObjectId('64e752e0a75b3214ee2ba077')
id: 86
name: "Camisa"
price: 7.5
quantity: 2

```

The shop yes sell and generate pdf and show the MongoDB

MongoDB CRUD

Create

A Java application window titled "Añadir Producto" is shown, with fields for Nombre (Chaqueta), Precio (20), Cantidad (45), and Talla (34). Below the form is a red button labeled "AÑADIR". In the background, a terminal window shows MongoDB logs indicating successful insertions of new documents into the "Carts" collection.

```

import com.mongodb.client.MongoClient;
import ec.edu.espe.bethsabeboutique.utils.DbConnectionManager;
import java.util.List;
import org.bson.Document;
import org.bson.conversions.Bson;

public class Pnl {
    public static void main(String[] args) {
        MongoClient mongoClient = DbConnectionManager.getConnection();
        Document product = new Document("Nombre", "Chaqueta")
            .append("Precio", 20)
            .append("Cantidad", 45)
            .append("Talla", 34);
        mongoClient.getDatabase("BethsabeBoutique").getCollection("Carts").insertOne(product);
        System.out.println("Documento insertado con éxito.");
        mongoClient.close();
    }
}

```

```

07:53:28.164 [cluster-ClusterId{value='64e74cfdea679e34cc3605b0', description='null'}-ac-fg8uheshard-00-01.erktrrv.mongodb.net:27017
07:53:28.164 [cluster-ClusterId{value='64e74cfdea679e34cc3605b0', description='null'}-ac-fg8uheshard-00-01.erktrrv.mongodb.net:27017
07:53:38.054 [cluster-ClusterId{value='64e74cfdea679e34cc3605b0', description='null'}-ac-fg8uheshard-00-02.erktrrv.mongodb.net:27017
07:53:38.054 [cluster-ClusterId{value='64e74cfdea679e34cc3605b0', description='null'}-ac-fg8uheshard-00-02.erktrrv.mongodb.net:27017
07:53:38.101 [cluster-ClusterId{value='64e74cfdea679e34cc3605b0', description='null'}-ac-fg8uheshard-00-00.erktrrv.mongodb.net:27017
07:53:38.101 [cluster-ClusterId{value='64e74cfdea679e34cc3605b0', description='null'}-ac-fg8uheshard-00-00.erktrrv.mongodb.net:27017

```

BethsabeBoutique.Dresses

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options

Project { field: 0 }

Sort { field: -1 } or [['field', -1]] MaxTimeMS 60000

Collation { locale: 'simple' } Skip 0 Limit 0

ADD DATA EXPORT DATA

```
_id: ObjectId('64e75284ea679e34cc3605c2')
id: 86
price: 7.5
name: "Camisa"
size: "M"
quantity: 21
registerDate: "24/08/2023"

_id: ObjectId('64e752deea679e34cc3605c3')
id: 87
price: 20
name: "Chaqueta"
size: "34"
quantity: 45
registerDate: "24/08/2023"
```

READ

Catálogo

Buscar:

id	nombre	precio	fecha registro	cantidad
80	Pantalón	10.0	24/08/2023	14
86	Camisa	7.5	24/08/2023	19
87	Chaqueta	20.0	24/08/2023	45

Añadir Producto **Eliminar Producto** **Editar Producto**

UPDATE

BethsabeBoutique.Dresses

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' }

Project { field: 0 }

Sort { field: -1 } or [['field', -1]]

MaxTimeMS 60000

Collection { locale: 'simple' }

Skip 0 Limit 0

ADD DATA EXPORT DATA

_id: ObjectId('64e75096ea679e34cc3605bc')
id: 80
price: 10
name: "Pantalon"
size: "34"
quantity: 14
registerDate: "24/08/2023"

_id: ObjectId('64e752deea679e34cc3605c3')
id: 87
price: 28
name: "Chaqueta"
size: "34"
quantity: 45
registerDate: "24/08/2023"

Design patterns code

```

1 package ec.edu.espe.bethsabeboutique.utils;
2
3 import com.mongodb.MongoException;
4 import com.mongodb.client.MongoClient;
5 import com.mongodb.client.MongoClients;
6 import javax.swing.JOptionPane;
7
8 public class DbConnectionManager {
9
10    public static DbConnectionManager instance;
11    private MongoClient client;
12
13    public void connectDb() {
14        String url = "mongodb+srv://PCaetano:PCaetano@cluster0.erkrtrv.mongodb.net/?retryWrites=true&w=majority";
15        try {
16            client = MongoClients.create(url);
17        } catch (MongoException e) {
18            JOptionPane.showMessageDialog(null, message: "Error al conectar con la base de datos!", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
19        }
20
21        public void closeConnection() {
22            getClient().close();
23        }
24
25        private DbConnectionManager() {
26
27        }
28
29        public static synchronized DbConnectionManager getInstance() {
30            if (instance == null) {
31                instance = new DbConnectionManager();
32            }
33            return instance;
34        }
35
36        /**
37         * @return the client
38     }
39 }

```

singleton

Project Maintenance

Project Letter

Team 4: Code Crafters Leader: Josue Marin

Inspector: Team 5 The FAMSE Leader: Anthony Morales

Project: Hardware Store

GitHub: <https://github.com/marinjosue/ESPE2305-CodeCrafters.git>

Compass: <mongodb+srv://josuemarin@cluster0.lntjz9j.mongodb.net/>

Grade: PR: /25, PM: /25, Letter: /25 → 54/75

17. MANOBANDA CHABLA JEFFREY ALEXANDER

- 18. MARCILLO MORA JHORDY PAUL**
- 19. MARIN ALQUINGA JOSUE ISAAC**
- 20. MARISCAL OÑA MESIAS ORLANDO**

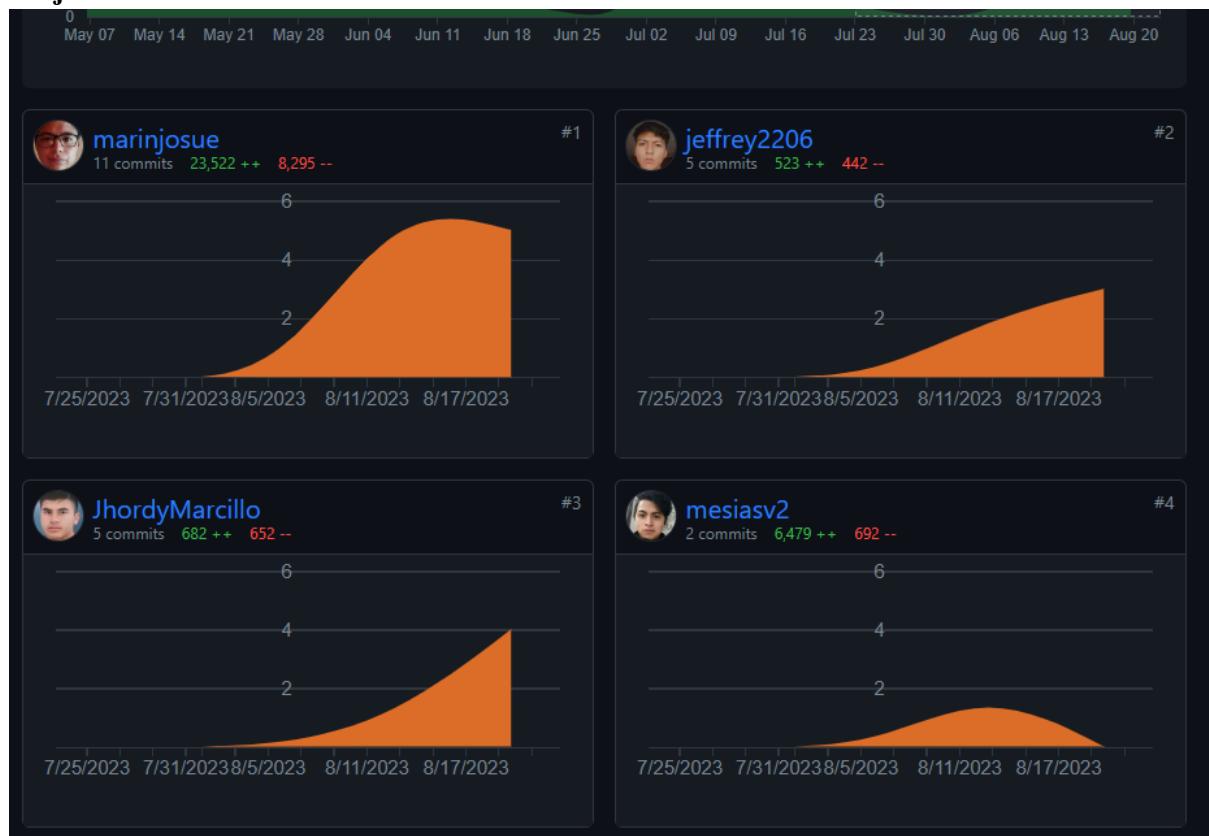
Rubric:

Project GitHub	/25 (Individual Contribution)
Project Review (Inspection)	19/25
Requirements and modeling (updated and design patterns)	3.5/5
No Dependencies	
Principles, CleanCode, Pitfalls	4/5
Functional Requirements user test (No CRUD operations)	5/5
Discount Product, Add to cart	
MongoDB	4/5
Design patterns code	2.5/5
Project Maintenance	20/25 (Time effort)
new user / user type	5/10
report screen	10/10
print out	5/5

Project Letter **15/25 (Letter of acceptance or review)**
missing contact information

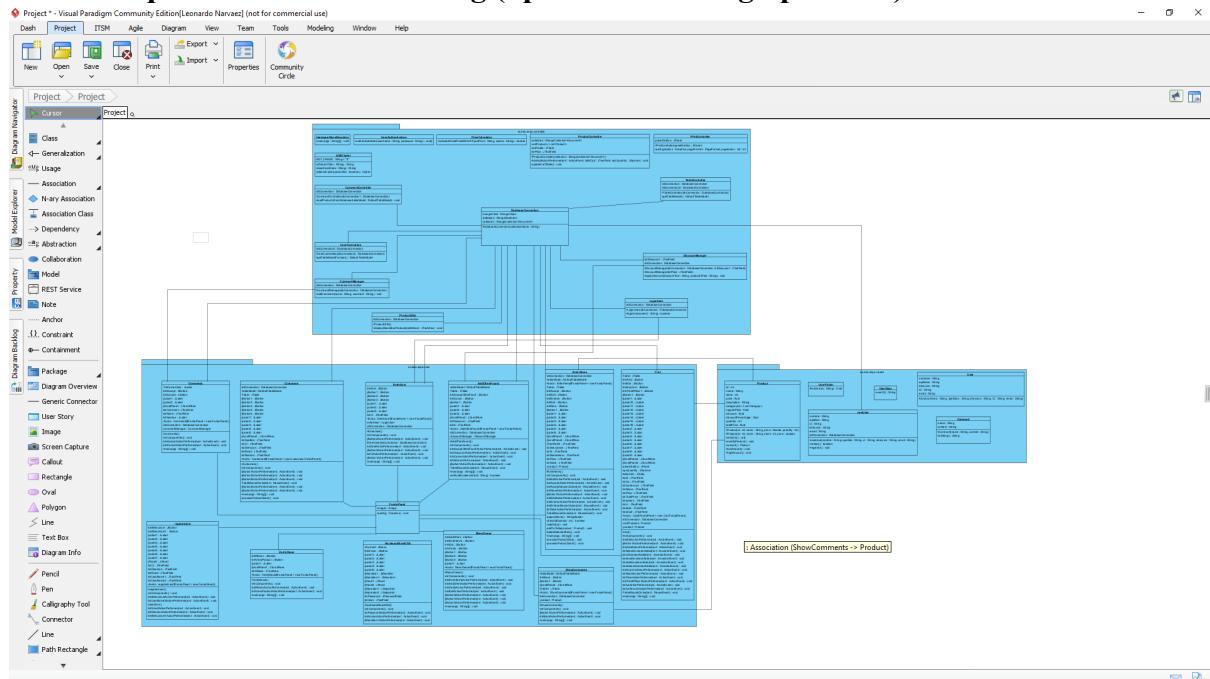
Evidence:

Project GitHub



Project Review (Inspection)

Requirements and modeling (updated and design patterns)



3.3 Functional requirements

OWNER

3.3.1 Functional requirement 1

User registration: The application should allow users to register and create an account in order to access all the functionalities of the application, such as viewing products, making purchases, managing their profile, among others.

3.3.2 Functional requirement 2

Product search: The application should allow owner to search for specific products using different search criteria, such as name, brand, Id, category, price, among others.

3.3.3 Functional requirement 3

Add products to shopping cart: The application must allow the owner to add products to the cart to generate a purchase invoice.

3.3.4 Functional requirement 4

Product offer: The owner of a hardware business requires a system that allows him to manage promotions and offers effectively. The software must offer the ability to create and schedule

special promotions, set percentage or dollar discounts, and define start and end dates for each promotion.

3.3.5 Functional requirement 5

User profile management: the application must allow the owner to manage a profile for customers where they will include the information to generate the invoice data.

3.3.6 Functional requirement 6

User profile management: The application should allow users to manage their profile, including updating their personal information, managing their purchase history.

3.3.7 Functional requirement 7

Add or remove a product to the catalog: The software should allow the owner to add a new product to the system, providing details such as name, description, category, price, and quantity in stock. If the product is available in inventory, the corresponding information must be recorded. In the event that there is no stock of the product, the system must allow the owner to record this situation and establish a notification to replenish the inventory. In addition, the owner must have the ability to remove a product from the catalog, when necessary, automatically updating the information related to the inventory.

3.3.8 Functional requirement 8

Modify product data: The system will have a function to enter the product catalog in the database. Certified users can add new information about each product, such as name, description, prices, and categories. Before allowing entry, you will be confirmed that the essential fields have been completed.

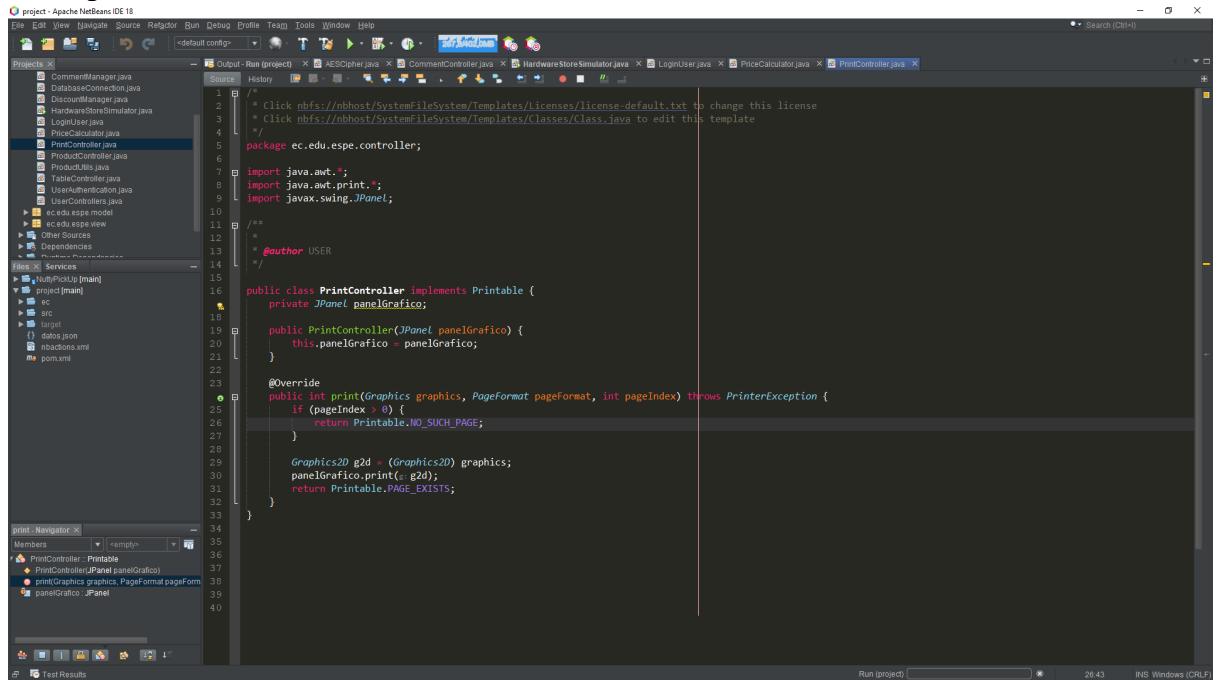
3.3.9 Functional requirement 9

Modify product data in the catalog: The system will allow the modification of the data of the existing products in the catalogue. Authorized users will be able to access and edit product information in a timely manner. Modifications can be made in fields such as name, description, price and category. The system will keep a record of all changes made, including the date and the user responsible for each change.

3.3.10 Functional requirements 10

You can add comments about the store in general to get feedback to continually improve the user experience. Users will be able to leave comments and opinions about the store, the products, the customer service, and any other relevant aspect. These comments will be visible to other users who visit the online store.

Principles,CleanCode, Pitfalls



```
/*
 * Click nbfs://nhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package ec.edu.espe.controller;

import java.awt.*;
import java.awt.print.*;
import javax.swing.JPanel;

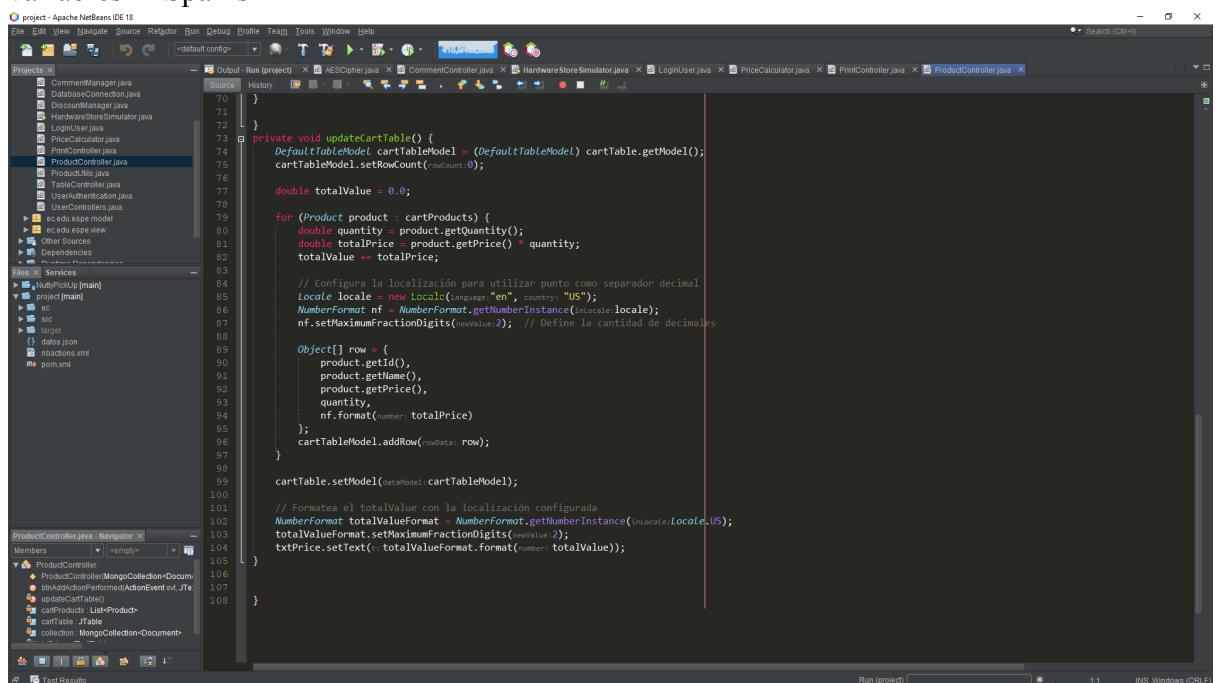
/*
 * @author USER
 */

public class PrintController implements Printable {
    private JPanel panelGrafico;

    public PrintController(JPanel panelGrafico) {
        this.panelGrafico = panelGrafico;
    }

    @Override
    public int print(Graphics graphics, PageFormat pageFormat, int pageIndex) throws PrinterException {
        if (pageIndex > 0) {
            return Printable.NO_SUCH_PAGE;
        }
        Graphics2D g2d = (Graphics2D) graphics;
        panelGrafico.print(g2d);
        return Printable.PAGE_EXISTS;
    }
}
```

Variables in Spanish



```

    }
}

private void updateCartTable() {
    DefaultTableModel cartableModel = (DefaultTableModel) cartTable.getModel();
    cartableModel.setRowCount(0);
    double totalValue = 0.0;

    for (Product product : cartProducts) {
        double quantity = product.getQuantity();
        double totalPrice = product.getPrice() * quantity;
        totalValue += totalPrice;
    }

    // Configura la localización para utilizar punto como separador decimal
    Locale locale = new Locale("en", "US");
    NumberFormat nf = NumberFormat.getNumberInstance(locale);
    nf.setMaximumFractionDigits(newValue);
    nf.setDecimalSeparator(true);

    Object[] row = {
        product.getId(),
        product.getName(),
        product.getPrice(),
        quantity,
        nf.format(totalPrice)
    };
    cartTableModel.addRow(row);
}

cartTable.setModel(cartableModel);

// Formatea el totalValue con la localización configurada
NumberFormat totalValueFormat = NumberFormat.getNumberInstance(locale);
totalValueFormat.setMaximumFractionDigits(newValue);
txtPrice.setText(totalValueFormat.format(totalValue));
}
}
```

Comments in spanish.

project - Apache NetBeans IDE 18

File Edit View Navigator Source Refactor Run Debug Profile Team Tools Window Help

Projects X Services

Source Design History

Cart.java

```
private void txtNameActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here.  
}  
  
private void txtPriceActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here.  
}  
  
private void btnTotalPriceActionPerfomed(java.awt.event.ActionEvent evt) {  
    String useIva = txtIva.getText();  
    double totalPriceWithVAT = PriceCalculator.calculateTotalPriceWithVAT(usePrice, useIva);  
    String formattedTotalPriceWithVAT = String.format("%.2f", args:totalPriceWithVAT);  
    txtTotalPrice.setText(formattedTotalPriceWithVAT);  
    double ivaAmount = totalPriceWithVAT - Double.parseDouble(args:usePrice);  
    Locale locale = new Locale(language:"en", country: "US");  
    NumberFormat nf = NumberFormat.getNumberInstance(locale);  
    nf.setMaximumFractionDigits(2);  
    String formattedIvaAmount = nf.format(args:ivaAmount);  
    txtIvaAmount.setText(formattedIvaAmount);  
}  
  
private void txtIvaActionPerformed(java.awt.event.ActionEvent evt) {  
}  
  
private void txtIvaAmountActionPerfomed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here.  
}  
  
private void TableMouseClicked(java.awt.event.MouseEvent evt) {  
    int fila = Table.getSelectedRow();  
    if (fila == -1) {  
        JOptionPane.showMessageDialog(parentComponent: null, message: "No se encontró ninguna fila seleccionada");  
    } else {  
        try {  
            int id = Integer.parseInt(Table.getValueAt(args:fila, column: 0).toString());  
            txtId.setText(String.valueOf(args:id));  
        } catch (Exception e) {  
            JOptionPane.showMessageDialog(parentComponent: null, message: "Error al obtener el ID de la fila seleccionada");  
        }  
    }  
}
```

Does not follow clean code standards.

project - Apache NetBeans IDE 18

File Edit View Navigator Source Refactor Run Debug Profile Team Tools Window Help

Projects X Services

Source Design History

Customers.java

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    int selectedRow = Table.getSelectedRow();  
    if (selectedRow == -1) {  
        JOptionPane.showMessageDialog(parentComponent: this, message: "Seleccione un articulo para eliminar", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);  
    } else {  
        String nom1 = (String) Table.getValueAt(args:selectedRow, column: 0);  
  
        int option = JOptionPane.showConfirmDialog(parentComponent: this, message: "Está seguro de eliminar el articulo seleccionado?", title: "Confirmar eliminación", optionType: JOptionPane.YES_NO_OPTION);  
  
        if (option == JOptionPane.YES_OPTION) {  
            collection.deleteOne(args:filters.eq("nombres", values:nom1));  
            tableModel.removeRow(args:selectedRow);  
            JOptionPane.showMessageDialog(parentComponent: this, message: "Articulo eliminado correctamente");  
        }  
    }  
}  
  
private void jButtonActionPerfomed(java.awt.event.ActionEvent evt) {  
    processProductData();  
}  
  
/* @param args the command line arguments */  
public static void main(String args[]) {  
    /* Set the Nimbus look and feel */  
    LookAndFeel.setNimbusLookAndFeel();  
  
    /* Create and display the form */  
    java.awt.EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            new Customers().setVisible(true);  
        }  
    });  
}
```

A lot of code in many buttons

The screenshot shows the Apache NetBeans IDE interface with the following details:

- Title Bar:** project - Apache NetBeans IDE 18
- Menu Bar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help
- Toolbar:** Standard icons for Save, Open, Run, Stop, etc.
- Project Explorer:** Shows a project named "project" with several source packages and files like AESCipher.java, CommandController.java, DatabaseConnection.java, DiscountManager.java, HardwareStoreSimulator.java, LoginUser.java, MongoDBController.java, PrintController.java, ProductController.java, ProductUtils.java, TableController.java, UserControllers.java, and NutriPickUp.java.
- Code Editor:** Displays the `ProductController.java` file with the following code snippet (lines 25-67):

```

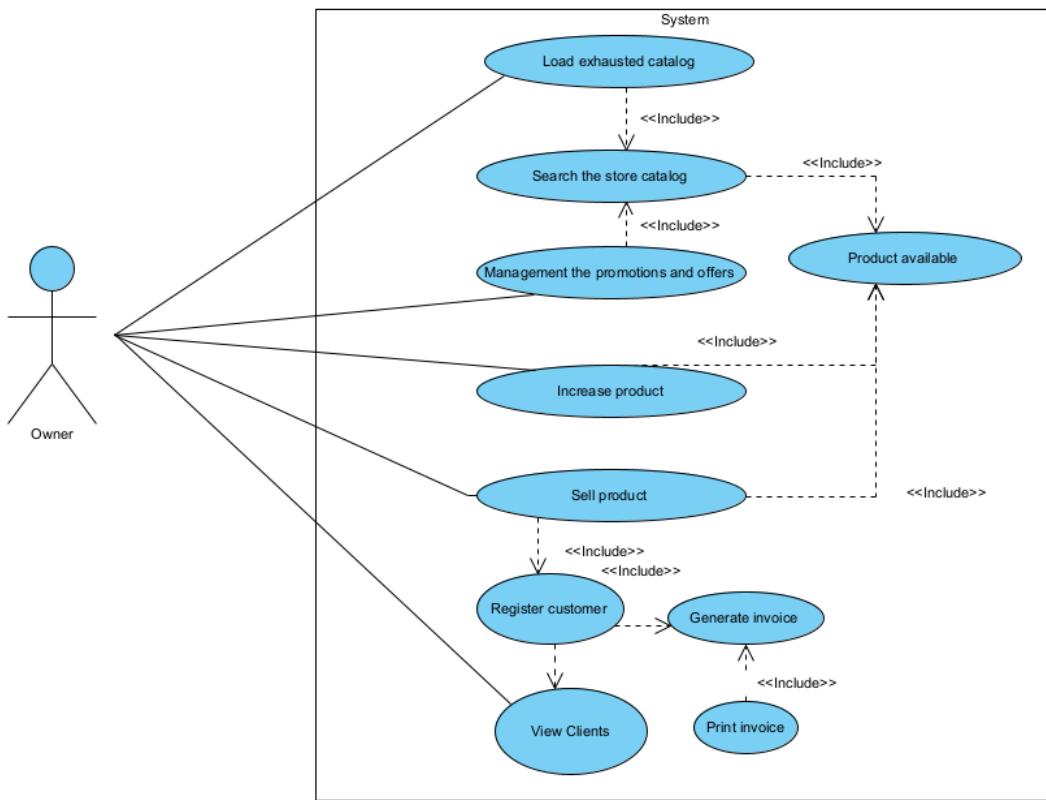
    this.cartProducts = cartProducts;
    this.cartTable = cartTable;
    this.txtPrice = txtPrice;
}

public void btnAddActionPerformed(java.awt.event.ActionEvent evt, JTextField txtIdCart, JSpinner spnQuantity) {
    int id = Integer.parseInt(txtIdCart.getText());
    Document document = collection.find(new Document(key: "id", value: id)).first();
    if (document != null) {
        String name = document.getString(key: "name");
        Double price = document.getDouble(key: "price");
        int stock = document.getInteger(key: "stock");

        if (stock > 0) {
            int quantity = (int) spnQuantity.getValue();
            if (quantity > 0 && quantity <= stock) {
                int updatedStock = stock - quantity;
                document.put(key: "stock", value: updatedStock);
                collection.replaceOne(new Document(key: "id", value: id), tdi, document);
            }
            boolean found = false;
            for (Product product : cartProducts) {
                if (product.getId() == id) {
                    product.setQuantity(product.getQuantity() + quantity);
                    product.setTotalPrice((float) (product.getTotalPrice() + (price * quantity)));
                    found = true;
                    break;
                }
            }
            if (!found) {
                Product product = new Product(id, name, price, quantity);
                cartProducts.add(product);
            }
        } else {
            JOptionPane.showMessageDialog(parentComponent: null, message: "La cantidad ingresada debe ser mayor que 0 y no exceder el stock disponible", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(parentComponent: null, message: "El producto seleccionado no se encuentra disponible", title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

```
- Toolbars:** Standard Java development toolbars.
- Status Bar:** 59 17 INS Windows (CRLF)

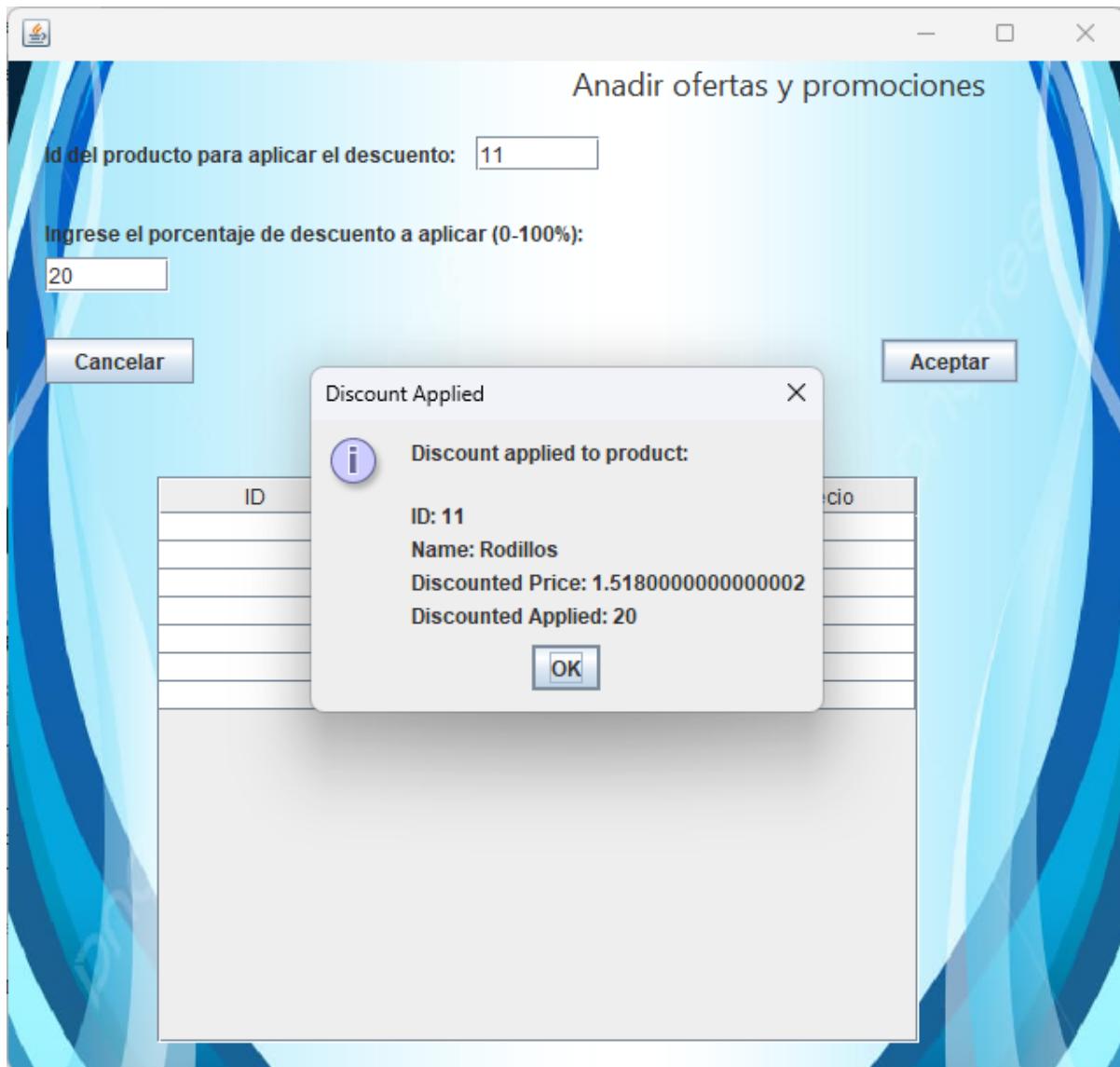
Some variables like spnQuantity could be more descriptive.
Use case diagram



Functional Requirements user test (No CRUD operations)

3.3.4 Functional requirement 4

Product offer: The owner of a hardware business requires a system that allows him to manage promotions and offers effectively. The software must offer the ability to create and schedule special promotions, set percentage or dollar discounts, and define start and end dates for each promotion.



MongoDB

```
_id: ObjectId('64b03a4751d68d7b8639b811')
id: 11
name: "Rodillos"
stock: 0
price: 1.8975000000000002
description: "Producto de trabajo de Pintura"
regularPrice: 0
discount: 50
discountPercentage: 45
```

3.3.3 Functional requirement 3

Add products to shopping cart: The application must allow the owner to add products to the cart to generate a purchase invoice.

The screenshot shows a Microsoft Word document titled "OPP Project (Team CodeCrafters) Requirements Update.docx". The content includes functional requirements 3.3.1 through 3.3.4, which describe the ability for the store owner to add products to a shopping cart. Below the requirements is a screenshot of a software interface titled "FERRETERIA_DSA Factura". The interface shows a catalog table with columns ID, Nombre, Stock, Precio, and Description. A shopping cart table shows a single item: "pernos" with ID 15, Nombre "pernos", Precio unitario 0.18, Cantidad 1.0, and Precio total 0.18. The software also displays fields for Nombre, Fecha y Hora, Dirección, Email, Iva %, Monto, and Total.

The screenshot shows a web browser displaying a purchase invoice from "FERRETERIA_DSA". The invoice details are: Nombre: Leonardo Narvaez, Fecha y Hora: 24/08/2023 07:56:28, Dirección: Valle de los chillos, Email: sapo@gmail.com. The items purchased are pernos (ID 15), with a Precio unitario of 0.18, Cantidad of 2.0, and Precio total of 0.36. The invoice also shows a Total of \$ 0.36, Iva % of 12, Monto of 0.04, and a final Total of \$ 0.40.

MongoDB

The screenshot shows a MongoDB interface with a document in the collection "pernos". The document contains the following fields and values:

```

_id: ObjectId('64e44d34aa19c0140917758c')
id: 15
name: "pernos"
stock: 29
price: 0.18
description: "calvos de metal galvanizado"
regularPrice: 0
discount: 0
discountPercentage: 0
quantity: 0
totalPrice: 0
    
```

Design patterns code

Project Maintenance Project Letter

Team 5: The FAMSE Leader: Anthony Morales

Inspector: Team 6 Code Warriors **Leader:** Edison Verdesoto

Project: NPU System

GitHub: https://github.com/LeoNarvaez2503/The_FAMSE.git

Compass:

mongodb+srv://anthonymorales:anthonymorales@cluster0.nngqbpt.mongodb.net/

Grade: PR: /25, PM: /25, Letter: /25 → 54 /75

21. MORALES CARVAJAL ANTHONY ALAIN

22. MORAN PALOMO LENIN DAVID

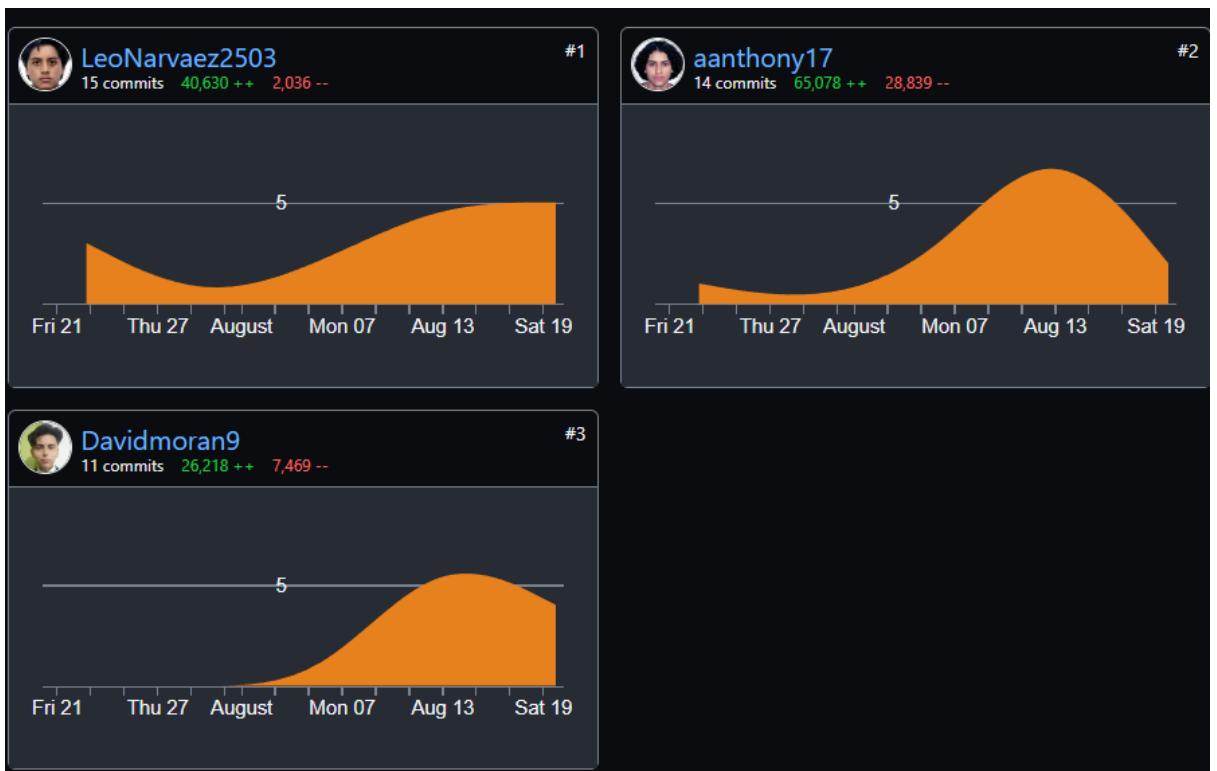
23. NARVAEZ CRIOLLO LEONARDO VINICIO

Rubric:

Project Review (Inspection)	16/25
Requirements and modeling (updated and design patterns)	4.5/5
Principles,CleanCode, Pitfalls	2.5/5
troubles: variables name, no indented, no Single responsibility, no vertical alignment	
Functional Requirements user test (No CRUD operations)	5/5
Sales proccesing , report generator	
MongoDB	4/5
Missing Login in the BD and Users in the code.	
Design patterns code	0/5
No pattern design	
Project Maintenance	22/25 (Time effort)
new user / user type	8/10
report screen	9/10
print out	5/5
Project GitHub	25/25 (Individual Contribution)
Project Letter	15/25 (Letter of acceptance or
review)	missing contact information

Evidence:

Project GitHub



Project Review (Inspection)

Requirements and modeling (updated and design patterns)

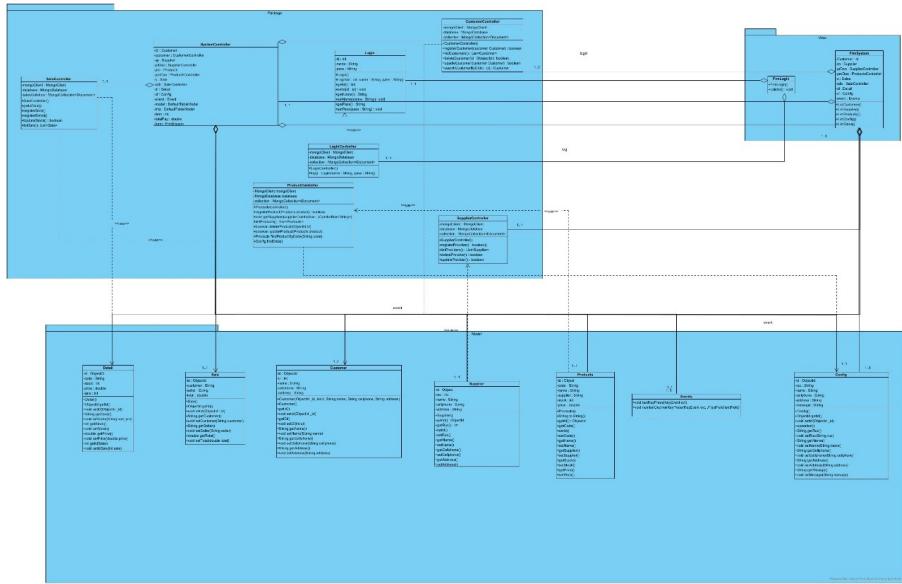
Requirement Number Name Requirement Type Requirement Priority	R1		
	Product management		
		Requirement	Restriction
		High/Essential	Medium/Desired

Requirement Number Name Requirement Type Requirement Priority	R2		
	Price control		
		Requirement	Restriction
		High/Essential	Medium/Desired

Requirement Number Name Requirement Type Requirement Priority	R3		
	Stock management		
		Requirement	Restriction
		High/Essential	Medium/Desired

Requirement Number Name Requirement Type Requirement Priority	R4		
	Sales processing		
		Requirement	Restriction
		High/Essential	Medium/Desired

Requirement Number Name Requirement Type Requirement Priority	R5		
	Report generator		
		Requirement	Restriction
		High/Essential	Medium/Desired



Principles, CleanCode, Pitfalls

```

Object[] ob = new Object[6];
for(int i = 0; i < ListarCl.size(); i++){
    ob[0] = ListarCl.get(index: i).getId();
    ob[1] = ListarCl.get(index: i).getCi();
    ob[2] = ListarCl.get(index: i).getName();
    ob[3] = ListarCl.get(index: i).getCellphone();
    ob[4] = ListarCl.get(index: i).getAddress();
    modelo.addRow(rowData: ob);
}
TableCustomer.setModel(dataModel:modelo);
}

public class FrmSystem extends javax.swing.JFrame {
    Customer cl = new Customer();
    CustomerController customer = CustomerController.getInstance();
    Supplier sp = new Supplier();
    SupplierController prDao = SupplierController.getInstance();
    Products pro = new Products();
    ProductsController proDao = ProductsController.getInstance();
    Sale s = new Sale();
    SaleController sdb = SaleController.getInstance();
    Detail di = new Detail();
    Config cf = new Config();
    Events event = new Events();
    DefaultTableModel modelo = new DefaultTableModel();
    DefaultTableModel tmp = new DefaultTableModel();
    SystemController controller = new SystemController(form: this);
    ...

    public boolean deleteCustomer(ObjectId id) {
        try {
            Bson filter = Filters.eq(fieldName: "_id", value: id);
            DeleteResult result = collection.deleteOne(bson: filter);
            return result.getDeletedCount() > 0;
        } catch (MongoException e) {
            System.out.println(e.toString());
            return false;
        }
    }
}

```

```

public boolean deleteCustomer(ObjectId id) {
    try {
        BSON filter = Filters.eq("fieldName", value: id);
        DeleteResult result = collection.deleteOne(bson: filter);
        return result.getDeletedCount() > 0;
    } catch (MongoException e) {
        System.out.println(e.toString());
        return false;
    }
}

public boolean updateCustomer(Customer customer) {
    try {
        Document query = new Document(key: "_id", value: customer.getId());
        Document update = new Document(key: "$set", value: new Document(key: "ci", value: customer.getCi())
            .append(key: "name", value: customer.getName())
            .append(key: "cellphone", value: customer.getCellphone())
            .append(key: "address", value: customer.getAddress()));
        collection.updateOne(bson: query, bson: update);

        return true;
    } catch (MongoException e) {
        System.out.println(e.toString());
        return false;
    }
}

public Customer searchCustomerByCI(int ci) {
    Document query = new Document(key: "ci", value: ci);
    Document result = collection.find(bson: query).first();

    Customer customer = new Customer();
    if (result != null) {

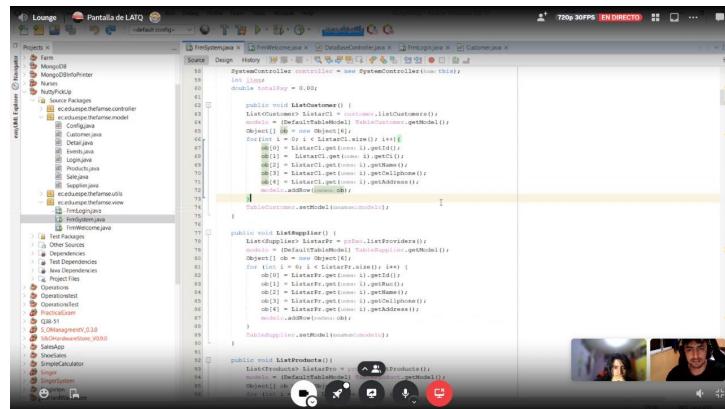
```

```

        public Customer(ObjectId _id,
                     this._id = _id;

```

meet evidence



Functional Requirements user test (No CRUD operations)



Ruc: 1726063223
 Nombre: Laura Jacome
 Telefono: 0888888
 Dirección: Quitumbe
 Razon: Tienda De Frutos Secos

Factura:64e74bc70352
 9c096f7954e0
 Fecha: 24-23-2023

Datos de los Clientes

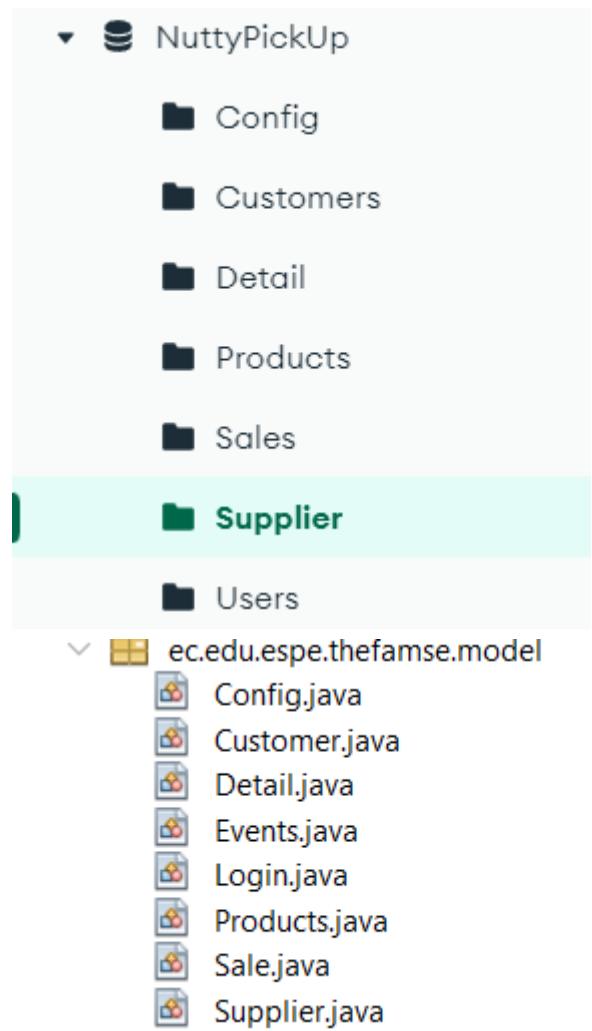
Ci/Ruc:	Nombre:	Teléfono:	Dirección:
501365845	Leonel Tipan	0992314893	Centro Historico
Cantidad:	Descripción:	Precio U:	Precio T:
5	Nuez	5.0	5.0
4	Peras	9.8	9.8
5	Pistacho	7.0	7.0

Total a Pagar: 21.8

Cancelación y Firma

 Gracias por su Compra

MongoDB



Design patterns code

Project Maintenance
Project Letter

n

Team 6: Code Warriors Leader: Edison Verdesoto

**24. QUISHPE LLUMIQUINGA JOHN MATEO
25. TIPAN QUINTO LEONEL ALEJANDRO
26. VERDESOTO SEGOVIA EDISON DAMIAN**

**Inspector: Team 1 Jsons Leader: Yeshua Amador
Project: S&O HardwareStore
GitHub: https://github.com/EDVerdesoto/T06_SandO-System.git**

MongoDB Compass:

mongodb+srv://latipan06:latipan06@cluster0.kawghe4.mongodb.net/?retryWrites=true&w=majority

Grade: PR: /25, PM: /25, Letter: /25 → 44 /75

Rubric:

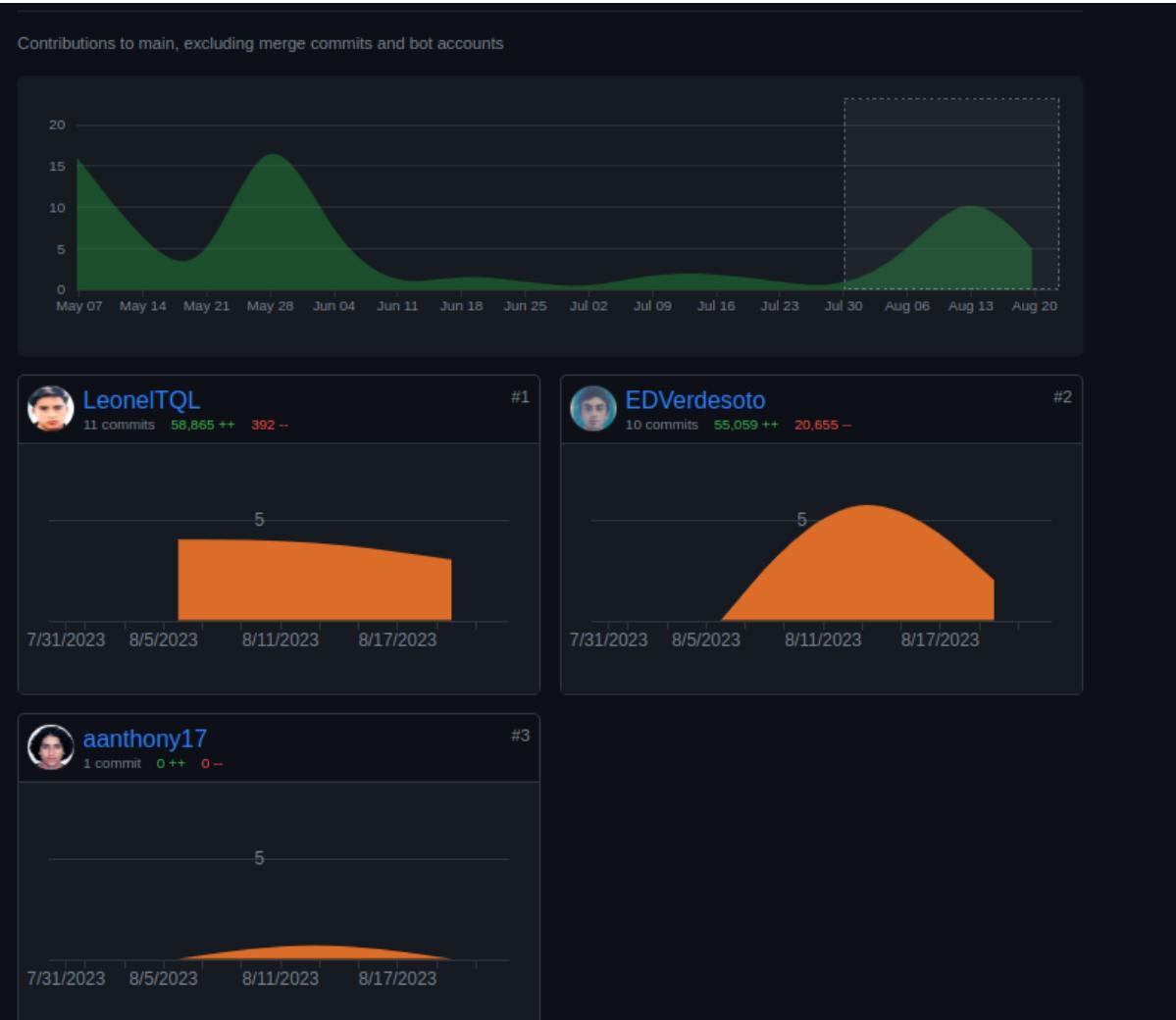
Project Review (Inspection)	17.5/25
Requirements and modeling (updated and design patterns)	4/5
Principles, CleanCode, Pitfalls	4/5
Functional Requirements user test (No CRUD operations)	4.5/5
MongoDB	5/5
Design patterns code	0/5
Project Maintenance	10/25 (Time effort)
new user / user type	0/10
report screen	5/10
print out	5/5

Project GitHub **/25 (Individual Contribution)**

Project Letter **15/25 (Letter of acceptance or review)**
missing contact information

Evidence:

Project GitHub

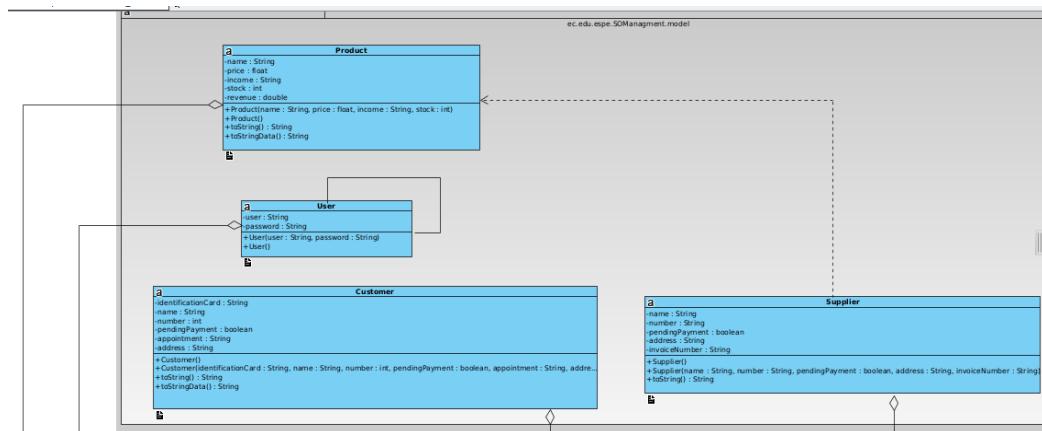


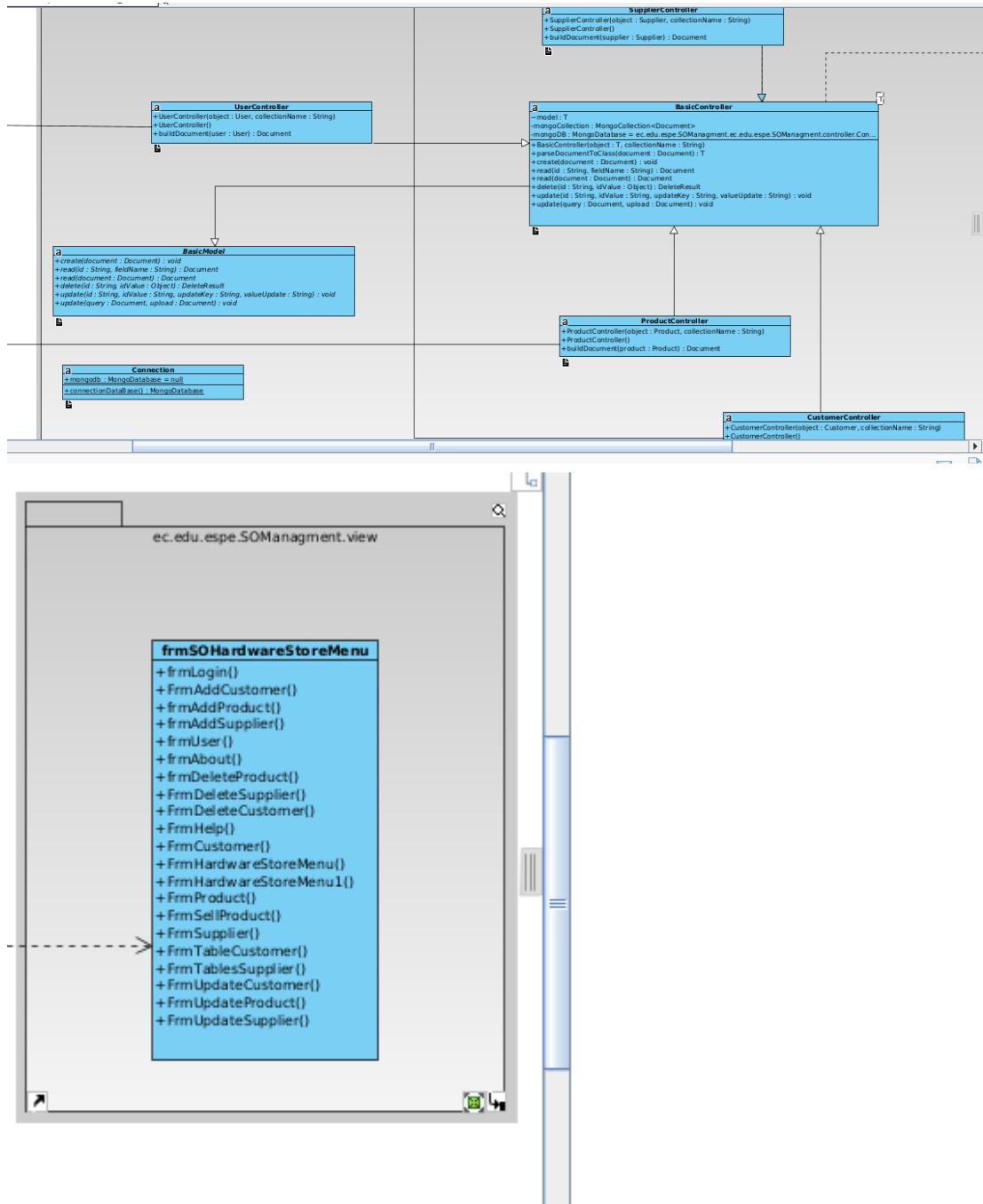
Project Review (Inspection) Requirements and modeling (updated and design patterns)

3.2. Functions

The software shall have the following functions:

- **Inventory management:**
- Add new inventory items: The user shall be able to add new items to the inventory database.
- Delete inventory items: The user shall be able to delete items for the inventory database.
- Update inventory items: The user shall be able to update the information of existing items in the inventory database.
- View inventory status: The user shall be able to view the current status of the inventory, including the quantity in stock and the location of the inventory.
- Calculate taxes and profits: The software shall automatically calculate the IVA and 30% of the products to determine the store's profit.
- User management: The program must allow more than one user.
- Sell Products: Update the BD by selling products but don't show the bill.





```

/*
 * @author Edison Verdesoto, Code Warriors, DCCO-ESPE
 */
public class Decimals {
    public static Double roundToTwoTenths(Double amountToRound) {
        Double amountRounded;
        int dollars = (int) (amountToRound/1) ;
        Double cents = (amountToRound%1)*100;

        if(cents == 0){
            amountRounded = amountToRound;
        }
        else {
            cents = (double) Math.round(a: cents);
            amountRounded = dollars + (cents/100);
        }

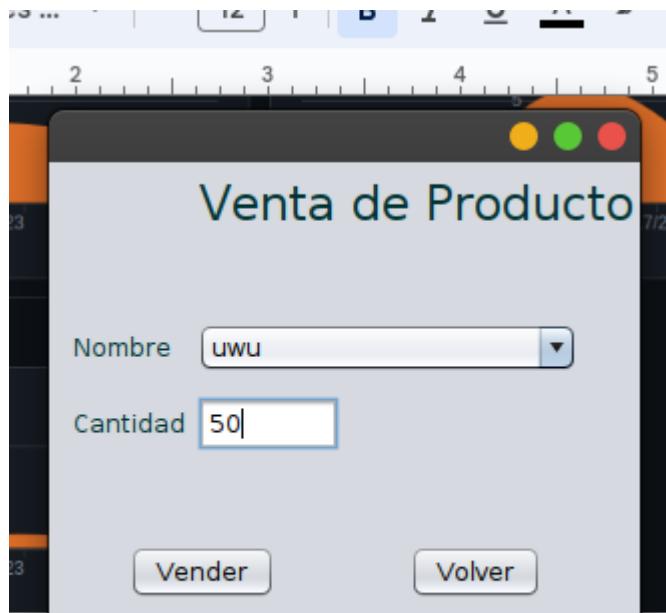
        return amountRounded;
    }
}

```

Principles,CleanCode, Pitfalls

```
public static boolean validateID(String identificationCard) {  
    int suma = 0;  
    if (identificationCard.length() != 10) {  
        return false;  
    }  
  
    for (int i = 0; i < 9; i++) {  
        int digit = Integer.parseInt(identificationCard.charAt(index: i) + "");  
        if (i % 2 == 0) {  
            digit = digit * 2;  
            if (digit > 9) {  
                digit -= 9;  
            }  
  
            String identificationCard;  
String name;  
int number;  
boolean pendingPayment;  
boolean validateId = true;  
String appointment;  
String address;  
  
identificationCard = txtId.getText();  
name = txtName.getText();  
number = Integer.parseInt(s: txtNumber.getText());  
pendingPayment = false;  
appointment = formatDate.format(date:txtDateService.getDate());  
address = txtAddress.getText();  
  
validateId = IdentificationCardController.validateID(identificationCard);
```

Functional Requirements user test (No CRUD operations)



nd modeling (updated and design patterns)



–Second functional requirement: PVP with IVA and 30% of the initial price

— □ ×

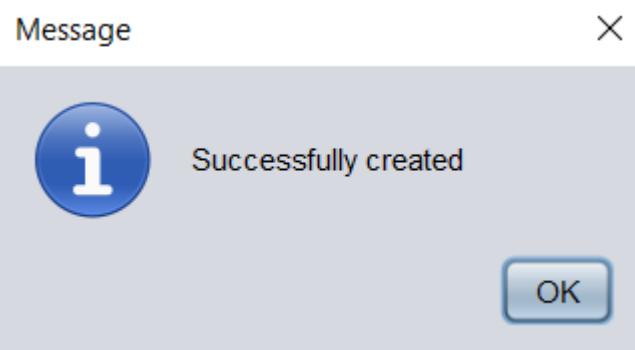
Producto

Nombre Tijeras

Fecha de ingreso 24 ago 2023

Precio 15

Stock 10



Productos

Nombre	Fecha de ingreso	Costo	Stock	Precio de venta
Brochas lizas	11-07-2023	5.0	29	6.5
Serrucho	16-07-2023	30.0	20	39.0
Pintura en spray	14-07-2023	20.0	17	26.0
Cenemto de cont...	17-07-2023	4.0	64	5.2
Tornillo en cruz	12-08-2023	14.0	18	18.2
Tornillo plano	25-07-2023	50.0	89	65.0
Destornillador	25-07-2023	20.5	83	26.65
Martillo	25-07-2023	20.5	89	26.65
Cinta de embalaje	25-07-2023	20.5	23	26.65
pintura	11-08-2023	23.0	3	29.9
Canastas	22-08-2023	20.0	20	26.0
uwu	19-08-2023	1.0	0	1.3
Tijeras	24-08-2023	15.0	10	19.5

Creating Users Modified:

MongoDB Compass - cluster0.kawgh4.mongodb.net/S&OHardWareStore.customers

Connect Edit View Collection Help

cluster0.kawgh... S&OHardWareSt...

My Queries Databases +

Search

- Movies
 - TicketsSold
- S&OHardWareStore
 - customers
 - products
 - suppliers
 - users
- Singers
- admin
- guitars
- local
- sample_airbnb
- sample_analytics
- sample_geospatial
- sample_guides
- sample_mflix

MongoDB - cluster0.kawgh4.mongodb.net/

Filter Type a query: { field: 'value' }

1-5 of 5 Find Options

S&OHardWareStore.customers

4 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' }

1-5 of 5 Find Options

S&OHardWareStore.products

11 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' }

1-12 of 12 Find Options

MongoDB

5/5

MongoDB

cluster0.kawgh... S&OHardWareSt...

My Queries Databases +

Search

- Movies
 - TicketsSold
- S&OHardWareStore
 - customers
 - products
 - suppliers
 - users
- Singers
- admin
- guitars
- local
- sample_airbnb
- sample_analytics
- sample_geospatial
- sample_guides
- sample_mflix

MongoDB - cluster0.kawgh4.mongodb.net/

Filter Type a query: { field: 'value' }

1-12 of 12 Find Options

S&OHardWareStore.products

11 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' }

1-12 of 12 Find Options

-Second functional requirement updated in database

```
_id: ObjectId('64e7525af195e276e75953ed')
name: "Tijeras"
price: 15
income: "24-08-2023"
stock: 10
PVP: 19.5
```

-Third functional requirement updated in database
CRUD modifications in MongoDB

Design patterns code 0/5

Project Maintenance
Project Letter