



“UNIVERSIDAD DE LAS FUERZAS ARMADAS”

ESPE

SOFTWARE ENGINEERING

OBJECT - ORIENTED PROGRAMMING

TOPIC:

WORKSHOP 30 - Project to review

TEACHER:

PHD. EDISON LASCANO

NRC:

4680

DATE:

July 25, 2022

SEDE MATRIZ SANGOLQUÍ

QUITO-ECUADOR

2022

EDISON LASCANO

Rubric:

| | |
|-----------------------|-----|
| 1. Requirements | /10 |
| 2. UML Class Diagram | /10 |
| 3. Clean Code | /10 |
| 4. Db (MongoDb Atlas) | /10 |
| 5. GUI | /10 |
| 6. Tables | /10 |
| 7. Forms | /30 |
| 8. Unit tests | /10 |

| | |
|---------|-------|
| TOTAL | /100 |
| GitHub | /100% |
| Average | /100 |

Proof:

Requirements (Features/items)

- UML Class Diagram
 - Classes -2
 - Attributes -2
 - Methods -2
 - Relations -2
 - Abstract Classes/Interfaces 2
 - Abstract methods 2
 - Polymorphism (overridden methods) 2
 - Inheritance 2
 - MVC -> packages model (POJO classes), view(GUI), controller (Abstract C./Concrete C./Interfaces) 2
 - (getters and setters 9(-.01), labels(-0.1), panels(-0.1), no dependencies in the view package (0.1))

Clean Code

Pitfalls/Code Smells (0.1)

DB (MongoDb Atlas)

| | |
|--|------|
| Database Name | -0.5 |
| Collections vs Model classes | -0.5 |
| attributes (collections) vs attributes (classes) | -0.5 |

GUI

Splash Window

| | | | |
|--------------------|---|------------------|-------------------|
| Authentication | 2 | (it should work) | |
| Application (Menu) | 2 | (requirements) | (-0.2) |
| Forms | 6 | (requirements) | Navigability -0.2 |

Tables:

| | |
|-----------------------|---|
| data come from DB | 5 |
| print data to printer | 5 |

Forms

at least, there must be 6 different forms, every form is worth 5 points

- Login Form
- 5 Forms for Information (CRUD operations, 4 business rules)
- Every form
 - Every input (JTextField) is validated
 - There are different types of inputs (not everything is a JTextField) -1 for each JTextField
 - Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

every mistake: -1

less than 5 forms -> /50%

Unit Test

At least 100 unit tests.

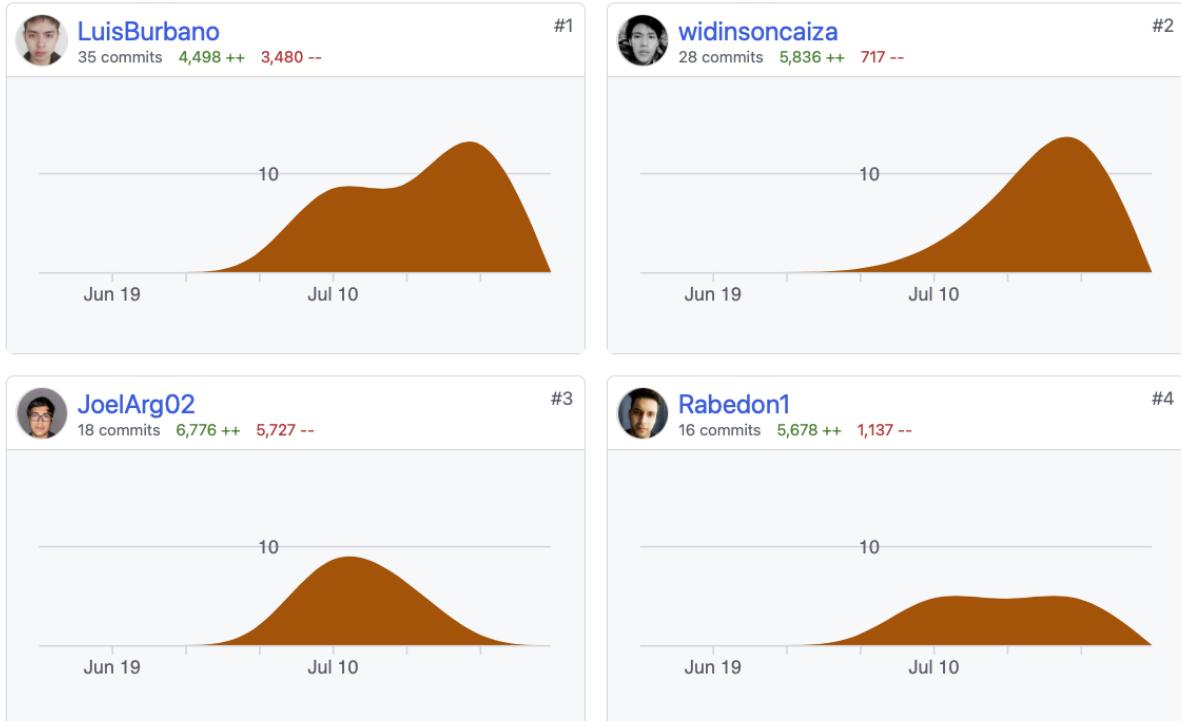
- review that the unit tests are not empty
- there must be at least one unit test that is failing.
- If all unit tests are OK, then it is half the grade. It means that the tests are not good enough
- every unit test is worth 0.1

TEAM 01 BETTACODERS INSPECTOR: TEAM 02 (Mateo Condor

) GitHub: <https://github.com/LuisBurbano/SMC-styles-irelia>

Project: SMC-styles-irelia

1. Arguello Espinosa Joel Daniel 90%
2. Bedon Guevara Roberto Alexander 80%
3. Burbano Pacheco Luis Ariel 100%
4. Caiza Pullotasig Widinson Danilo 100%



Rubric:

| | |
|------------------------|--------|
| 1. Requirements | 9/10 |
| 2. UML | 8.6/10 |
| 3. Clean Code | 9.4/10 |
| 4. DB (Mongo DB Atlas) | 9.5/10 |
| 5. GUI | 8.6/10 |
| 6. Tables | 5/10 |
| 7. Forms | 25/30 |
| 8. Unit tests | 0.8/10 |

| | |
|-------|----------|
| TOTAL | 75.9/100 |
|-------|----------|

Proof:

1. Requirements (Features/items)
 - Shifts to customer
 - List of services
 - Payment
 - Show the information tables of suppliers, products, customers
 - Register customer
 - Validate your login

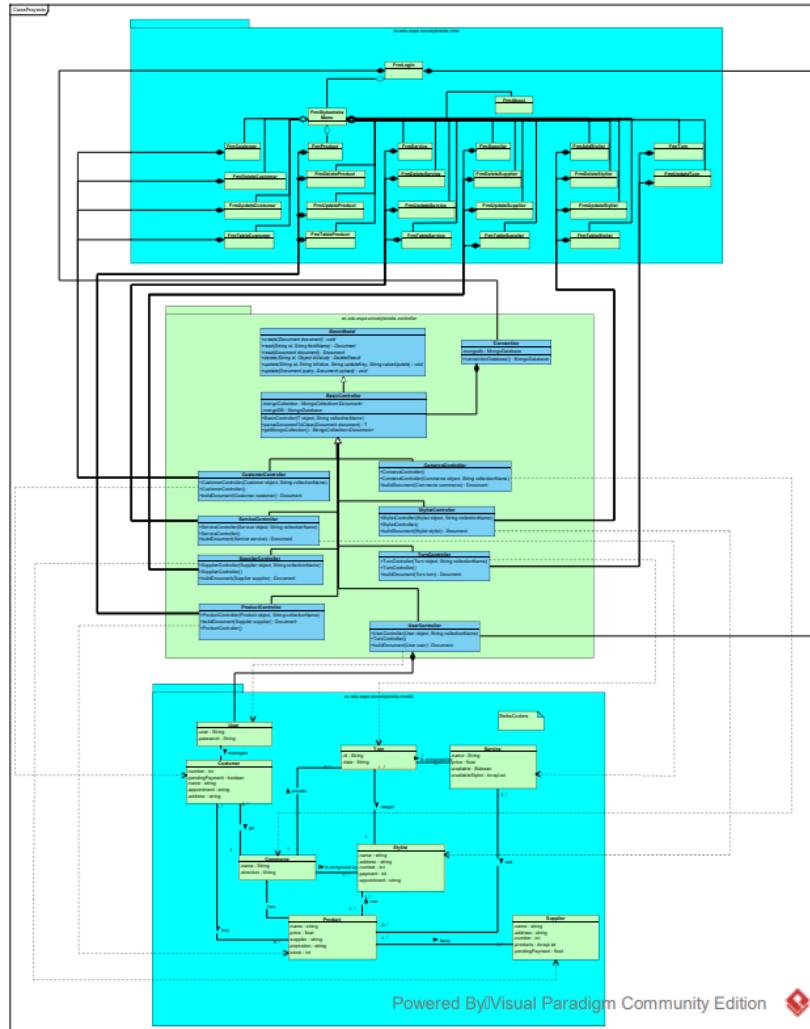
REQUIREMENTS

The requirements that the client needs to improve their hairdressing service are the following:

1. The system shall assign shifts to customers
2. The system shall give List of services, prices the most contracted
3. The system shall give the payment for each stylist.
4. The system shall give Stock of products, when they expire, the most used, my suppliers
5. The system shall register customers

2. UML Class Diagram

- Classes -2
- Attributes -2/-0.5
- Methods -2/-0.5
- Relations -2
- Abstract Classes/Interfaces 2/2
- Abstract methods 2/2
- Polymorphism (overridden methods) 2/2
- Inheritance 2/2
- MVC -> packages model (POJO classes), view(GUI), controller (Abstract C./Concrete C./Interfaces) 1.9/2
 - (getters and setters 9(-.01), labels(-0.1), panels(-0.1), no dependencies in the view package (0.1)



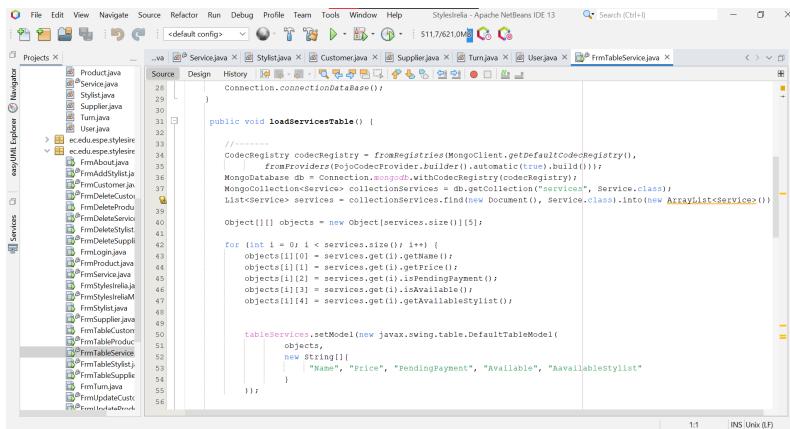
Clean Code

Pitfalls/Code Smells (0.1)
frame sin usar -0.1

```

Projects X Services X Files X Start Page X FrmServices.java X FrmMytestrela X FrmMytestrelaMen.java X
Source Design History Help
13 [ ] /* Create new frmMytestrela
14 [ ] */
15 [ ] public FrmMytestrela() {
16 [ ]     initComponents();
17 [ ] }
18 [ ]
19 [ ] /**
20 [ ] * This method is called from within the constructor to initialize the form.
21 [ ] * WARNING: Do NOT modify this code. The content of this method is always
22 [ ] * regenerated by the Form Editor.
23 [ ] */
24 [ ] @SuppressWarnings("unchecked")
25 [ ] // Generated using Eclipse IDE.
26 [ ] private void initComponents() {
27 [ ]     /* * Open args the command line arguments
28 [ ] */
29 [ ]     public static void main(String args[]) {
30 [ ]         /* Set the Nimbus look and feel */
31 [ ]         // Look and feel setting code optional */
32 [ ]         /* Create and display the form */
33 [ ]         java.awt.EventQueue.invokeLater(new Runnable() {
34 [ ]             public void run() {
35 [ ]                 new FrmMytestrela().setVisible(true);
36 [ ]             }
37 [ ]         }, 117);
38 [ ]     }
39 [ ] }
40 [ ]
41 [ ] 
```

metodo en el view y no en el controller -0.1



```

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

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

private void txtStockActionPerformed(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:
}

```

```

    }
217 }
218
219 private void txtAddressActionPerformed(java.awt.event.ActionEvent evt) {
220     // TODO add your handling code here:
221 }
222

```

Variable no usada

```

String name;
int number;
boolean pendingPayment; Variable pendingPayment is not used
String appointment;

```

Mala declaración

```

2 Customer customer = new Customer(identificationCard, name, number, false, appointment, address);
3
4 customerController.read(identificationCard, "identificationCard");
5
6
249 private void txtNameActionPerformed(java.awt.event.ActionEvent evt) {
250     // TODO add your handling code here:
251 }
252
253 private void txtIdActionPerformed(java.awt.event.ActionEvent evt) {
254     // TODO add your handling code here:
255 }
256

```

Comentarios

```

36
37
38 userController = new UserController();
39 setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
40 String path = System.getProperty("user.dir");
41 //    ImageIcon img = new ImageIcon(path + "\\src\\ec\\edu\\espe\\stylesirelia\\sources\\logoBettaCoders.png");
42 //    this.setIconImage(img.getImage());
43 this.setTitle("Login Window");
44

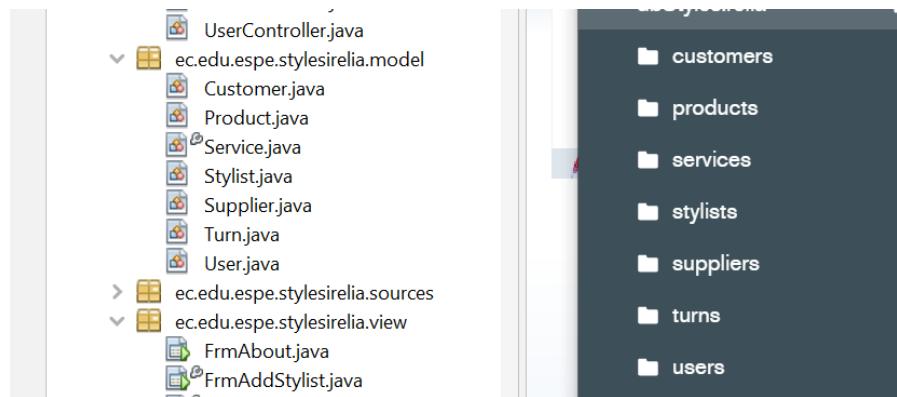
```

```

private void extNumberKeyTyped(java.awt.event.KeyEvent evt) {
    String value = txtNumber.getText();
    int length = value.length(); //saca la longitud del string
    if (evt.getKeyChar() >= '0' && evt.getKeyChar() <= '9') {
        txtNumber.setEditable(true);
        if (length >= 10) {
    
```

DB (MOngoDb Atlas)

| | |
|--|------|
| Database Name | -0.5 |
| Collections vs Model classes | -0.5 |
| attributes (collections) vs attributes (classes) | -0.5 |

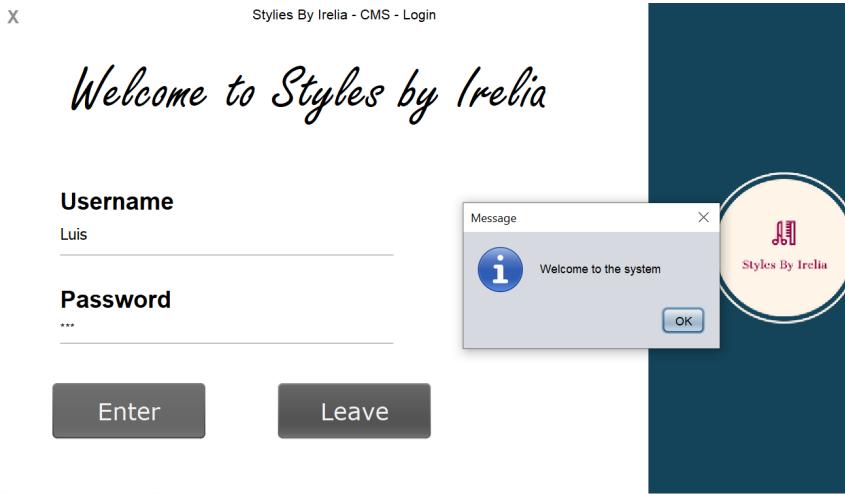


GUI

| | |
|--------------------|--------------------------------------|
| Splash Window | |
| Authentication | 2/2 (it should work) |
| Application (Menu) | 1.6/2 (requirements) |
| Forms | 5/6 (requirements) Navigability -0.2 |

The image shows a dual-pane interface. On the left is a login form titled 'Welcome to Styles by'. It has fields for 'Username' (Luis) and 'Password' (redacted). Below the fields are 'Enter' and 'Leave' buttons. On the right is a MongoDB browser for the 'dbStylesirelia' database, specifically the 'users' collection. It lists two documents:

- One document for user 'Alex' with id '62d946dc8ee631f69b250815', user 'Alex', and password '123'.
- One document for user 'Luis' with id '62d946dc8ee631f69b250816', user 'Luis', and password '123'.



Menú Help no hace nada

Tables:

data come from DB
print data to printer

5/5

0/5

| ID | Name | Number | Payment | Appointment | Address |
|-------------------|------------|--------|------------|-------------|---------|
| Peluquero ... | 0983272-37 | 0.0 | | | |
| Peluquero ... | 0983272-37 | 0.0 | | | |
| Peluquero ... | 0983272-37 | 0.0 | | | |
| Peluquero ... | 0983272-37 | 0.0 | | | |
| 1720876534 Melina | 0983265423 | 4656.0 | 15-07-2022 | Mi corazon | |
| 1720985533 Luis | 0983245621 | 600.0 | 23-07-2022 | Quito | |

Load Back

dbStylesIrelia.stylists

```
_id: ObjectId('62dfe4993eac671db1ed61')
identificationCard: "1720876534"
name: "Melina"
number: "0983265423"
payment: 4656
appointment: "15-07-2022"
address: "Mi corazon"

_id: ObjectId('62dfe66e1b19385cc389a4bc')
identificationCard: "1720985533"
name: "Luis"
number: "0983245621"
payment: 600
appointment: "23-07-2022"
address: "Quito"
```

Forms

at least, there must be 6 different forms, every form is worth 5 points

- Login Form
- 5 Forms for Information (CRUD operations, 4 business rules)

Every form

- Every input (JTextField) is validated
- There are different types of inputs (not everything is a JTextField) -1 for each JTextField

- Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.
- every mistake: -1
less than 5 forms -> /50%

8. Unit Test 0.8/10

At least 100 unit tests.

- review that de unit test are no empty
- there must be at least one unit test that is failing.
- If all unit tests are OK, then it is half the grade. It means that the tests are not good enough
- every unit test is worth 0.1

The screenshot shows an IDE interface with two main panes. The left pane displays a file tree with Java files like FmtTableCustomer.java, FmtTableProduct.java, etc., and resources like banner.png and menu_bg.png. The right pane shows a code editor with a red highlight over a test method:

```

    @Test
    public void testRead_Document() {
        System.out.println("read");
        Connection connectionDataBase();
        StylistController stylistController = new StylistController();
        Document document = new Document();
        document.append("_id", new ObjectId("62da0ccac5d5a16610ef0234"));
        .append("identificationCard", "1755231683").append("name", "Mero")
        .append("number", "12")
        .append("payment", 12.0)
        .append("appointment", "13-07-2022")
        .append("address", "asd");
    }
  
```

Below the code editor is a 'Test Results' window titled 'Stylestirelia'. It shows the following information:

- Tests passed: 75.00 %
- 6 tests parsed, 2 tests failed. (8.301 s)
- Failed tests:
 - ec.edu.espe.stylestirelia.controller.BasicCont
 - testRead_Document Failed expected<1> != actual<2>
 - testRead_String_String Failed expected<1> != actual<2>

Logs at the bottom show MongoDB connection details:

```

create
delete
read
Jul. 28, 2022 8:26:31 A. M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[127.0.0.1:27017], srvHost=stylestirelia.firebaseio.mongodb.net, mode=MULTIPLE, requiredClusterType=REPLICASET}
Jul. 28, 2022 8:26:31 A. M. com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
  
```

TEAM 02 DAMAGE
(Granda Carlos)

GITHUB <https://github.com/MateoCondor/PointOfSaleSystem>

Project: Bakery point of sale system

- | | |
|-----------------------------------|------|
| 5. Chavez Escudero Genaro David | 0% |
| 6. Chicaiza Ortiz Frank Sebastian | 0% |
| 7. Chiriboga Zurita Daniel Isai | 80% |
| 8. Condor Sosa Mateo Javier | 100% |



Rubric:

- | | |
|-----------------------|--------|
| 1. Requirements | 9.5/10 |
| 2. UML Class Diagram | 8.8/10 |
| 3. Clean Code | 9.8/10 |
| 4. Db (MongoDb Atlas) | 9.5/10 |
| 5. GUI | 10/10 |
| 6. Tables | 5/10 |
| 7. Forms | 19 /30 |
| 8. Unit Tests | 5/10 |

TOTAL: 76.6/100

GitHub: 90/100

Proof

1. Requirements (Features/items)
 - Cost
 - Payment
 - Worker
 - Product
 - Provider
 - Cash Register

REQUERIMIENTOS PARA EL SISTEMA DE PUNTO DE VENTA DE LA PANADERIA POMPEI

La **panadería** POMPEI ha registrado todos sus movimientos en los **Libros** de contabilidad de su negocio, pero no ha sido suficiente para llevar un control de calidad respecto de las **finanzas** y sus **productos**.

De este modo se necesita crear un sistema de punto de venta adecuado a la panadería para optimizar el tiempo con los siguientes requerimientos:

El usuario necesita que se registren los **trabajadores** de su negocio para que ellos manipulen el sistema y todo tenga un **registro**.

El **usuario** necesita que el sistema permita la creación de los diferentes **productos** de su negocio, para lo cual se establecerán parámetros según se cobre o no IVA. El **usuario** necesita clasificar sus productos de acuerdo a la **Marca y Categoría**, para una mejor **búsqueda** en el sistema.

El usuario necesita crear una sección para registrar la información el **Cliente**, ya que con esa información puede adecuar el servicio según las necesidades el **Cliente**.

El usuario necesita una **sección** en el sistema que le permita registrar las **ventas** que se hacen a diario en el negocio, cada una con su respectivo número de **factura**, productos comprados y valor total a pagar, toda esta información es importante que se guarde al instante y de haber algún inconveniente con el registro poder anularlo.

El usuario necesita una sección en el sistema que le permita registrar la información de los **Proveedores** del **negocio**, ya que es de vital importancia esos datos para el abastecimiento del negocio, además el usuario a veces adeuda con el Proveedor por lo cual se necesita registrar el valor pagado y el valor adeudado al Proveedor.

El usuario necesita una sección en el sistema que le permita registrar los **Pagos y Gastos** del sistema, tales como **Pago** al Proveedor, **Gastos** tales como servicios básicos, etc.

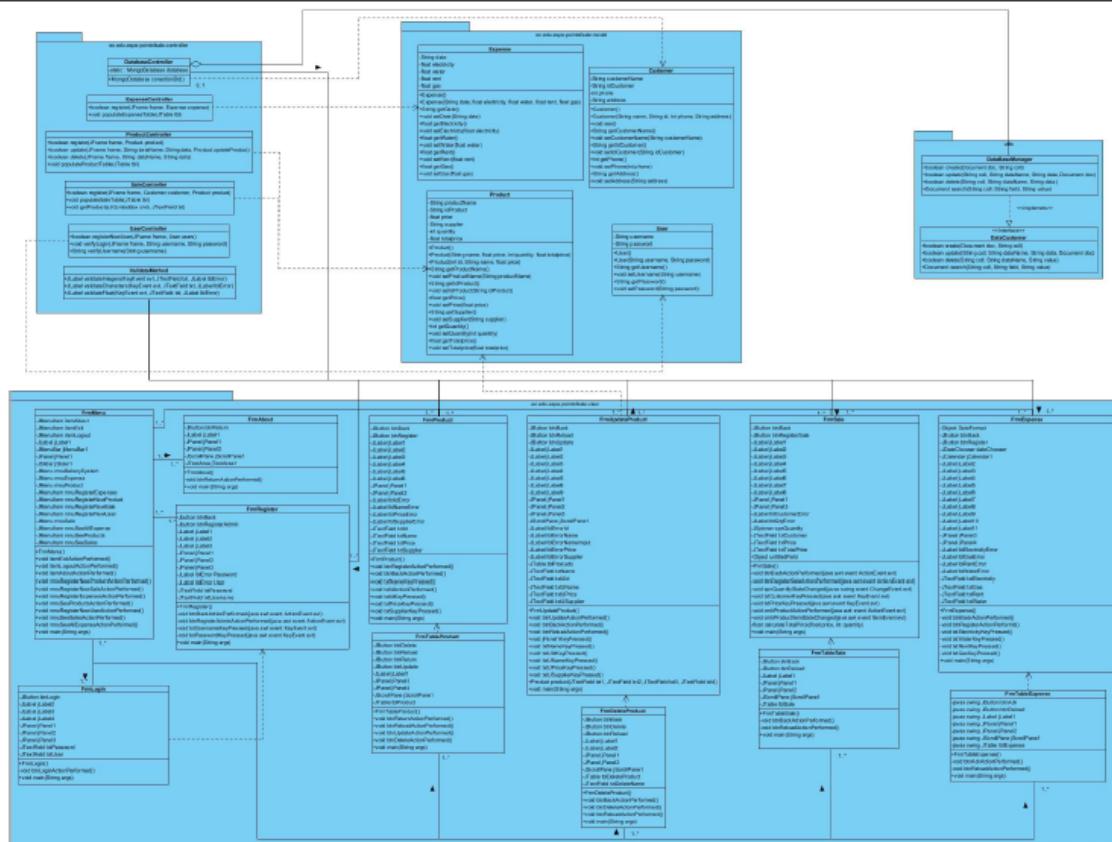
El usuario necesita una sección para la **Caja** del sistema, donde se registran la cantidad de **dinero** de los **ingresos del negocio**, los **retiros** que se hacen durante el día y los **Pagos y Gastos** del sistema.

El usuario necesita que cada Usuario cumpla una función indicada en el sistema, por lo cual se necesita privar de ciertas funciones a los **Trabajadores**, solo el **Administrador**/es tendrá acceso a todas las secciones del sistema.

El usuario necesita una sección de **Reportes** de cada una de las secciones antes mencionadas ya que necesita la información a la mano de los **Usuarios**, **Proveedores**, **de las Compras y Ventas** diarias por **fechas y productos** de acuerdo al stock,etc.

2.UML Class Diagram

- Classes
- Attributes
- Methods
- Abstract Classes/Interfaces 2/2
- Abstract methods 2/2
- Polymorphism (overridden methods) 2/2
- Inheritance 1 /2
- MVC-> packages model(POJO classes), view(GUI), controller(Abstract C./Concrete C./Interfaces) 1.8 /2



Observations

- Missing Inheritance
- In MVC have labels and panels

3.Clean Code

Pitfalls/Code Smells (0.1)

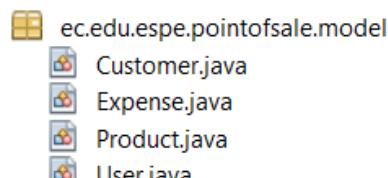
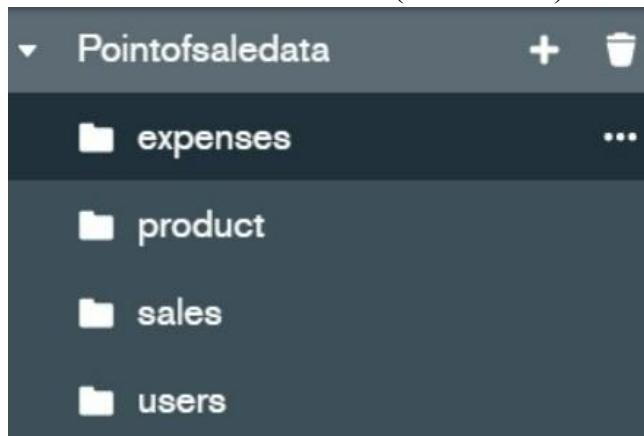
```
public float getGas() {  
    return gas;  
}  
  
public int getPhone() {  
    return phone;  
}  
  
/**  
 * @param phone the phone to set  
 */  
public void setPhone(int phone) {  
    this.phone = phone;  
}
```

4. Db (MongoDb Atlas)

Database Name

Collection vs Model classes

attributes (collections) vs attributes (classes)



Observation:

You have changed a class

5. GUI

Splash Window

Authentication

2/2 (it should work)

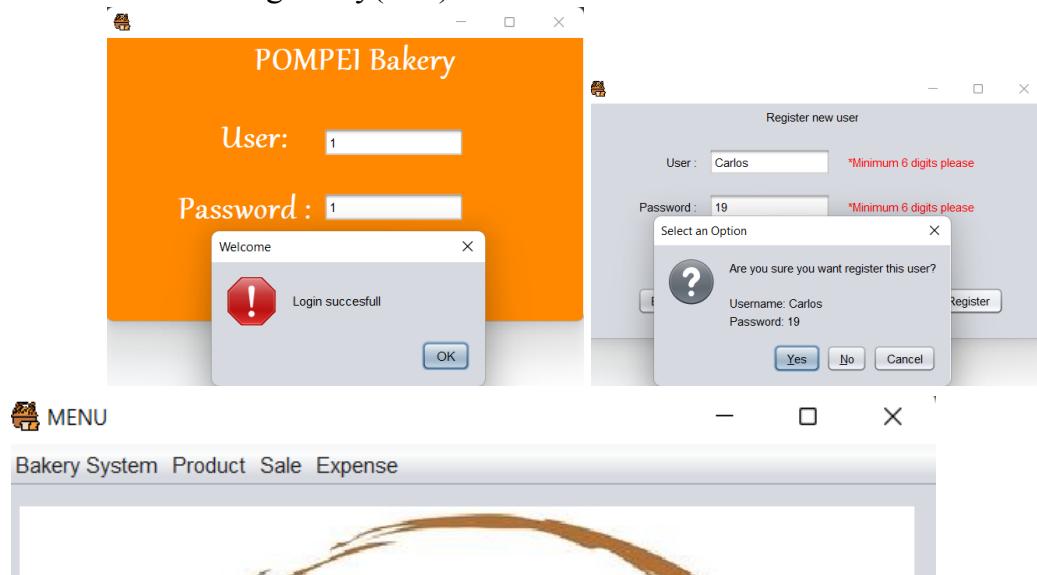
Application (Menu)

2/2 (requirements) (-0.2)

Forms

6/6 (requirements)

Navigability(-0.2)



6. Tables

Date come from DB

5/5

```
_id: ObjectId('62de949ec0b4cb16731b4ecd')
idProduct: "190622"
productName: "break"
price: 12
supplier: "morales"

_id: ObjectId('62e13ff0646b4e2a90c74e82')
idProduct: "001"
productName: "break"
price: 12
supplier: "Fmorales"
```

| Product list | | | |
|--------------|--------------|-------|---------------|
| Id | Name | Price | Supplier |
| 1 | pan de reyes | 1.5 | juan |
| 2 | cachito | 0.15 | esteban |
| 3 | biscocho | 0.25 | jose |
| 3 | Tres leches | 1.3 | Pompei Bakery |
| 190622 | break | 12.0 | morales |
| 001 | break | 12.0 | Fmorales |

Print date to printer

0/5

7. Forms

at least, there must be 6 different forms, every form is worth 5 points

-Login Form

-5 Forms for Information (CRUD operations, 4 business rules)

Every Form

- Every input (JTextField) is validated
- There are different types of inputs (not everything is a JTextField) -1 for each JTextField
- Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

every mistake: -1
less than 5 forms -> /50%

- a) Enter the name of the product without validating
- b) find the name of the product if you put letters

The screenshot shows a window titled "Delete Product". At the top, there is a table with columns: Id, Name, Price, and Supplier. The data is as follows:

| Id | Name | Price | Supplier |
|--------|--------------------|-------|----------------|
| 1 | pan de reyes | 1.5 | juan |
| 2 | cachito | 0.15 | esteban |
| 3 | biscocho | 0.25 | jose |
| 3 | Tres leches | 1.3 | Pompeii Bakery |
| 190622 | break | 12.0 | morales |
| 001 | break | 12.0 | Fmorales |
| 5 | melva de chocolate | 0.25 | adriana |

Below the table, there is a label "Insert the name of the product that you want delete:" followed by a text input field containing "9090d". At the bottom, there are three buttons: "Back", "Delete", and "Reload".

- Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller methods, and will use some model class.

a) the buttons of the crud are missing

The screenshot shows a window titled "Register new user". It contains two text input fields: one for "User" and one for "Password". To the right of the "User" field, a red message says "*Minimum 6 digits please". To the right of the "Password" field, there is a red asterisk symbol (*). At the bottom, there are two buttons: "Back" on the left and "Register" on the right.

 Register bills

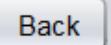
Date: 

Electricity: \$ * 

Water: \$ * 

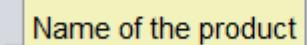
Rent: \$ * 

Gas: \$ * 

 Register product

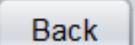
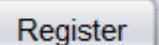
PRODUCT

Name : *Only characters please


ID : *

Price : \$ * 

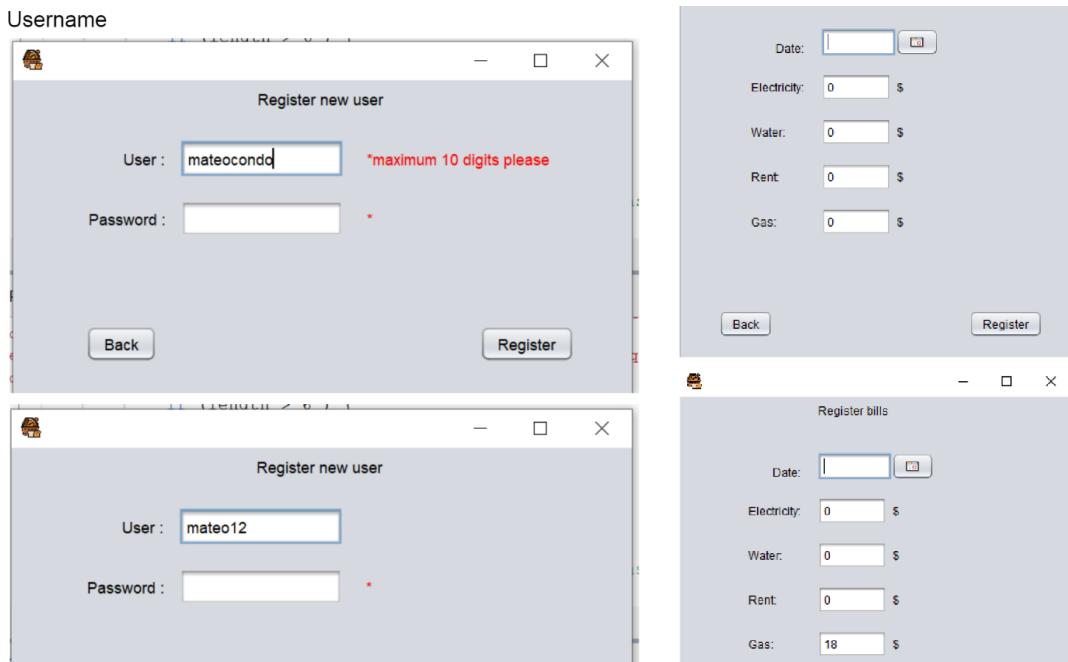
Supplier : *

8. Unit Test

At least 100 unit tests.

- review that de unit test are no empty
- there must be at least one unit test that is failing.
- If all unit tests are OK, then it is half the grade. It means that the tests are not good enough
- every unit test is worth 0.1



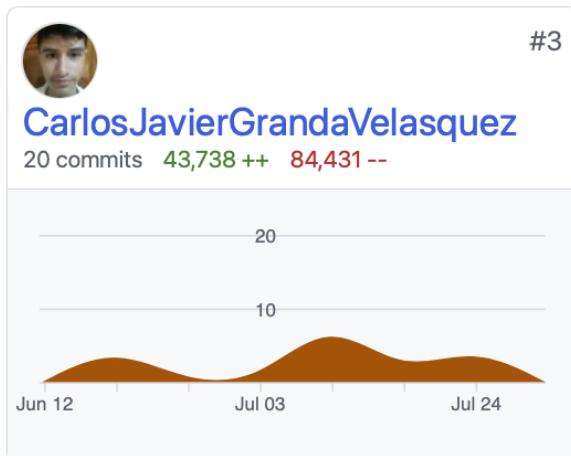
TEAM 03 Syntax Error (César Loor)

GITHUB: https://github.com/CarlosJavierGrandaVelasquez/SE_FrutiApp.git

Project: Fruit App

| | |
|-----------------------------------|------|
| 9. Granda Velasquez Carlos Javier | 90% |
| 10. Haro | |
| 11. Ibarra Gaona Ronny Joel | 90% |
| 12. Imbaquinga Guaña Jose Ricardo | 100% |

INSPECTOR TEAM 04 “GADC.MSI”



Rubric:

| | |
|------------------------|--------|
| 1. Requirements | 9/10 |
| 2. UML | 8.6/10 |
| 3. Clean Code | 9.7/10 |
| 4. DB (Mongo DB Atlas) | 8.5/10 |
| 5. GUI | 7.5/10 |
| 6. Tables | 5/10 |
| 7. Forms | 16/30 |
| 8. Unit tests | 3/10 |

| | |
|-------|----------|
| TOTAL | 69.1/100 |
|-------|----------|

Proof:

- Cost
- Sales
- Login
- Inventory
- Demand
- Payment
- earnings

Requirements (features/Items)

SE_FrutiApp

Universidad De Las Fuerzas Armadas Espe

Team 03 Syntax Error Integrants Carlos Granda (Leader) Ronny Ibarra Jose Imbaquinga

Project FRUIT APP

Date June 2, 2022

REQUIREMENTS

The system shall inform the user how much each product they buy costs The system shall indicate how many products are left to be sold The system shall save information of the money that has entered for the sale The system shall indicate the prices of each product found in the store. The system shall inform the availability of merchandise for the user's store. The system shall indicate how they carry out customer transactions. The system shall save the data of the workers each day they register The system shall give easy access to check the products that are in the box The system shall calculate in a statistic how many are sold per day

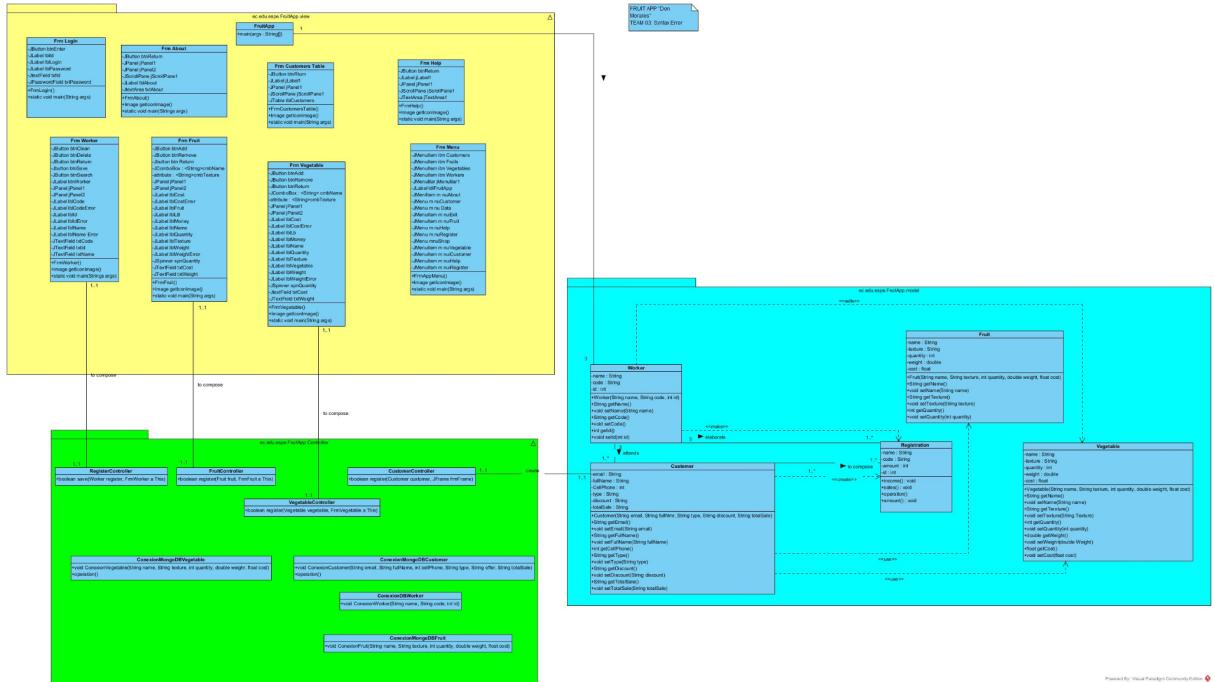
REQUIREMENTS

El sistema informará al usuario cuánto cuesta cada producto que compre el sistema guardará información de el dinero que a ingresado por la venta El sistema indicará los precios de cada producto que se encuentre en la tienda. El sistema indicará la demanda de productos para determinar cuándo comprar más o, por el contrario, cuando no. El sistema informará la disponibilidad de mercancía para la tienda del usuario. El sistema indicará la forma en que llevan a cabo las transacciones de los clientes. El sistema guardará los datos de los trabajadores cada día que se registren. El sistema deberá dar fácil acceso a revisar los productos que se encuentran en la caja El sistema deberá calcular en una estadística cuantos se vende por dias El sistema indicará cuántos productos quedan para ser vendidos

9. UML Class Diagram 8.5/10

- Classes -2
- Attributes -2
- Methods -2
- Abstract Classes/Interfaces 2/2
- Abstract methods 2/2
- Polymorphism (overridden methods) 2/2
- Inheritance 2/2
- MVC -> packages model (POJO classes), view(GUI), controller (Abstract C./Concrete C./Interfaces) 2/2

UML diagram



Clean Code

pitfalls(0.1)

a) Name of Variable

```
/*
public class Client {
    private String name;
    private int id;
    private int age;

    public Client() {
    }

    public Client(String name, int id, int age) {
        this.name = name;
        this.id = id;
        this.age = age;
    }
}
```

```
DefaultTableModel tabla = new DefaultTableModel() {
    @Override
```

View:

a) Dead Code -0.1

```
private void txtWeightActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
private void txtCostActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
private void cmbNameActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
private void mnuCustomerActionPerformed(java.awt.event.ActionEvent evt) {  
}
```

b) Unused imports -0.1

```
4  
5 import com.mongodb.MongoClient;  
6 import com.mongodb.client.MongoDatabase;  
7 import ec.edu.espe.FruitApp.model.ConexionDB;  
8 import ec.edu.espe.fruitApp.view.FrmAppMenu;  
9 import java.awt.Image;  
10 import java.awt.Toolkit;  
11  
12 import javax.swing.JFrame;  
13 import static org.bson.codecs.configuration.CodecRegistries.fromProviders;  
14 import static org.bson.codecs.configuration.CodecRegistries.fromRegistries;  
15 import org.bson.codecs.configuration.CodecRegistry;  
16 import org.bson.codecs.pojo.PojoCodecProvider;  
17 import com.mongodb.MongoClient;  
18 import com.mongodb.client.MongoCollection;  
19 import ec.edu.espe.fruitApp.model.Customer;  
20 import java.util.ArrayList;  
21 import java.util.List;  
22 import javax.swing.JFrame;  
23 import javax.swing.JOptionPane;  
24 import javax.swing.JTable;  
25 import org.bson.Document;  
26
```

c) Spanish -0.1

```
boolean mayusculas = key >= 65 && key <=90;  
boolean minusculas = key >= 97 && key <=122;  
boolean espacio = key == 32;  
  
if(! (minusculas || mayusculas || espacio )){  
    evt.consume();
```

d) Death Code

```

private void mnuCustomerActionPerformed(java.awt.event.ActionEvent evt) {
}

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

```

Mongo Atlas

| | |
|--|------|
| Database Name | -0.5 |
| Collections vs Model classes | -0.5 |
| attributes (collections) vs attributes (classes) | -0.5 |

▼ FruitDB

CustomerCollection

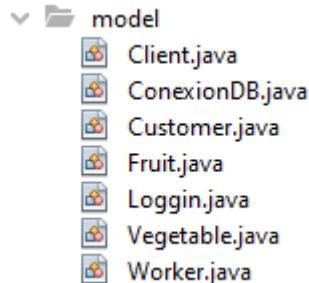
FruitCollection

VegetableCollection

WorkerCollection

Collection vs Model classes

Client collection is missing



GUI

| | | | |
|--------------------|-------|------------------|-------------------|
| Splash Window | | | |
| Authentication | 2/2 | (it should work) | |
| Application (Menu) | 1.6/2 | (requirements) | (-0.2) |
| Forms | 5/6 | (requirements) | Navigability -0.2 |

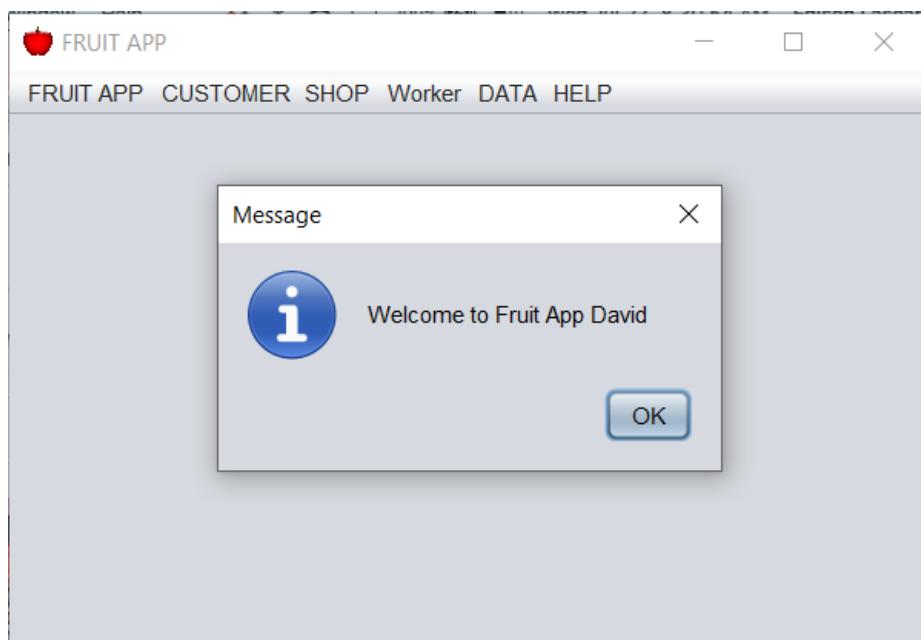
LOGGIN

Name:

Cell Phone: *Enter only digits (0)

Email:

Password



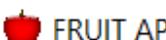


FRUIT APP

— □ ×

FRUIT APP CUSTOMER SHOP Worker DATA HELP

FRUIT APP 2.0.0



FRUIT APP

— □ ×

ABOUT FRUIT APP

FRUIT APP IS A INVENTORY IN THE CLOUD
IN THE DATA BASE OF MONGO DB

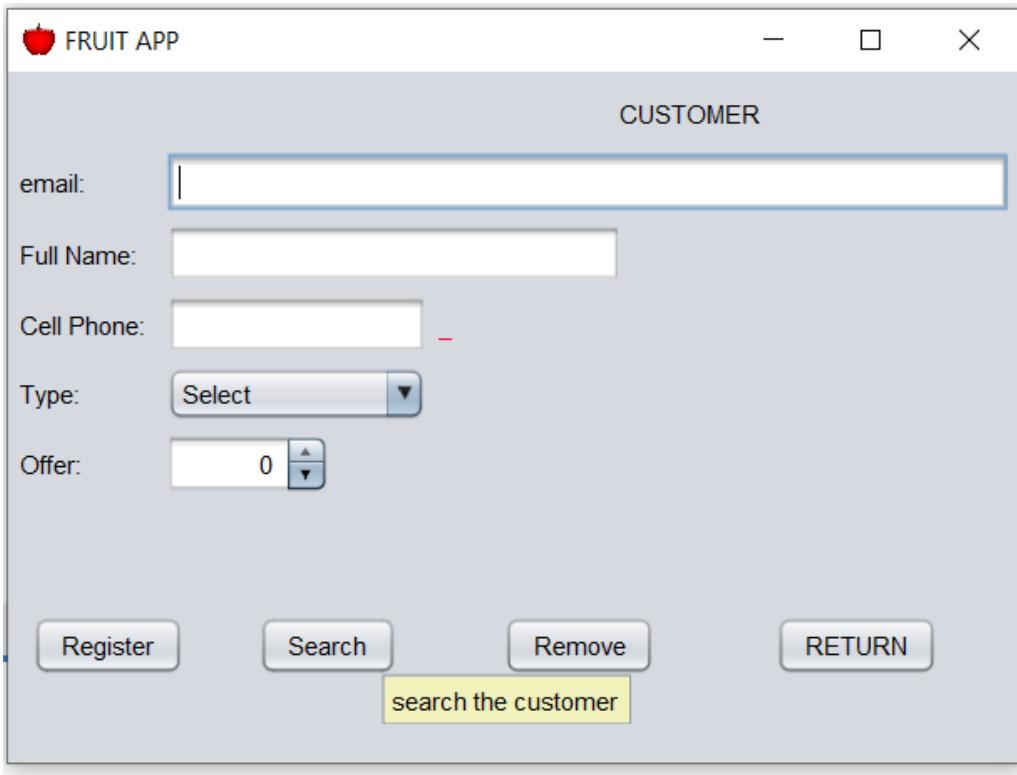
CREATOR: CARLOS GRANDA

Jose Imbaquinga

Roony Ibarra

VERSION: 1.0

RETURN



Forms

at least, there must be 6 different forms, every form is worth 5 points

- Login Form
 - 5 Forms for Information (CRUD operations, 4 business rules)
- Every form
- Every input (JTextField) is validated
 - There are different types of inputs (not everything is a JTextField) -1 for each JTextField
 - Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

every mistake: -1

less than 5 forms -> /50%

Unit Test

**TEAM 04 “GADC.MSI”
(Steven Pozo)**

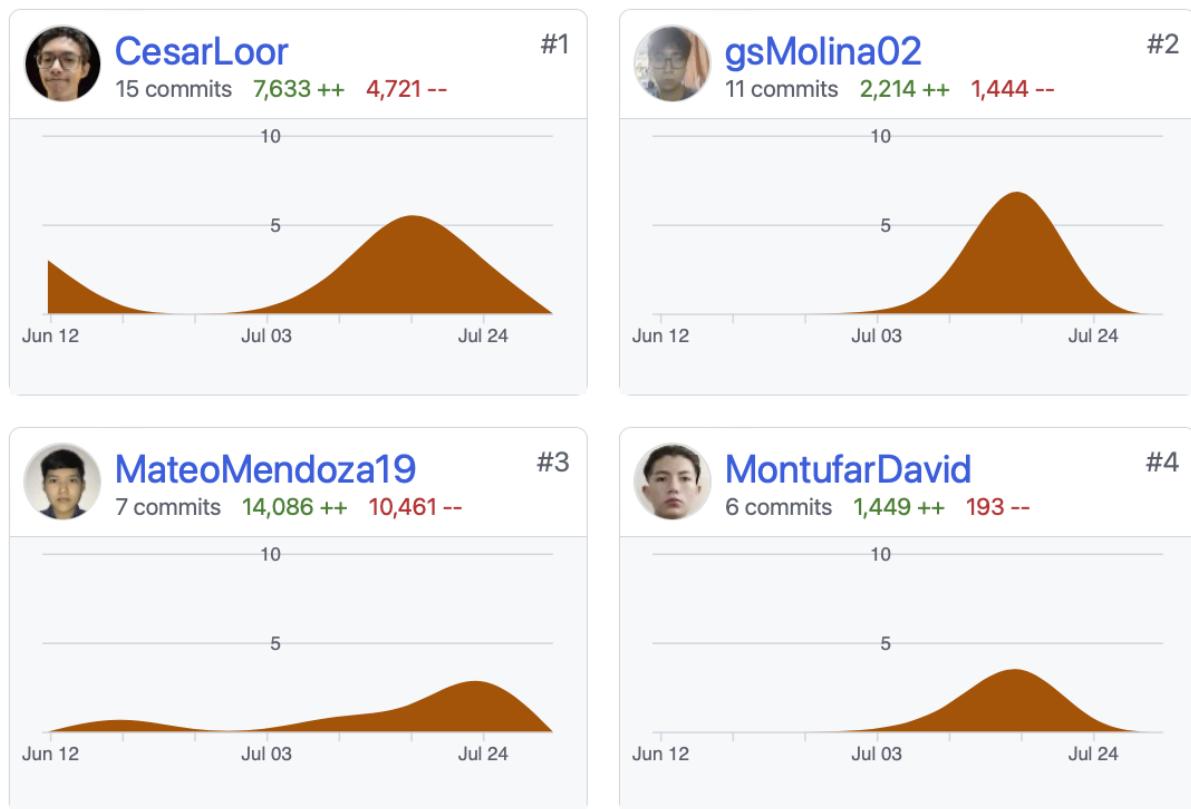
Github: <https://github.com/CesarLoor/G-4-GADC.MSI>

NameProject: CoopLatina

| | |
|----------------------------------|------|
| 13. Loor Mercado Cesar Ignacio | 100% |
| 14. Mendoza Arteaga Aldric Mateo | 90% |

INSPECTOR: TEAM 05 DEES Dev's

15. Molina Cuanco Gustavo Stiven 90%
16. Montufar Saltos David Ariel 80%



Rubric:

| | |
|-----------------------|---------|
| - requirements | 9/10 |
| - UML | 9.7/10 |
| - Clean Code | 9.4 /10 |
| - DB (Mongo DB Atlas) | 9.5 /10 |
| - GUI | 7.8710 |
| - TABLE | 4/10 |
| - FORMS | 14/30 |
| - UNIT TEST | 5/10 |

TOTAL: 68.47/100

Proof:

REQUIREMENTS

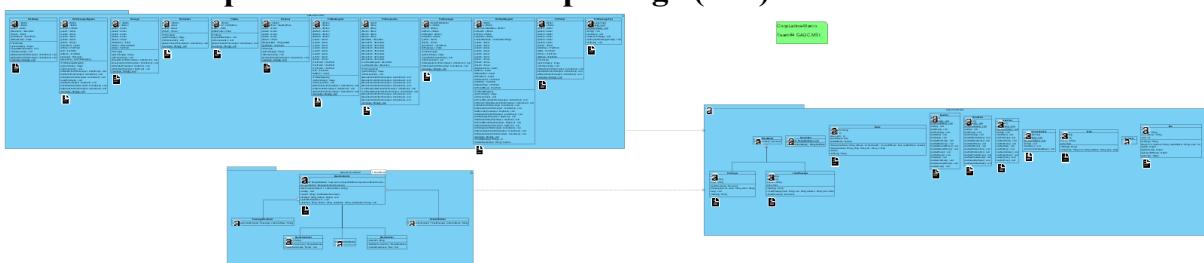
- Route
- Collection
- Passengers
- Consumption
- Earnings

Requirements

- The program allows the passenger to choose the type of route he/she needs.
- The program performs the passenger fare collection.
- The program counts how many passengers have already paid.
- The program calculates the fuel consumption depending on the type of route.
- The program shows the earnings at the end of the working day.

UML CLASS DIAGRAM

- Classes -2
- Attributes -2
- Methods -2
- Abstract Classes/Interfaces 2/2
- Abstract methods 2/2
- Polymorphism (overridden methods) 2/2
- Inheritance 2/2
- MVC -> packages model (POJO classes), view(GUI), controller (Abstract C./Concrete C./Interfaces) 2/2
 - * (getters and setters 9(-.01), labels(-0.1), panels(-0.1), no dependencies in the view package (-0.1)



Description:

- labels in package view -0.1
- panels in package view -0.1
- dependencies in package view and model -0.1

CLEAN CODE:

Model:

- a) Redundant or Meaningless Comments -0.1

```

public Object getid() {
    throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}

public Object getmatricule() {
    throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}

public Object getnameOfDriver() {
    throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}

public Object getroute() {
    throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}

public Route(String mananas, String string, String km, float f) {
    throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}

public Route() {
    throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}

```

b) Dead Code, this class is not used -0.1

```

1
2 package espe.edu.ec.CoopLatinaMarco.model;
3
4 /**
5 *
6 * @author Loor Cesar, DDCO-ESPE, GADC.MSI
7 */
8 public class Customer {
9
10 }

```

Controller: No problem.

View:

a) Dead code again, are not used

```

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
}

private void txtidActionPerformed(java.awt.event.ActionEvent evt) {

}

private void txtUserKeyPressed(java.awt.event.KeyEvent evt) {

}

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

private void txtIdActionPerformed(java.awt.event.ActionEvent evt) {
// 
}

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

```

```

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

private void lblIdKeyPressed(java.awt.event.KeyEvent evt) {
}

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

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

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

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

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

```

b) Instantiation must be in the next line -0.1

```

private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    FrmPassengerRegister view = new FrmPassengerRegister();
    view.setVisible(true);
    this.dispose();
}

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    Frmabout view = new Frmabout();
    view.setVisible(true);
    this.dispose();
}

private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
    FrmTicket view = new FrmTicket();
    view.setVisible(true);
    this.dispose();
}

private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
    FrmBus view = new FrmBus();
    view.setVisible(true);
    this.dispose();
}

private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt) {
    FrmBusRegister view = new FrmBusRegister();
}

private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    FrmPassengerRegister view = new FrmPassengerRegister();
}

private void btnMenuActionPerformed(java.awt.event.ActionEvent evt) {
    FrmCoopLatina view = new FrmCoopLatina();
}

private void btnTableActionPerformed(java.awt.event.ActionEvent evt) {
    FrmPassenger newFrame = new FrmPassenger();
}

private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
    FrmCoopLatina view = new FrmCoopLatina();
}

```

c) Dead code again, these imports are not used



import javax.swing.JTextField;

```

    | import java.sql.Connection;
    | import javax.swing.JOptionPane;
    | L import sun.security.krb5.internal.Ticket;

```

d) Spanish -0.1

```

String usuario = txtUser.getText();
DefaultTableModel modelo;

```

e) Redundant or Meaningless Comments again

```

//route.setPriceOfRoute(JTValidation(txtPriceOfRoute.getText()));
route.setAvailableRoute(availableRoute(cmbAvailableRoute.getSelectedItem().toString()));

```

f) Duplicate code -0.1

```

private void txtAddressKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();

    if((c<'a' || c>'z') && (c<'A' || c>'Z')) evt.consume();
}

private void txtBusKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();

    if((c<'a' || c>'z') && (c<'A' || c>'Z')) evt.consume();
}

private void txtRouteKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();

    if((c<'a' || c>'z') && (c<'A' || c>'Z')) evt.consume();
}

```

g) A java class in the view package -0.1

```

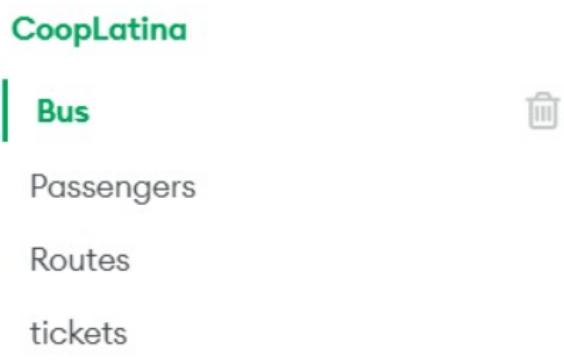
espe.edu.ec.CoopLatinaMarco.view
  FrmBus.java
  FrmBusRegister.java
  FrmContact.java
  FrmCoopLatina.java
  FrmLogin.java
  FrmPassenger.java
  FrmPassengerRegister.java
  FrmRouteRegister.java
  FrmRoutes.java
  FrmTicket.java
  Frmabout.java
  InputNumbersValidation.java

```

MONGODB ATLAS:

| | |
|---------------|------|
| Database Name | -0.5 |
|---------------|------|

| | |
|--|------|
| Collections vs Model classes | -0.5 |
| attributes (collections) vs attributes (classes) | -0.5 |



Collections Routes and tickets are voids (-0.5)

The image contains two side-by-side screenshots of the MongoDB interface. Both screenshots show the 'Bus' collection under the 'CoopLatina' namespace. In the first screenshot, the 'Find' tab is selected, showing a 'FILTER' button with the expression '{ field: 'value' }'. Below it, the text 'QUERY RESULTS: 0' is displayed. In the second screenshot, the 'Indexes' tab is selected, also showing a 'FILTER' button with the same expression and 'QUERY RESULTS: 0' text.

GUI:

| | |
|--------------------|--|
| Splash Window | |
| Authentication | 1.8/2 (it should work) |
| Application (Menu) | 1.6/2 (requirements) (-0.2) |
| Forms | 5.8/6 (requirements) Navigability -0.2 |

The username and password would be in mongoDB or a field txt, csv.
The program doesn't calculate the fuel consumption and doesn't show the earnings.



TABLES:

| | |
|-----------------------|----------------------------|
| data come from DB | 4/5 (Programming in mongo) |
| print data to printer | 0/5 |

FORMS:

at least, there must be 6 different forms, every form is worth 5 points

- Login Form
- 5 Forms for Information (CRUD operations, 4 business rules)
 - Every form
 - Every input (JTextField) is validated
 - There are different types of inputs (not everything is a JTextField) -1 for each JTextField
 - Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

every mistake: -1

less than 5 forms -> /50%

**TEAM 05 DEES Developers
Quimbiulco)**

GitHub: <https://github.com/stevenpozo/DEES-developers>

NameProject: LeatherFlowerSystem

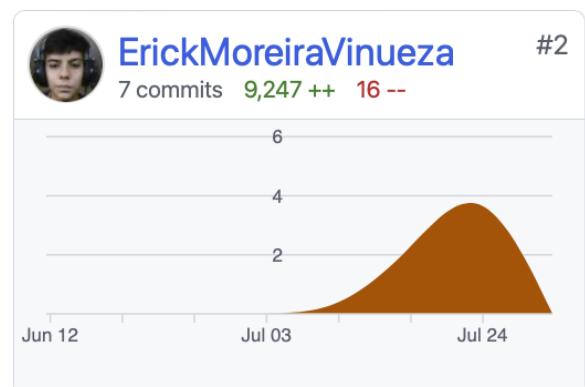
| | |
|------------------------------------|------|
| 17. Moreira Vinuela Erick Patricio | 95% |
| 18. Pabon Gonzalez Elkin Andres | 100% |
| 19. Ponce Arguello Diego Armando | 75% |
| 20. Pozo Analuisa Steven Jefferson | 90% |

INSPECTOR: TEAM 6 Codex++ (Juan

Quimbiulco)

GitHub: <https://github.com/stevenpozo/DEES-developers>

NameProject: LeatherFlowerSystem



Rubric:

| | |
|----------------------|--------|
| 1.Requirements | 9/10 |
| 2.UML Class Diagram | 9.7/10 |
| 3.Clean Code | 9.6/10 |
| 4.Db (MongoDb Atlas) | 10/10 |
| 5.GUI | 9.2/10 |
| 6.Tables | 10/10 |
| 7.Forms | 28/30 |
| 8.Unit tests | 8/10 |

TOTAL **93.5/100**

Proof:

1. Requirements (Features/items)

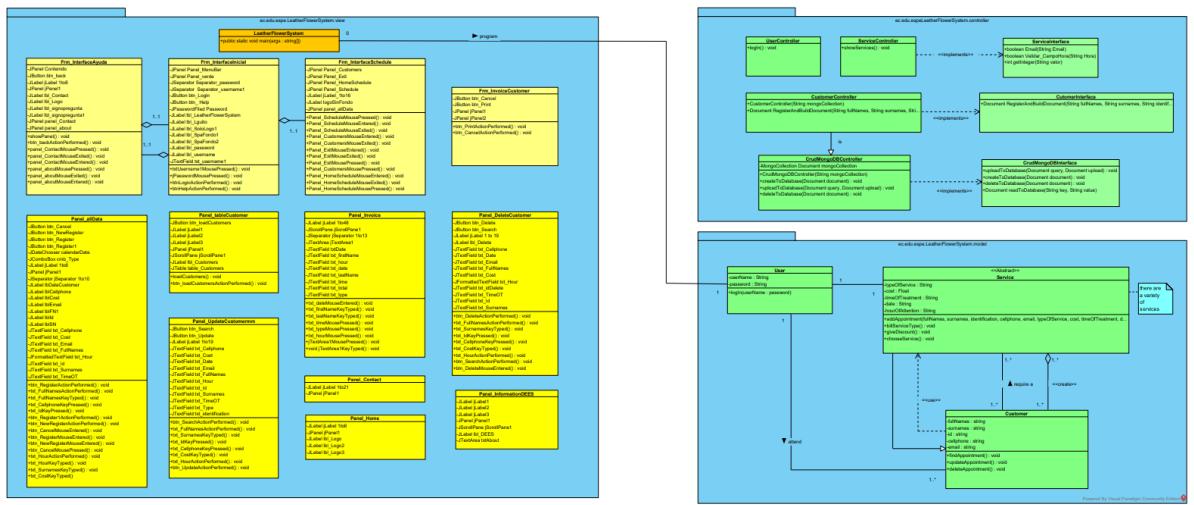
Text

- Log in
- Save customer data
- Show the amount to be charged for the service provided
- Edit customer data
- Delete customer data

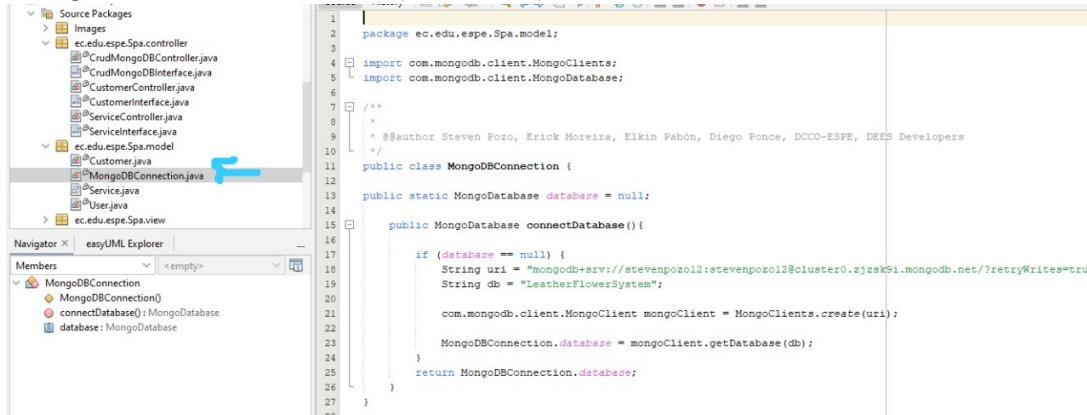
| HU Framework Matrix | | | | | | | | | | | | |
|---------------------|--------------------------|--|--|----------------------|---|---------------------------|------------------|------------------|----------|------------|---|---|
| ITEM | PROBLEM | WHAT (REQUEST) | WHAT FOR (SOLUTION) | FOR WHO (USER/OWNER) | HOW (DESCRIPTION OF TASKS) | MADE BY (PERSON RESPONS.) | HOW LONG (HOURS) | DATE OF DELIVERY | PRIORITY | STATUS | TEST (HOW TO VERIFY) | COMENTARIES |
| REQ001 | Options menu | Login | So you can schedule, postpone or cancel appointments | Customer (owner) | In the main interface a menu will be displayed where it will be indicated that you must log in to make any changes in the system, if there are no changes you can view the calendar in another option of the main menu. | Steven Poco | In process | August | high | In process | At the time of entering your previously created username and password, you can enter to make any changes in the system. | The username and password must be alphanumeric. |
| REQ002 | Schedule appointments | Save customer data | To be able to schedule a day with your appointment | Customer (owner) | The owner must enter the name, surname, DNI number and email of the client that will be requested. | Steven Poco | In process | August | high | In process | Once the data is saved, the appointment already scheduled can be viewed later in a record. | The data entered must be correct. |
| REQ003 | Generate service billing | Show the amount to be charged for the service provided | So that there is proof of the service provided and a backup of it | Customer (owner) | Once the appointment is scheduled, you can pay for it and generate the payment document, this document will be generated at the beginning of this you must review the amount that the service that will be provided acquires. | Erick Pabón - Diego Ponce | In process | August | high | In process | Everything will be verified once the payment invoice is issued. | The data entered must be correct. |
| REQ004 | Defer appointments | Edit customer data | To be able to defer the date of your appointment | Customer (owner) | The owner must edit the date of the customer that will change the day | Erick Moreira | In process | August | high | In process | Once the change is saved, the day already altered can be viewed later in a record by the customer. | The data changed must be correct. |
| REQ005 | Delete Appointment | Delete customer data | To be able to delete an appointment that has been previously scheduled | Customer (owner) | The owner must delete the data recorded about the appointment. | Erick Moreira | In process | August | high | In process | Once the data is deleted, the owner could verify it looking the list of scheduled appointments | The data must be deleted correctly. |

2.UML Class Diagram

- Classes 2
- Attributes 2
- Methods 2
- Abstract Classes/Interfaces 2
- Abstract methods 2
- **Polymorphism (overridden methods) 0**
- Inheritance 2
- MVC -> packages model (POJO classes), view(GUI), controller (Abstract C./Concrete C./Interfaces) 2/2
- * (getters and setters 9(-.01), **labels(-0.1)**, **panels(-0.1)**, **no dependencies in the view package (0.1)**



3.Clean Code Pitfalls/Code Smells (0.1) MongoDb class is in model (controller)



Spanish

```

@Override
public Boolean Validar_CampoHora(String Hora) {
    boolean b;
    char[] a = Hora.toString().toCharArray();
    String[] c = Hora.split(":");
    if ((a[0] == ' ') || (a[1] == ' ') || (a[2] == ' ')
        || (a[3] == ' ') || (a[4] == ' ')
        || (getInteger(c[0]) > 24) || (getInteger(c[1]) > 59)){
        b = false;
    }else {
        b = true;
    }
    return b;
}

@Override
public int getInteger(String valor) {
    int integer = Integer.parseInt(valor);
    return integer;
}
}

```

```

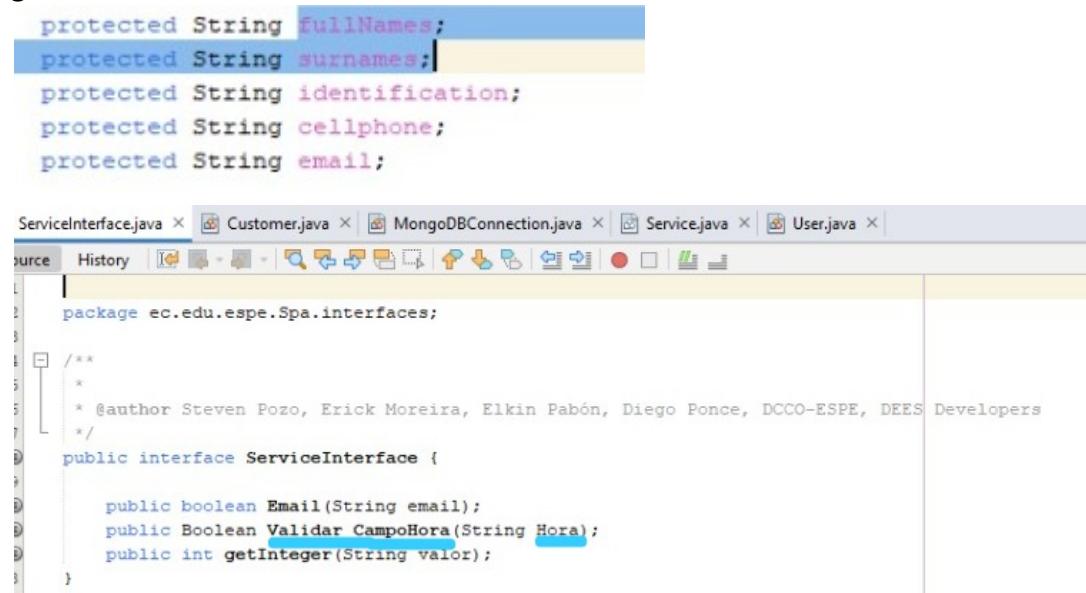
package ec.edu.espe.Spa.controller;

/**
 *
 * @author Steven Pozo, Erick Moreira, Elkin Pabón, Diego
 */
public interface ServiceInterface {

    public boolean Email(String email);
    public Boolean Validar_CampoHora(String Hora);
    public int getInteger(String valor);
}

```

Attributes full names and surnames are unnecessary since full name the most general.



```

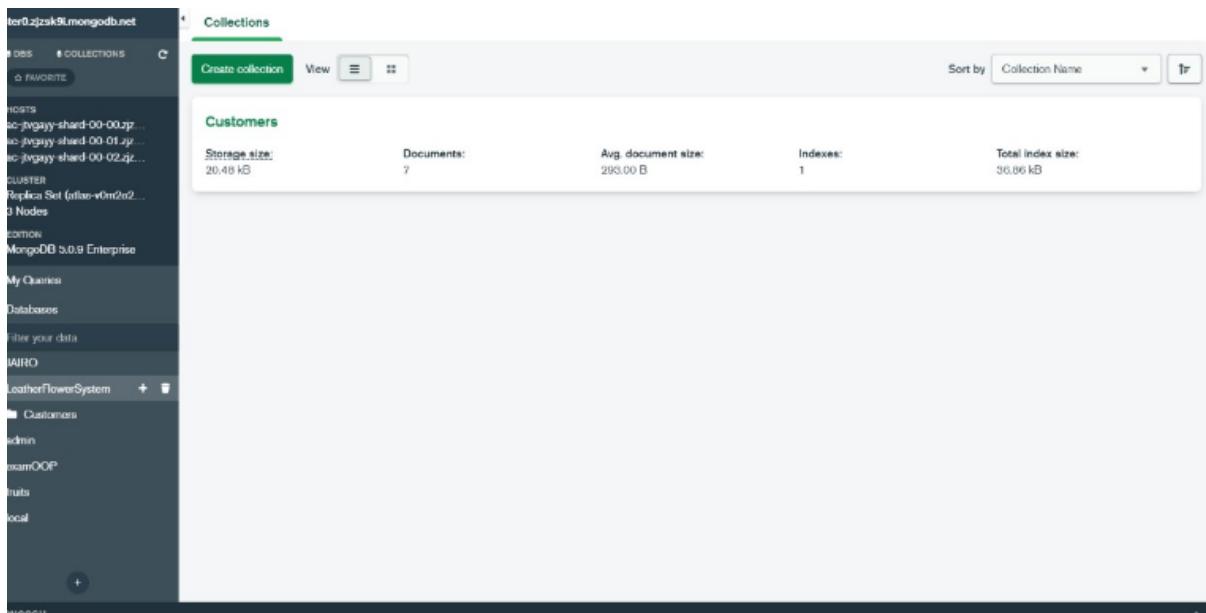
protected String fullNames;
protected String surnames;
protected String identification;
protected String cellphone;
protected String email;

```

The word Email and Validar Campo Hora isn't working with the appropriate language

DB (MONGOdb Atlas)

| | |
|--|------|
| Database Name | -0.5 |
| Collections vs Model classes | -0.5 |
| attributes (collections) vs attributes (classes) | -0.5 |



GUI

Splash Window

| | | |
|--------------------|---|---|
| Authentication | 2 | (it should work) |
| Application (Menu) | 2 | (requirements) |
| Forms | 6 | (requirements) Navigability -0.2 |

Tables:

| | |
|-----------------------|-----|
| data come from DB | 5/5 |
| print data to printer | 5/5 |

Forms 28/30

at least, there must be 6 different forms, every form is worth 5 points

- Login Form Quilumbaquin Lanchimba Jairo Smith
 - Quimbiulco Juan Diego
 - Rivera Espin Carlos Sebastian
 - Silva Velasquez Raul Andres
 - 5 Forms for Information (CRUD operations, 4 business rules)
- Every form
- Every input (JTextField) is validated
 - There are different types of inputs (not everything is a JTextField) -1 for each JTextField
 - Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

every mistake: -1

less than 5 forms -> /50%

Unit Test 0.08

At least 100 unit tests.

- review that the unit tests are not empty
- there must be at least one unit test that is failing.
- If all unit tests are OK, then it is half the grade. It means that the tests are not good enough
- every unit test is worth 0.1

**TEAM 06 CODEX++
(Daniela Tituaña)**

| | |
|--------------|------|
| Quilumbaquin | 100% |
| Quimbiulco | 95% |
| Rivera | 75% |
| Silva | 80% |

INSPECTOR: TEAM 07 MyWayCode



Github:https://github.com/Jdquimbiulco/TC_BooKify.git

Project name: BookiFiy

21.

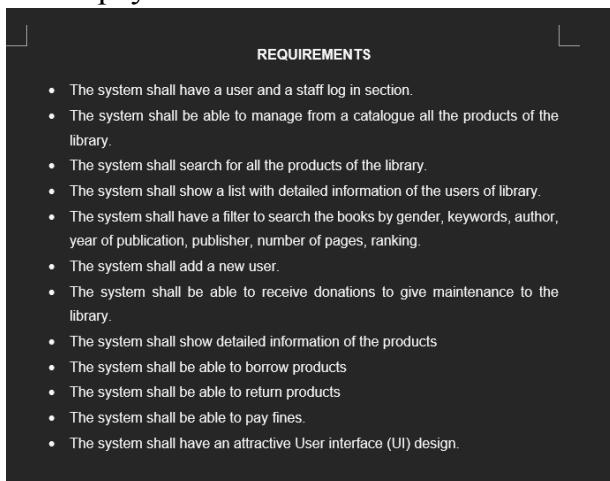
Rubric:

1. Requirements 8/10
2. UML Class Diagram 7.7/10
3. Clean Code 8.5/10
4. DB (MONGOdb Atlas) 7.9/10
5. GUI 8 /10
6. Table: 4 /10

| | |
|---------------|----------|
| 7. Forms | 15/30 |
| 8. Unit tests | 3/10 |
| <hr/> | |
| TOTAL | 62.1/100 |
| GitHub | /100 |
| Avegarre | /100 |

1. Requirements (Features/items)

- ★ log in section
- ★ user list
- ★ search product
- ★ delete information
- ★ product catalog
- ★ user information
- ★ product information
- ★ book filter
- ★ payment of fines

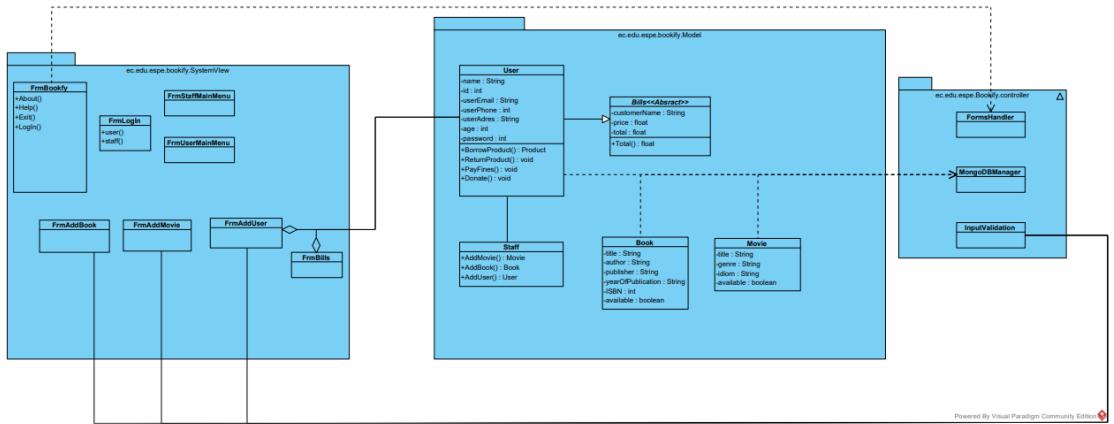


2. UML Class Diagram

Requirements (Features/items)

UML Class Diagram

- Classes 2
 - Attributes 2
 - Methods 2
 - Abstract Classes/Interfaces 2
 - Abstract methods 2
 - Polymorphism (overridden methods) 0
 - Inheritance 2
 - MVC -> packages model (POJO classes), view(GUI), controller (Abstract C./Concrete C./Interfaces)(getters and setters (-.01), labels(-0.1), panels(-0.1), no dependencies in the view package (0.1)
- 1.7/2



Observations:

- There are no methods in the controller package.
- The methods must not be in the model package.
- Missing relationships in package model.
- Methods must have parameters.

3.Clean Code

Pitfalls/Code Smells (0.1)

import javax.swing.Icon;
 import javax.swing.ImageIcon;

Unused Import

```
import ec.edu.espe.Bookify.controller.FormsHandler;
import ec.edu.espe.Bookify.controller.InputValidation;
```

unused imports -0.1

```
public class Staff {
    private String StaffName;
    private int StaffId;
    private int StaffAge;
    private int StaffPhone;
    private String StaffAddress;
    private String StaffPassword;
    private boolean StaffBlackList;
```

variables that start with a capital letter -0.1

```
Toolkit miPantalla
Image miIcono = miE
```

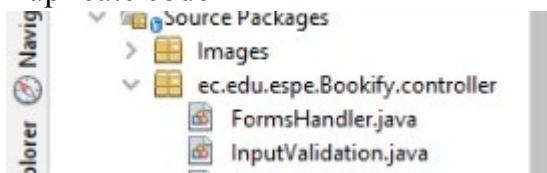
inconsistent variables

```

this.setLocationRelativeTo(null);
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
Toolkit miPantalla = Toolkit.getDefaultToolkit();
Image miIcono = miPantalla.getImage("src/Images/BookIco.png");
setIconImage(miIcono);
btnSearch.setOpaque(false);
btnSearch.setContentAreaFilled(false);
btnSearch.setBorderPainted(false);

```

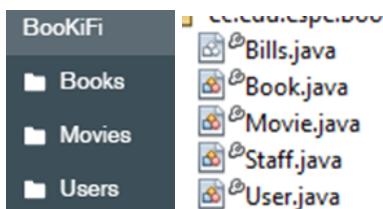
Duplicate code



Package with different name to the other

4.DB (MongoDb Atlas)

| | |
|--|------|
| Database Name | -0.5 |
| Collections vs Model classes | -0.5 |
| attributes (collections) vs attributes (classes) | -0.5 |



The class of Bills and staffs is missing in the database.

```

/*
public abstract class Bills {

    private String customerNames;
    private String dateOfReturn;
    private float    price;
    private float    total;
}

    public void total(JTextField txtField) {
        total = price*12;
        txtField.setText("$"+total);
}

```

dataOfReturn is missing from class diagram customerNames does not match diagram.

The total function does not match the diagram.

```

/*
public class Staff {
    private String StaffName;
    private int StaffId;
    private int StaffAge;
    private int StaffPhone;
    private String StaffAddress;
    private int StaffPasword;
    private boolean StaffBlackList;
}

public class User extends Bills{
    private String UserName;
    private int UserId;
    private String UserEmail;
    private int UserPhone;
    private String UserAddress;
    private int UserAge;
    private int UserPassword;
    int Pasuwu;
    int Password = 1945;
}

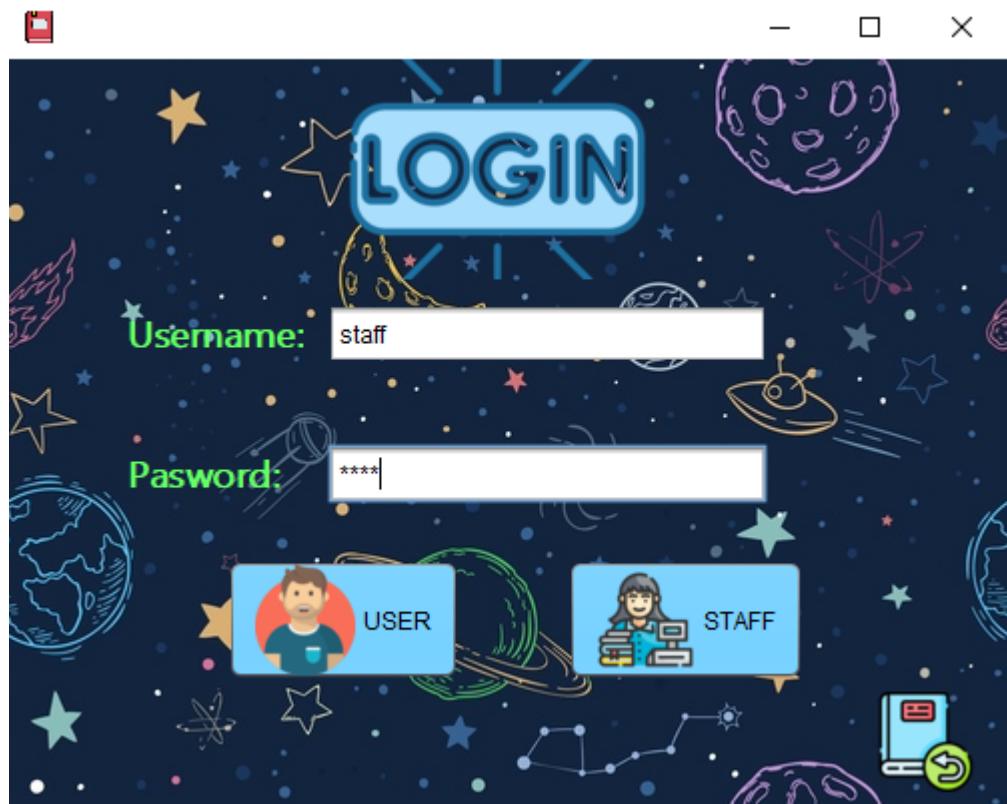
```

The attributes of the user and staff class do not match the diagram.

| Collection | Documents | Avg. document size | Indexes | Total index size |
|------------|-----------|--------------------|---------|------------------|
| Books | 1 | 111.00 B | 1 | 24.56 kB |
| Movies | 0 | 0 B | 1 | 12.29 kB |
| Users | 0 | 0 B | 1 | 12.29 kB |

5.GUI

| | | | |
|--------------------|---|------------------|-------------------|
| Splash Window | | | |
| Authentication | 2 | (it should work) | |
| Application (Menu) | 2 | (requirements) | (-0.2) |
| Forms | 6 | (requirements) | Navigability -0.2 |



LogIn information it's a predesignated password and user (-0.1)

6.Tables:

data come from DB 5

print data to printer 5

| PayBills | | | | |
|----------|---------|----------------------|------|--|
| Name | Id | Book | Bill | |
| jairo | 1714564 | Algebra de baldor | 5 | |
| Luis | 1231231 | Fundamentos d... | 5 | |
| Carlos | 1231231 | Principios de ele... | 5 | |
| Ricardo | 1231231 | La divina comedia | 5 | |

```

Object[] userRow = new Object[]{"jairo", "1714564","Algebra de baldor"};
table.addRow(userRow);
userRow = new Object[]{"Luis", "1231231","Fundamentos de Circuitos I"};
table.addRow(userRow);
userRow = new Object[]{"Carlos", "1231231","Principios de electronica"};
table.addRow(userRow);
userRow = new Object[]{"Ricardo", "1231231","La divina comedia","5"};
table.addRow(userRow);

```

The system does not display database information.

7.Forms

at least, there must be 6 different forms, every form is worth 5 points

- Login Form
 - 5 Forms for Information (CRUD operations, 4 business rules)
 - Every form
 - Every input (JTextField) is validated
 - There are different types of inputs (not everything is a JTextField) -1 for each JTextField
 - Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.
- every mistake: -1
 less than 5 forms -> /50%

Unit Test

**TEAM 07 MyWayCode
BettaCoders(Luis Burbano)**

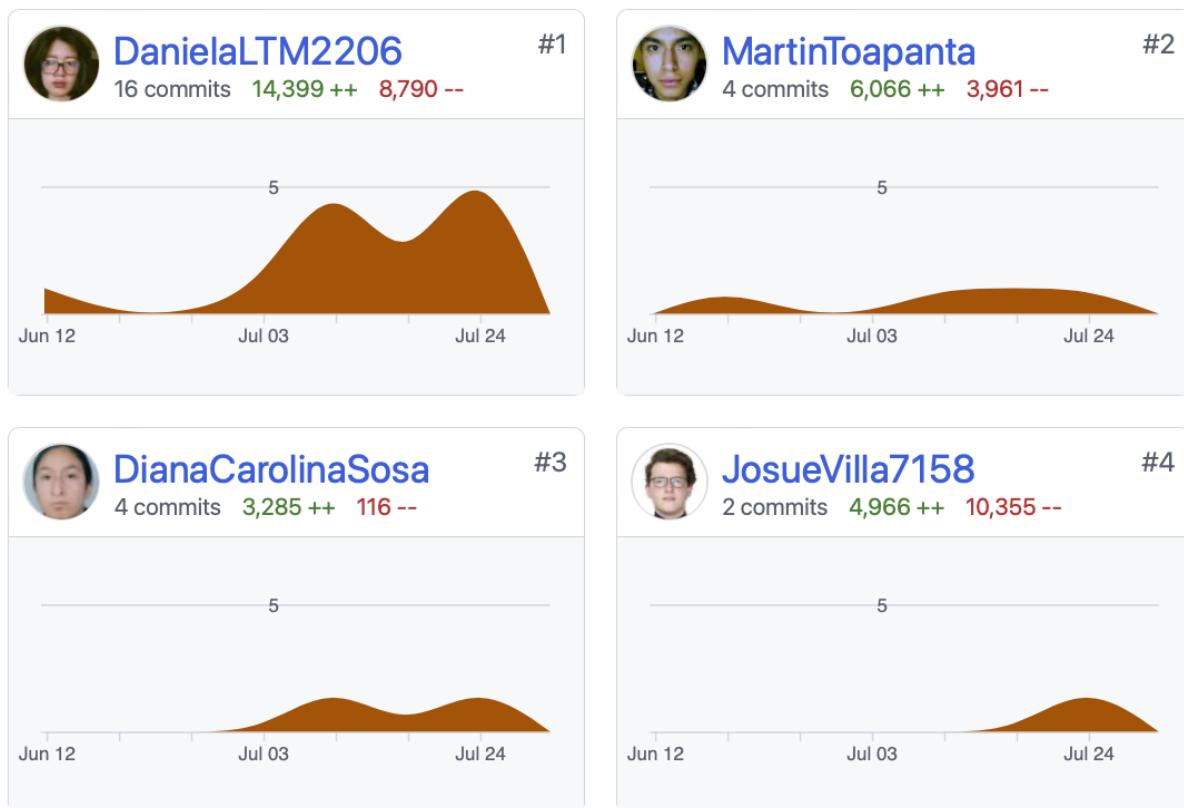
INSPECTOR: TEAM 01

Git Hub: <https://github.com/DanielaLTM2206/TWC-Store-System-.git>

Project: TWC-Store-System-

| | |
|-------------------------------------|------|
| 22. Sosa Romero Diana Carolina | 70% |
| 23. Tituaña Moreno Daniela Lissette | 100% |
| 24. Toapanta Vásquez Martin Honorio | 80% |

25. Villavicencio Campoverde Bolivar Josue 70%



Rubric:

| | |
|--------------------|---------|
| Requirements | 10/10 |
| UML Class Diagram | 7.8/10 |
| Clean Code | 9.6/10 |
| Db (MongoDb Atlas) | 9.5 /10 |
| GUI | 7/10 |
| Tables | 5/10 |
| Forms | 22/30 |
| UnitTest | 0.2/10 |

TOTAL 71.1/100

Proof:

Requirements (Features/items)

Requerimientos:

- El sistema debe contar con una sección de ingreso de usuarios y personal.
- El sistema tendrá la opción de agregar un nuevo usuario cuando sea necesario.
- El sistema mostrará el tiempo de vida de cada producto cuando se registren, teniendo así un producto perfecto para la venta.
- El sistema debe contar con el registro y ubicación de cada producto.
- El sistema debe tener un registro constante de cada venta realizada.
- El sistema debe aceptar devoluciones de ciertos productos por mal estado o por caducidad del producto.
- El sistema debe mostrar la cantidad para que sepa la cantidad necesaria para reponerlo.
- El sistema debe mostrar los horarios de mayor venta para una mejor organización dentro del local.
- El sistema debe mostrar una lista con información detallada sobre los usuarios del producto.
- El sistema debe mostrar información detallada sobre los productos.

- A. Login
- B. Add user
- C. Registration product
- D. Accept refund of product
- E. Stock of product
- F. Show the quantity to replenish the product
- G. Show horary
- H. List users
- I. List products

UML Class Diagram

- Classes -2
- Attributes -2
- Methods -2
- Relations -2
- Abstract Classes/Interfaces 2
- Abstract methods 2
- Polymorphism (overridden methods) 2
- Inheritance 2
- MVC -> packages model (POJO classes), view(GUI), controller (Abstract C./Concrete C./Interfaces) 2
 - (getters and setters 9(-.01), labels(-0.1), panels(-0.1), no dependencies in the view package (0.1)

GUI

| | | |
|--------------------|---|-----------------------|
| Splash Window | | |
| Authentication | 2 | (it should work) |
| Application (Menu) | 2 | (requirements) (-0.2) |

Forms

6 (requirements)

Navigability -0.2

Tables:

| | |
|-----------------------|---|
| data come from DB | 5 |
| print data to printer | 5 |

Forms

at least, there must be 6 different forms, every form is worth 5 points

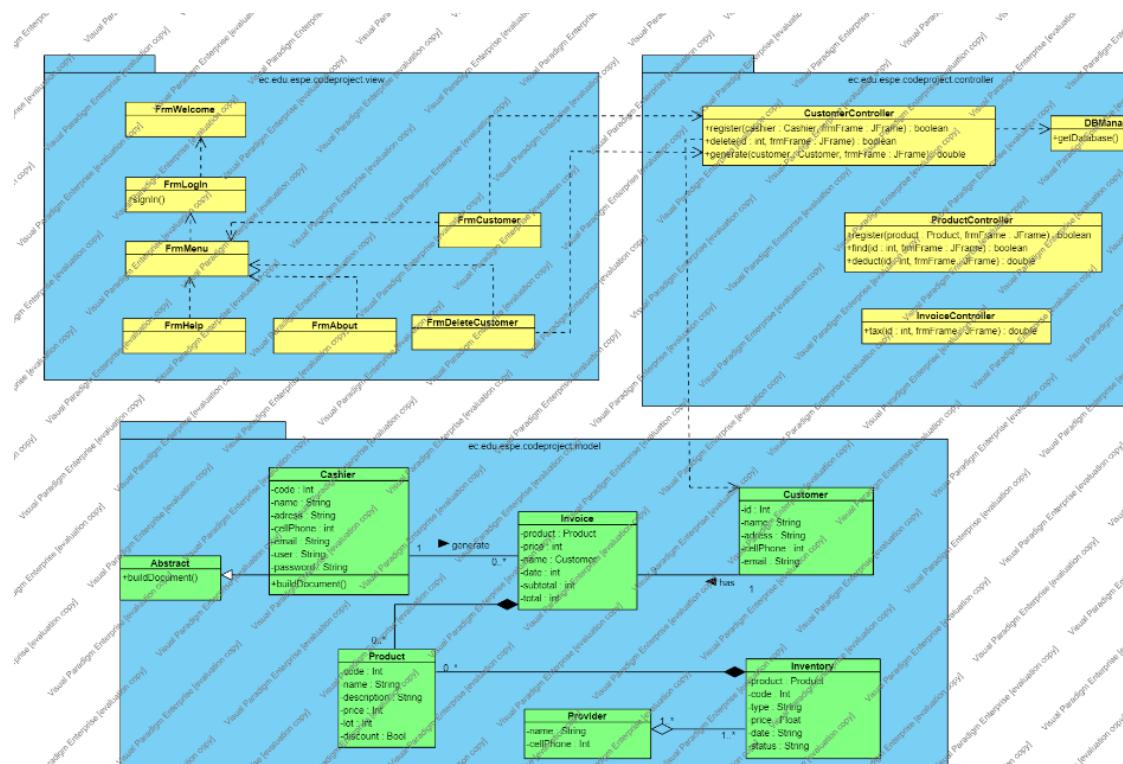
- Login Form
 - 5 Forms for Information (CRUD operations, 4 business rules)
- Every form

- Every input (JTextField) is validated
- There are different types of inputs (not everything is a JTextField) -1 for each JTextField
- Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

every mistake: -1

less than 5 forms -> /50%

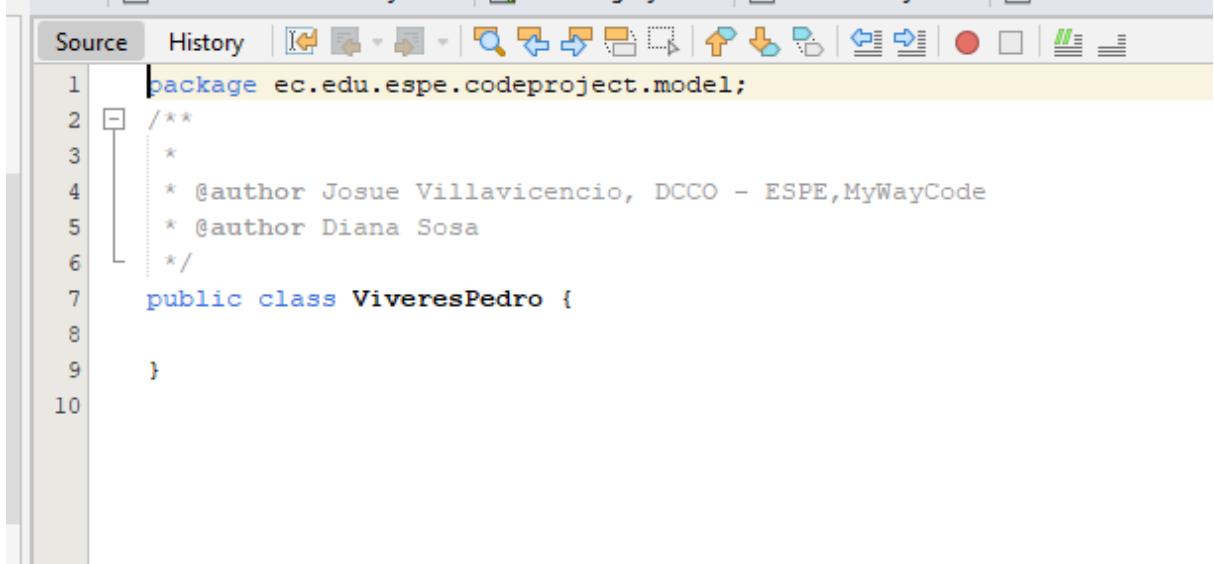
Unit Test



Clean Code

```
private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {  
    String Username="oop";  
    String Password="1234";  
  
    String Pass=new String(this.Password.getPassword());  
  
    if(txtUsername.getText().equals(Username)&&Pass.equals(Password)){
```

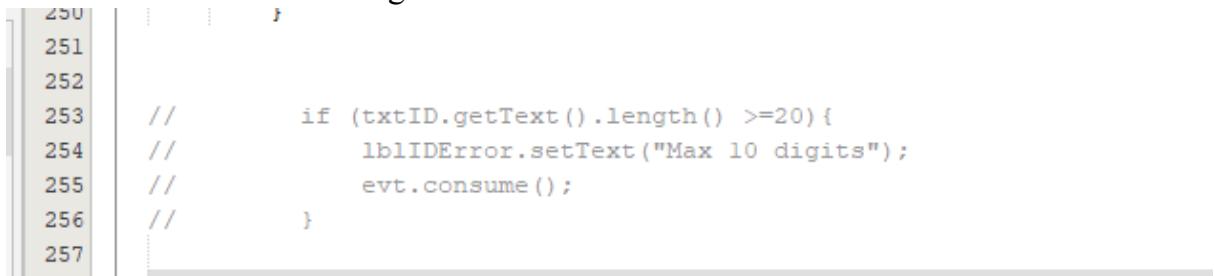
Wrong name of button variables



A screenshot of a Java code editor showing a single file named "ViveresPedro.java". The code defines a class with the same name, containing a package declaration and a class definition. The code is mostly blank, with some comments and a closing brace.

```
1 package ec.edu.espe.codeproject.model;  
2  
3   
4 * @author Josue Villavicencio, DCCO - ESPE, MyWayCode  
5 * @author Diana Sosa  
6 */  
7 public class ViveresPedro {  
8  
9 }  
10
```

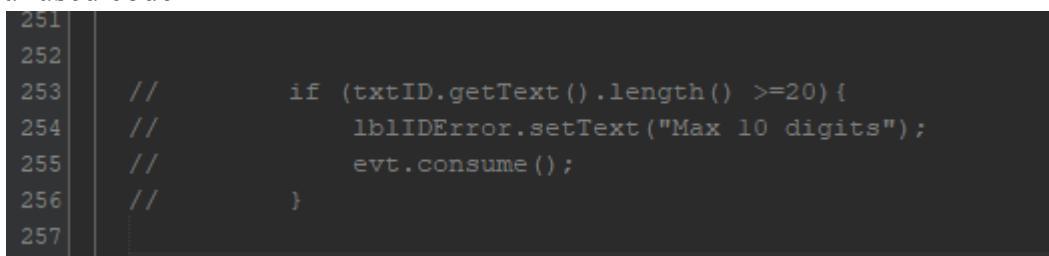
classes created without using



A screenshot of a Java code editor showing a section of code with several lines starting with double slashes (//), indicating they are comments or unused code. The code is part of a larger method, likely for validating a text input field.

```
250 ...  
251 ...  
252 ...  
253 // if (txtID.getText().length() >=20){  
254 // lblIDError.setText("Max 10 digits");  
255 // evt.consume();  
256 // }  
257 ...
```

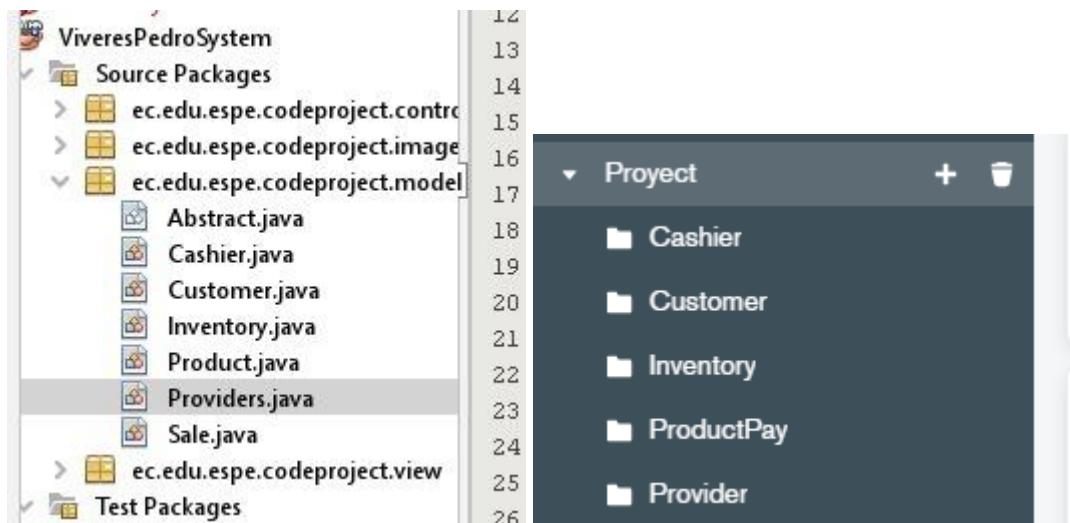
unused code



A screenshot of a Java code editor showing a block of code that is entirely commented out with double slashes (//). The code is intended to validate a text input field, similar to the code above it.

```
251 ...  
252 ...  
253 // if (txtID.getText().length() >=20){  
254 // lblIDError.setText("Max 10 digits");  
255 // evt.consume();  
256 // }  
257 ...
```

MONGO



Authentication(With mongoDB not with burned data)

```

141 }
142 
143     private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
144         String username="oop";
145         String password="1234";
146 
147         String Pass=new String(this.Password.getPassword());
148 
149         if(txtUsername.getText().equals(username)&&Pass.equals(password)) {
150 
151             FrmMenu EP=new FrmMenu();
152             EP.setVisible(true);
153             dispose();
154 
155         }
156         else{
157             JOptionPane.showMessageDialog(this,"Incorrect Username / Password");
158         }
159     }
160 }

Document doc;
    doc= createDBObject(cashier);
MongoDatabase userDB = DBManager.getDatabase();
MongoCollection<Document> col = userDB.getCollection("Cashier");

```

 - X

Provider

Id:

Full Name:

Company:

Cellphone:

Email:

register **exit**