“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%
Avegarge /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 JTextFields) -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.

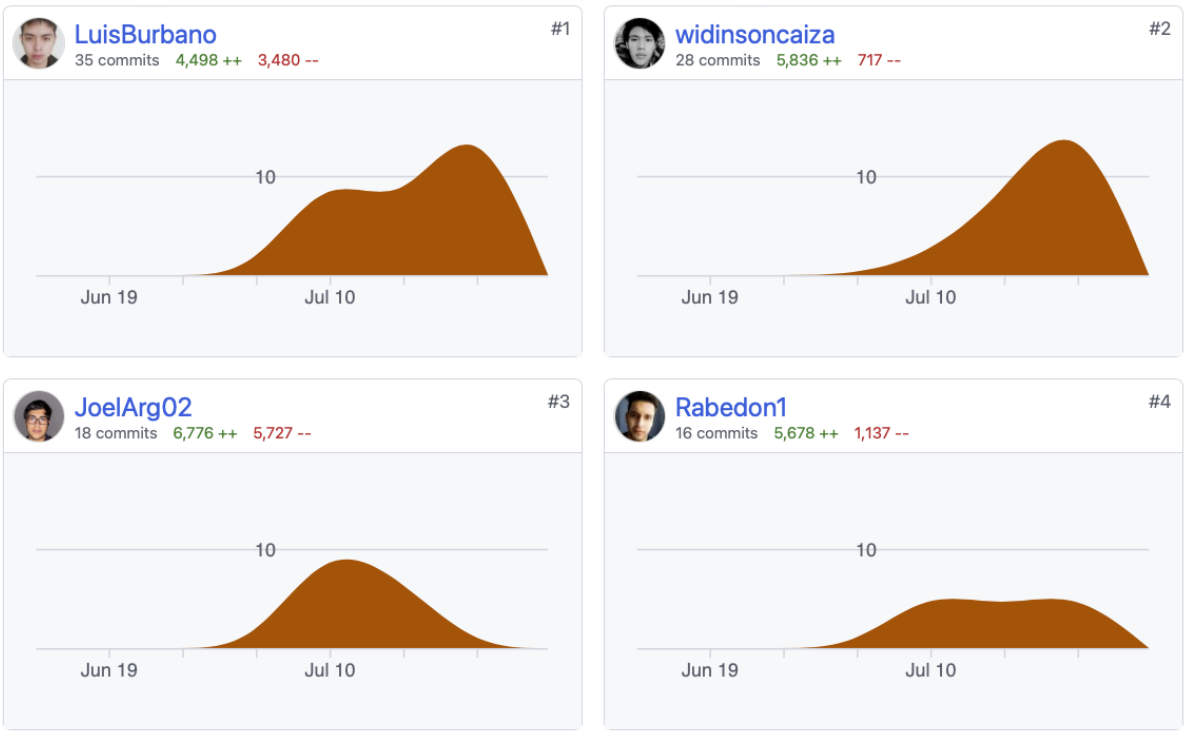
- review that the unit tests are not empty
- there must be at least one unit test that is failing.
- If all unit tests are OK, then it is half the grade. It means that the tests are not good enough
- every unit test is worth 0.1

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

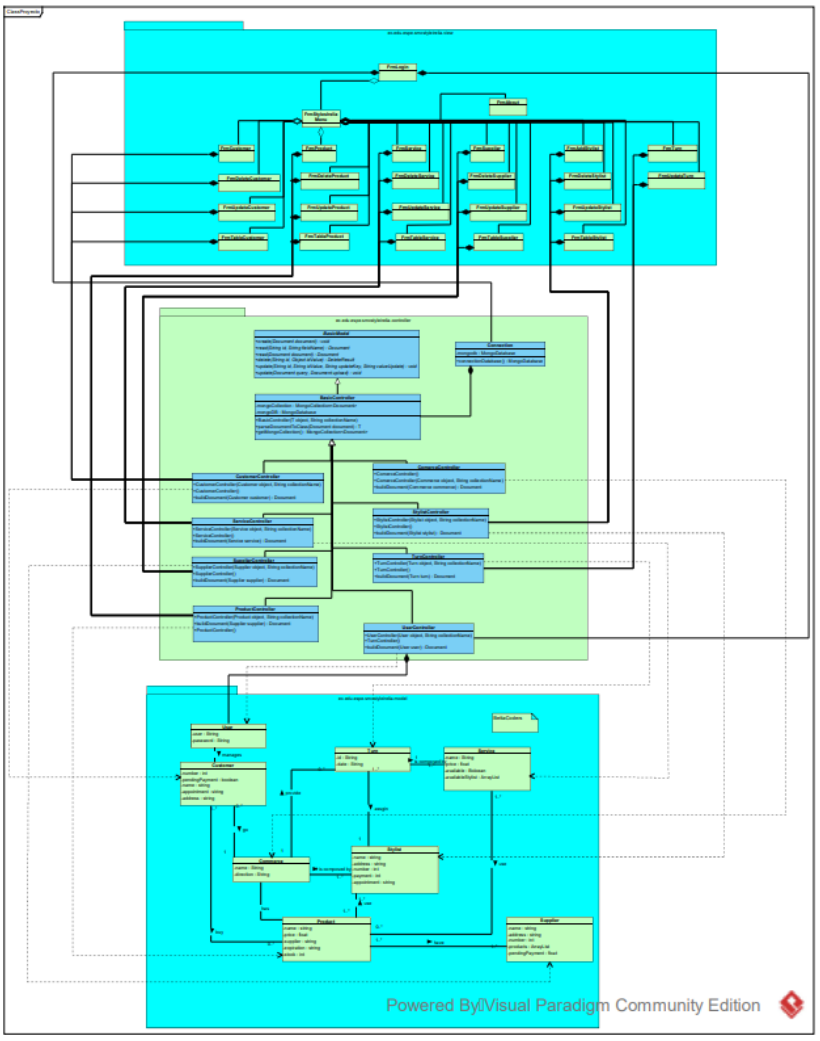
2. The system shall give List of services, prices the most contracted

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

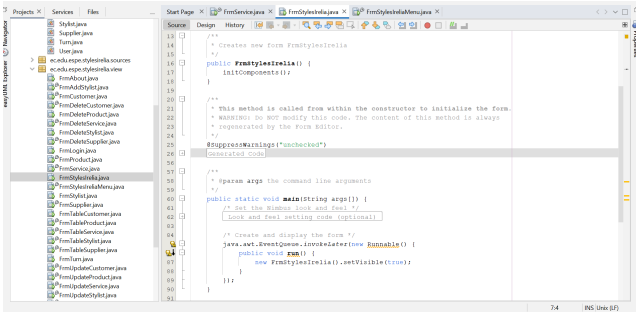
2. UML Class Diagram
   - Classes -2
   - Attributes -2/-0.5
   - Methods -2/-0.5
   - Relations -2
   - Abstract Classes/Interfaces 2/2
   - Abstract methods 2/2
   - Polymorphism (overridden methods) 2/2
   - Inheritance 2/2
   - MVC -> packages model (POJO classes), view(GUI), controller (Abstract 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)



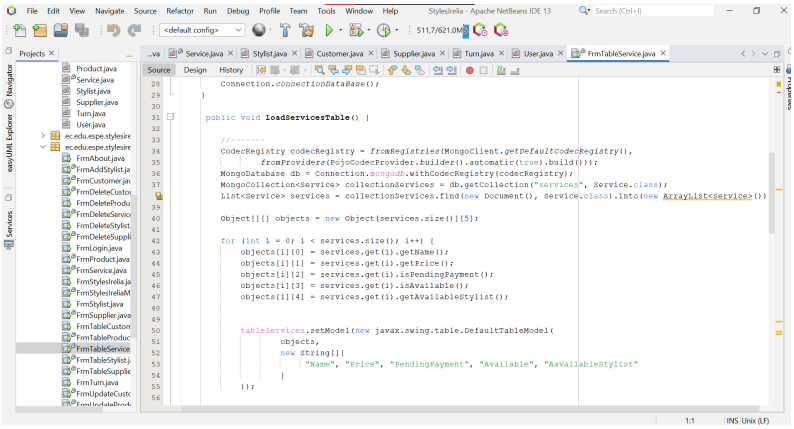
Clean Code

Pitfalls/Code Smells (0.1)

frame sin usar -0.1



metodo en el view y no en el controller -0.1



```
Connection.connectionDataBase();
}

public void loadServicesTable() {

    //-------
    CodecRegistry codecRegistry = fromRegistries(MongoClient.getDefaultCodecRegistry(),
            fromProviders(PojoCodecProvider.builder().automatic(true).build()));
    MongoDatabase db = Connection.mongodb.withCodecRegistry(codecRegistry);
    MongoCollection<Service> collectionServices = db.getCollection("services", Service.class);
    List<Service> services = collectionServices.find(new Document(), Service.class).into(new ArrayList<Service>());

    Object[][] objects = new Object[services.size()][5];

    for (int i = 0; i < services.size(); i++) {
        objects[i][0] = services.get(i).getName();
        objects[i][1] = services.get(i).getPrice();
        objects[i][2] = services.get(i).isPendingPayment();
        objects[i][3] = services.get(i).isAvailable();
        objects[i][4] = services.get(i).getAvailableStylist();


        tableServices.setModel(new javax.swing.table.DefaultTableModel(
                objects,
                new String[]{
                    "Name", "Price", "PendingPayment", "Available", "AavailableStylist"
                }
        ));
```

```
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:
}
```

```
    }

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

Variable no usada

```
String name;
int number;
boolean pendigPayment;
String appointment;
```

Variable pendigPayment is not used

Mala declaración

Variable customer is not used

```
Customer customer = new Customer(identificationCard, name, number, false, appointment, address);
```

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

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

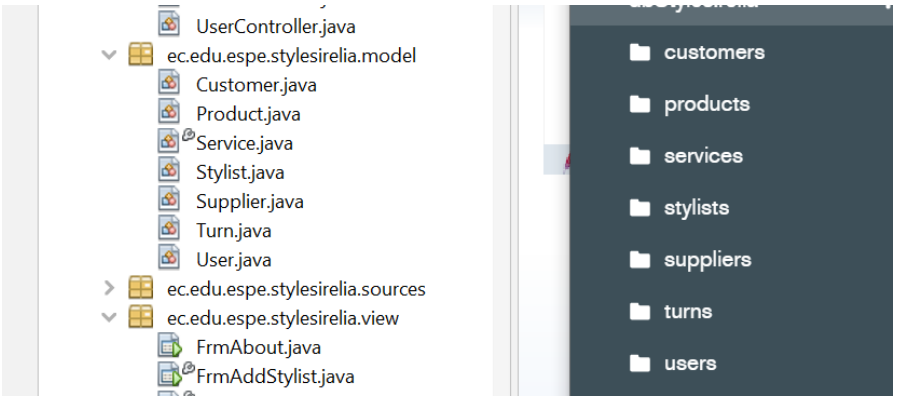
Comentarios

```
userController = new UserController();
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
String path = System.getProperty("user.dir");
//        ImageIcon img = new ImageIcon(path +"\\src\\ec\\edu\\espe\\stylesirelia\\sources\\logoBettaCoders.png");
//        this.setIconImage(img.getImage());
this.setTitle("Login Window");
```

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

DB (MOngoDb Atlas)

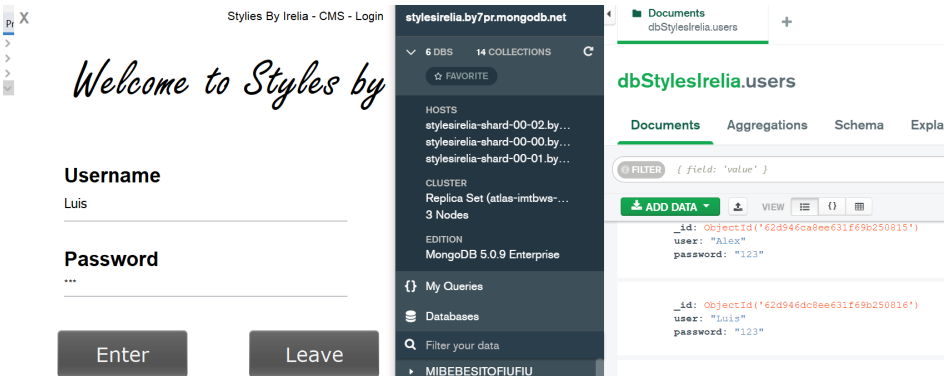
Database Name	-0.5
Collections vs Model classes	-0.5
attributes (collections) vs attributes (classes)	-0.5

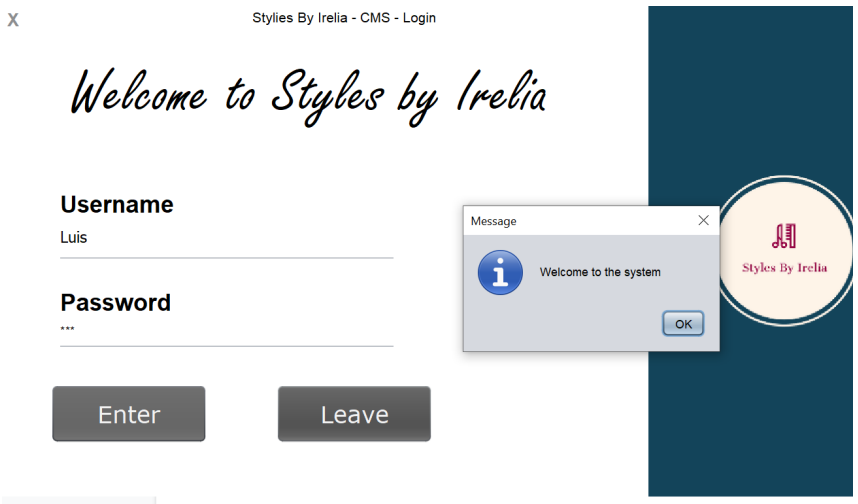


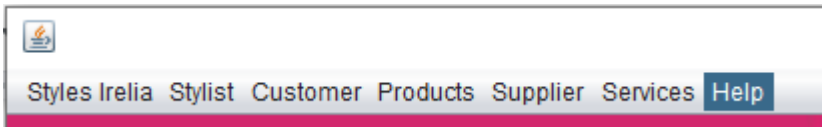
GUI

Splash Window

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





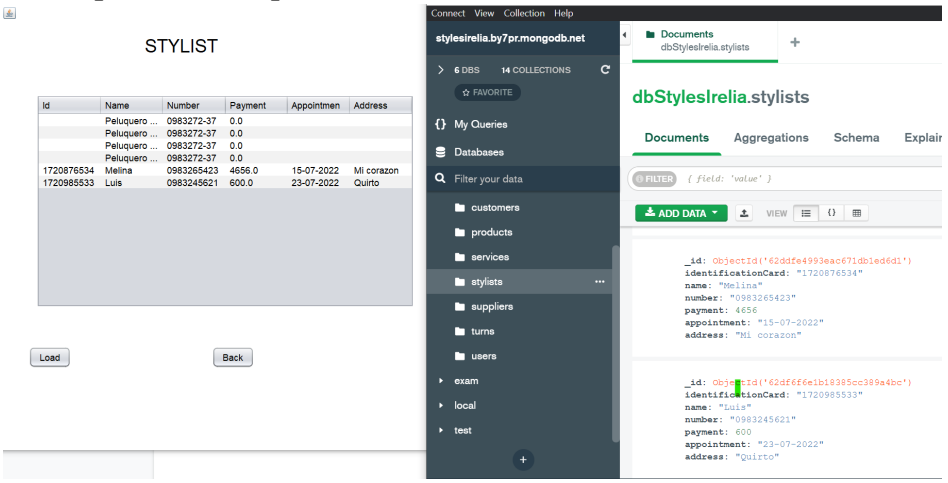


Menú Help no hace nada

Tables:

data come from DB 5/5

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 JTextFields) -1 for each JTextField

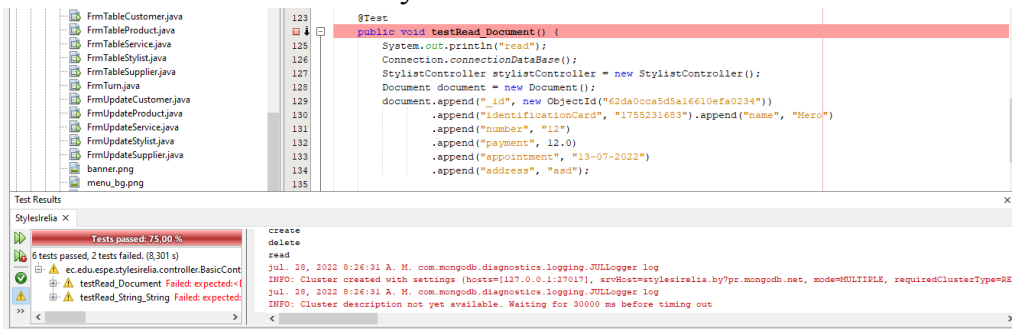
- Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

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

8. Unit Test 0.8/10

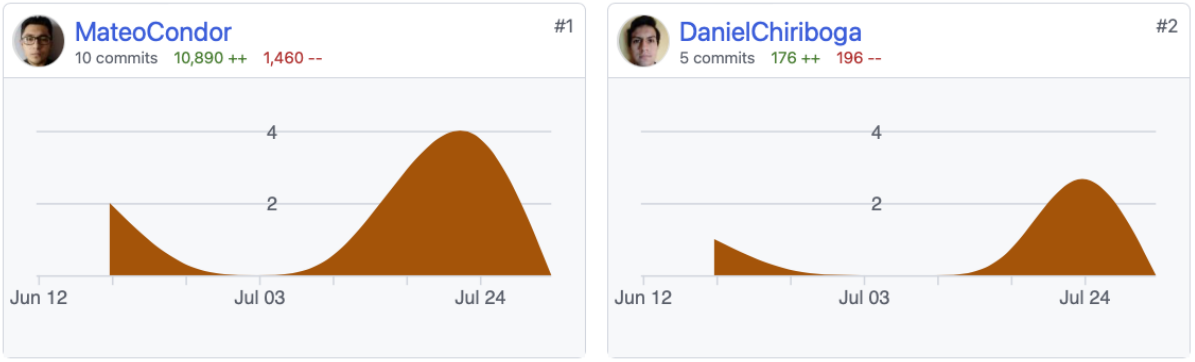
At least 100 unit tests.

- 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



TEAM 02 DAMAGE INSPECTOR: TEAM 03 Syntax Error (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 Pagos al Proveedor, Gastos tales como servicios básicos, etc.

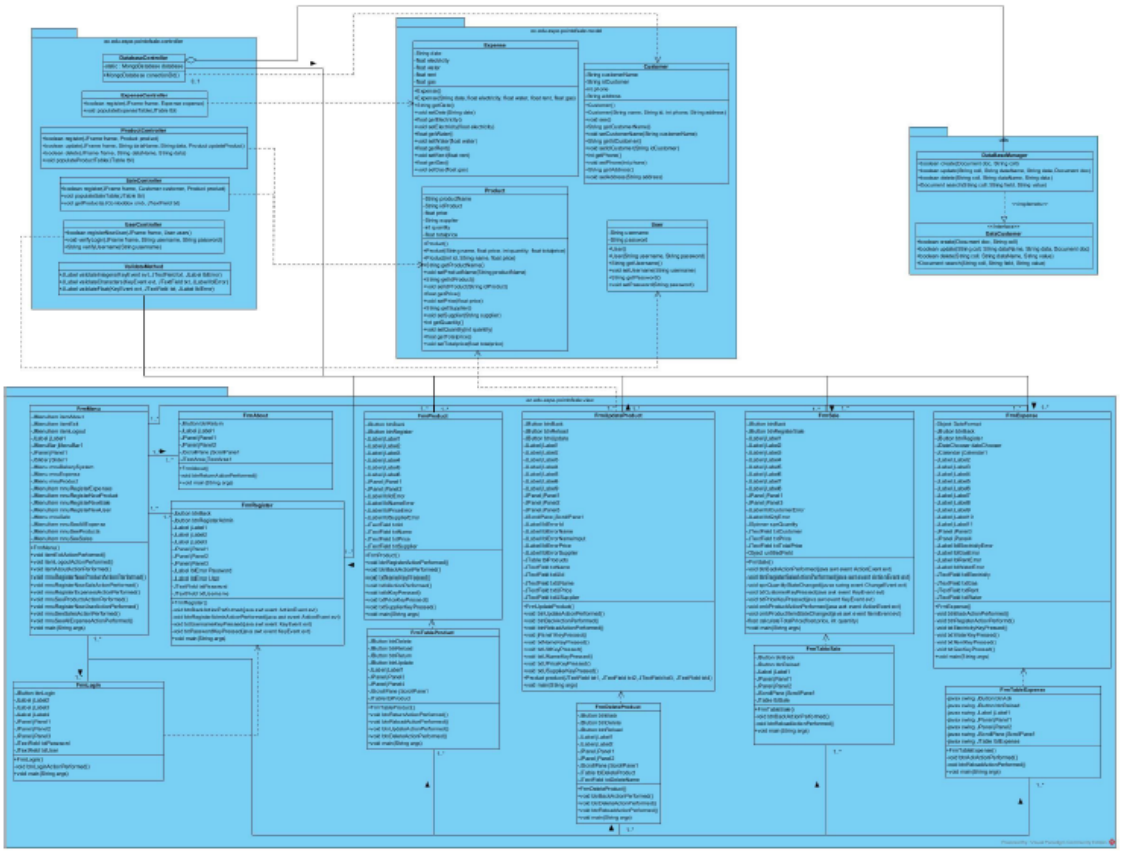
El usuario necesita una sección para la Caja del sistema, donde se registrarán la cantidad de dinero de los ingresos del negocio, los retiros que se hacen durante el día y los Pagos y Gastos del sistema.

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 int getPhone() {
    return phone;
}

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

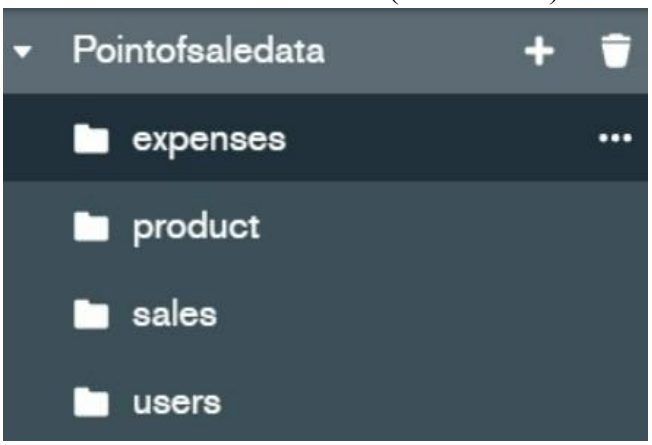
```
public float getGas() {
    return gas;
}

/**
 * @param gas the gas to set
 */
public void setGas(float gas) {
    this.gas = gas;
}
```

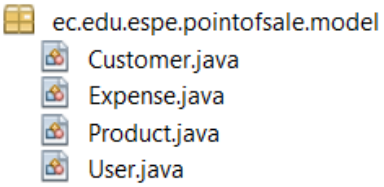
4. Db (MongoDb Atlas)

Database Name

Collection vs Model classes

attributes (collections) vs attributes (classes)



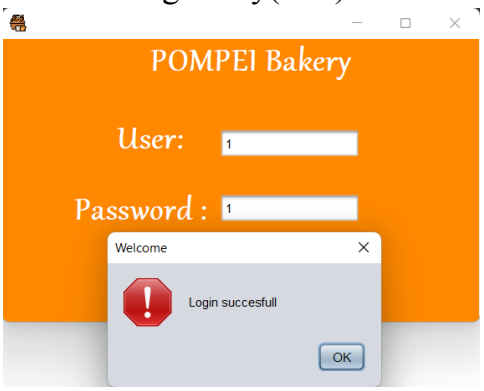


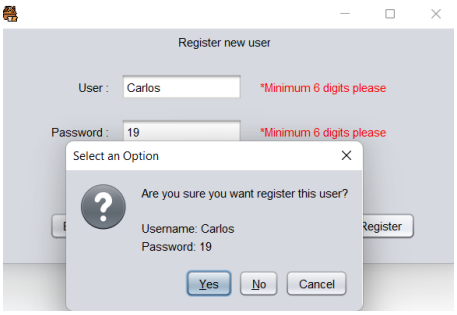
Observation:

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"
```

Poduct 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

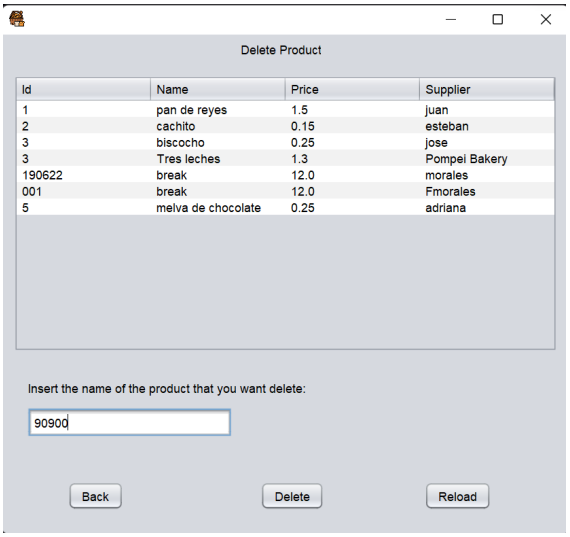
-5 Forms for Information (CRUD operations, 4 business rules)

Every Form

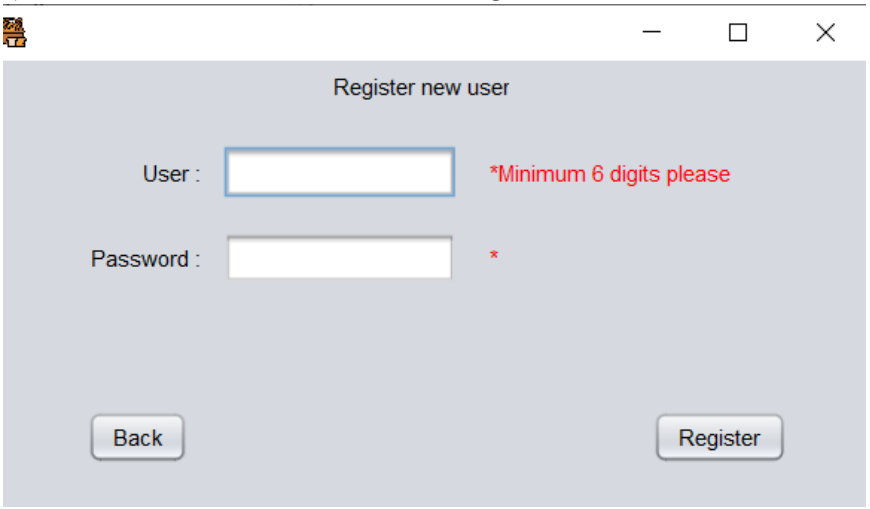
- Every input (JTextField) is validated
- There are different types of inputs (not everything is a JTextFields) -1 for each JTextField
- Buttons' events, no code is here for CRUD operations or business rules, i.e., these functions will call some controller method, and will use some model class.

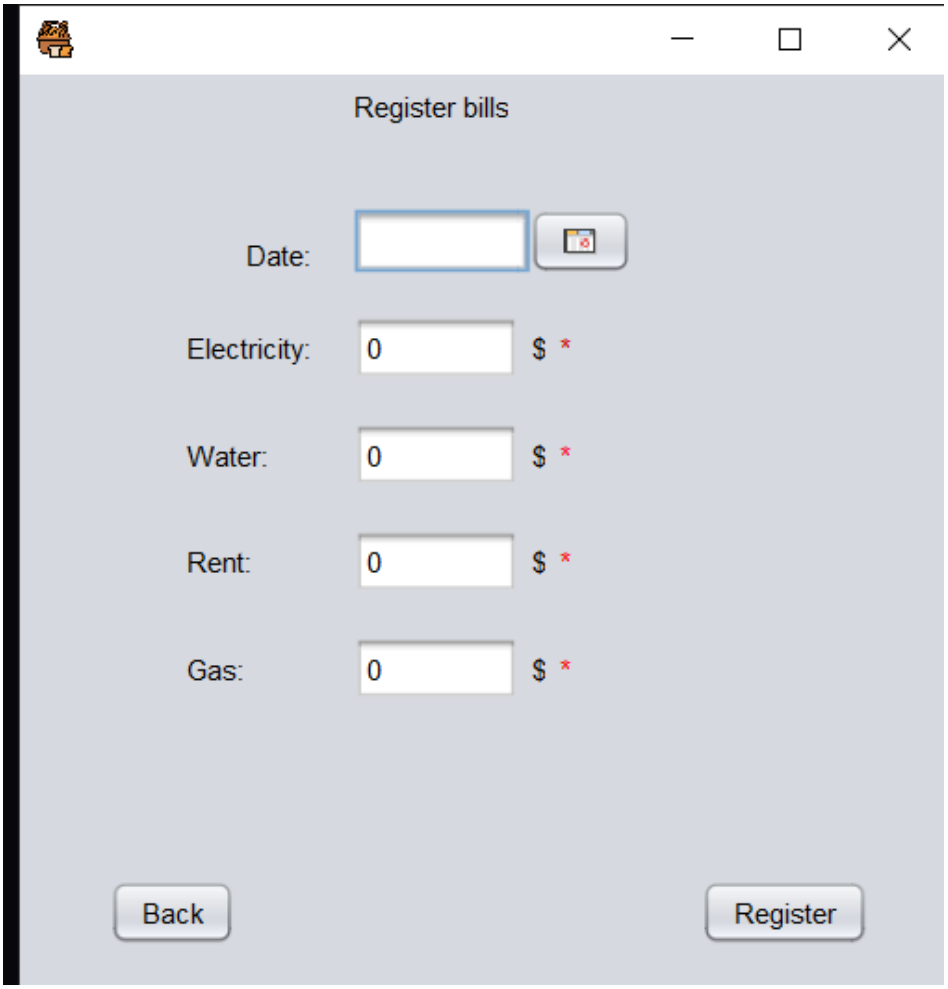
every mistake: -1
less than 5 forms -> /50%

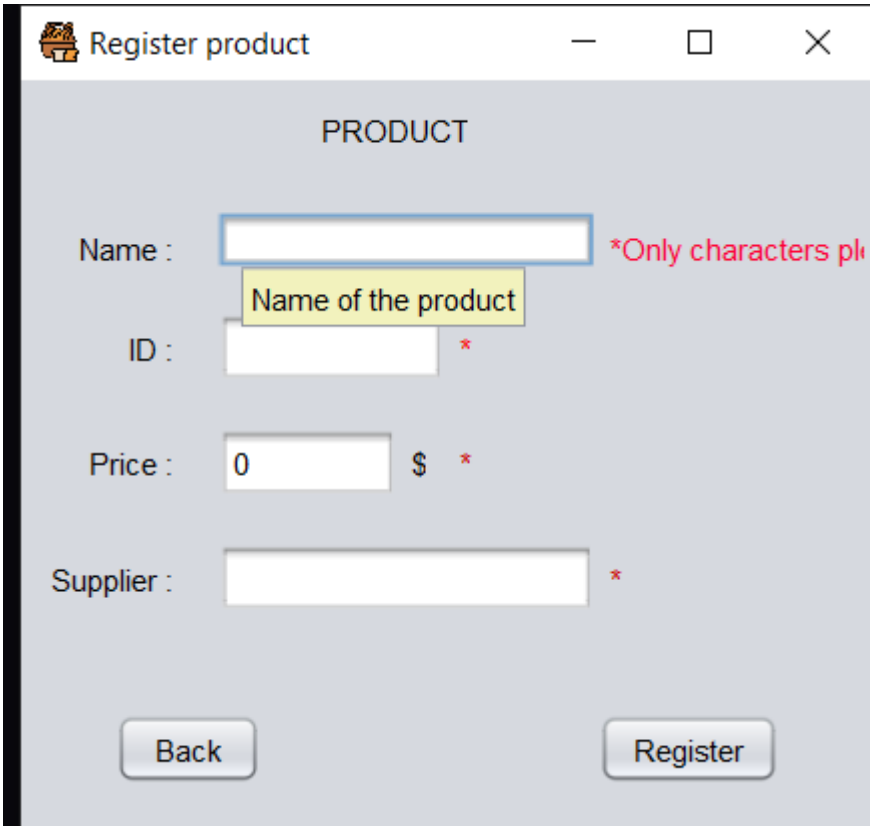
a) Enter the name of the product without validating
b) find the name of the product if you put letters



- 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



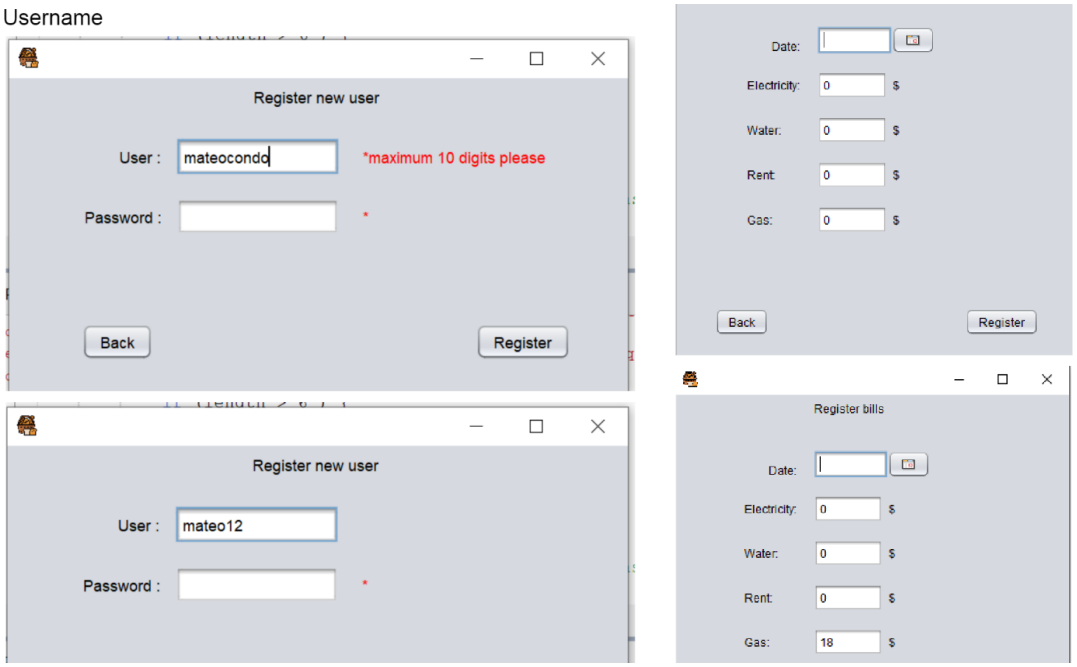

Register bills
Date:
Electricity:
0
$ *
Water:
0
$ *
Rent:
0
$ *
Gas:
0
$ *
Back
Register


Register product
PRODUCT
Name :
*Only characters pl
Name of the product
ID :
*
Price :
0
$ *
Supplier :
*
Back
Register

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

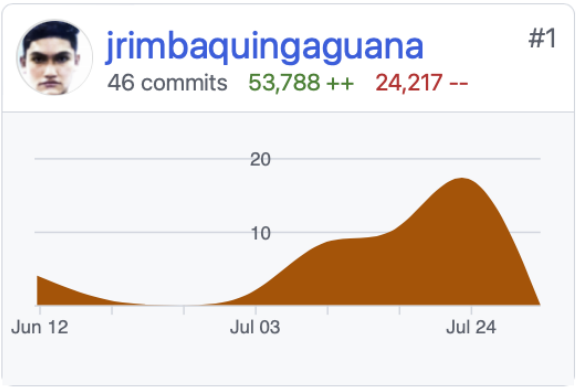


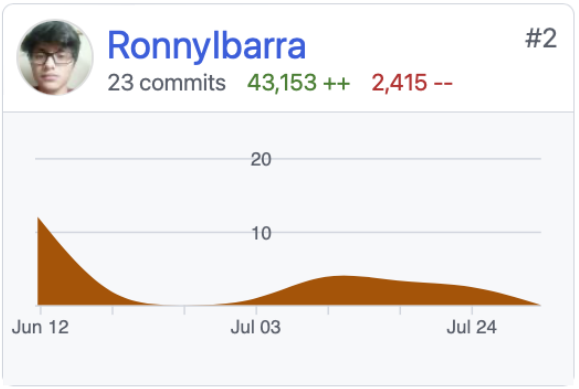
TEAM 03 Syntax Error **INSPECTOR TEAM 04 "GADC.MSI"**
(César Loor)

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%



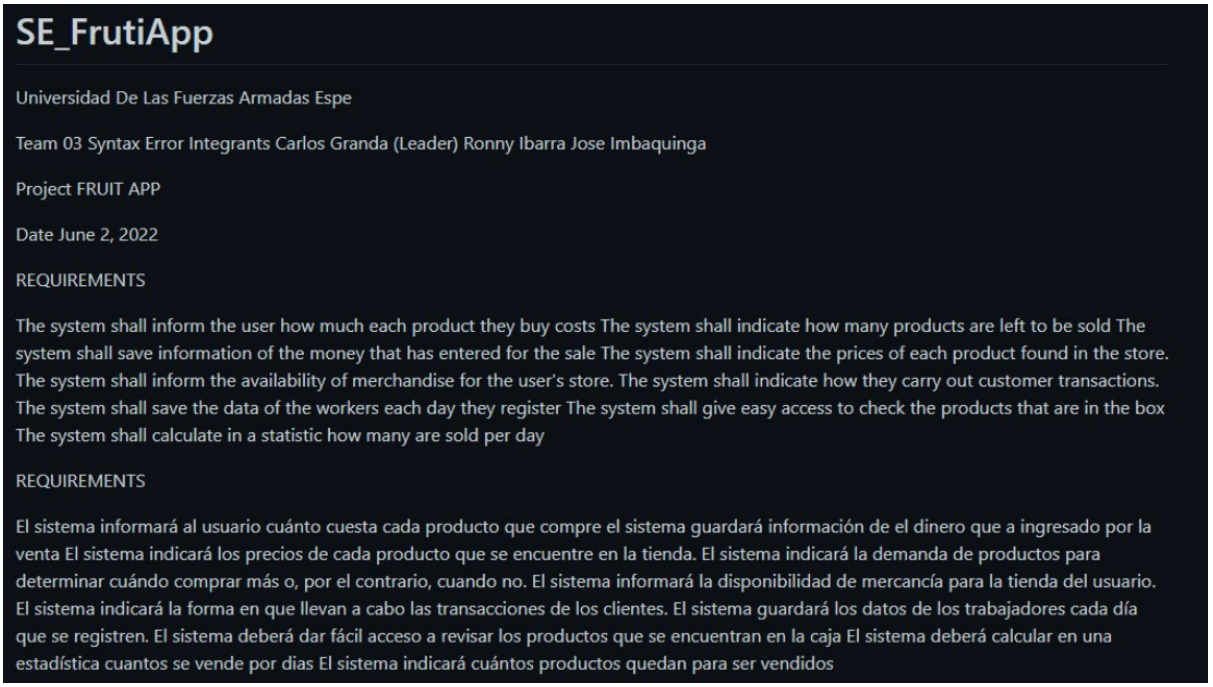


Rubric:

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

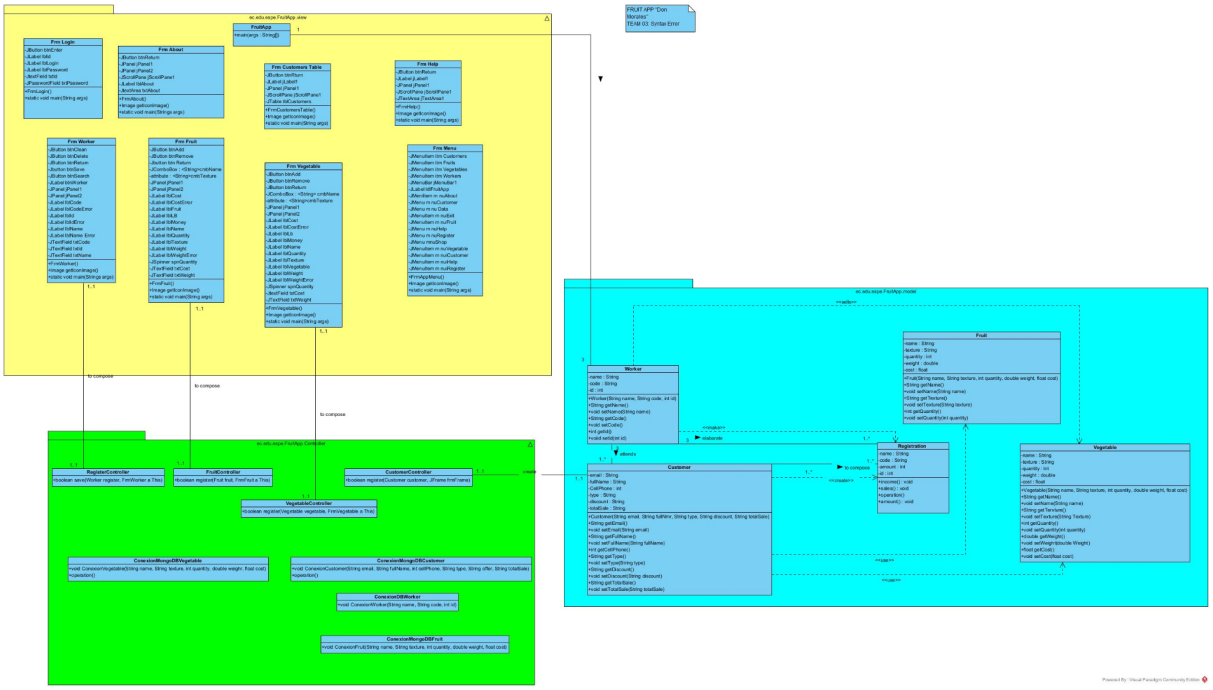
Requirements (features/Items)



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
import com.mongodb.MongoClient;
import com.mongodb.client.MongoDatabase;
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
    import javax.swing.JFrame;
    import static org.bson.codecs.configuration.CodecRegistries.fromProviders;
    import static org.bson.codecs.configuration.CodecRegistries.fromRegistries;
    import org.bson.codecs.configuration.CodecRegistry;
    import org.bson.codecs.pojo.PojoCodecProvider;
    import com.mongodb.MongoClient;
    import com.mongodb.client.MongoCollection;
    import ec.edu.espe.fruitApp.model.Customer;
    import java.util.ArrayList;
    import java.util.List;
22  import javax.swing.JFrame;
    import javax.swing.JOptionPane;
    import javax.swing.JTable;
    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();
```

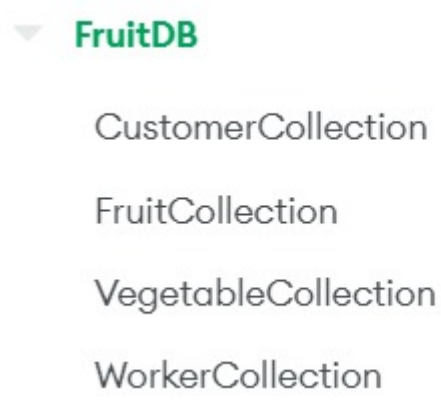
d) Death Code

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

}
```

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

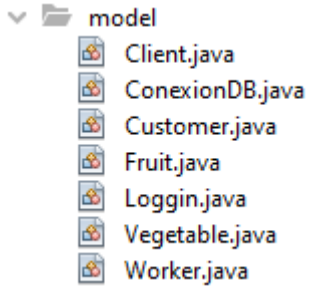
Mongo Atlas

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



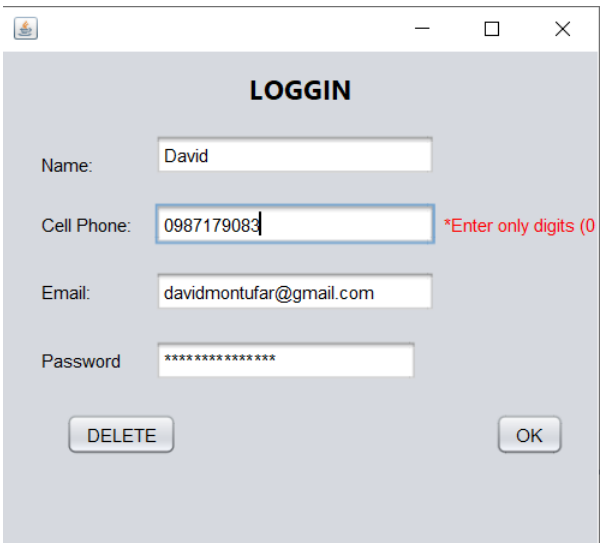
Collection vs Model classes

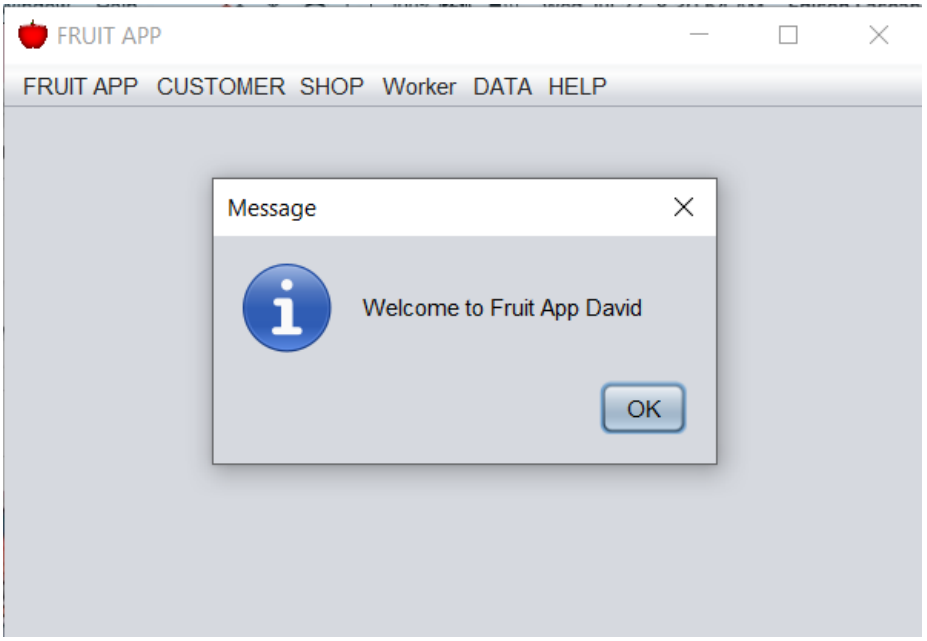
Client collection is missing

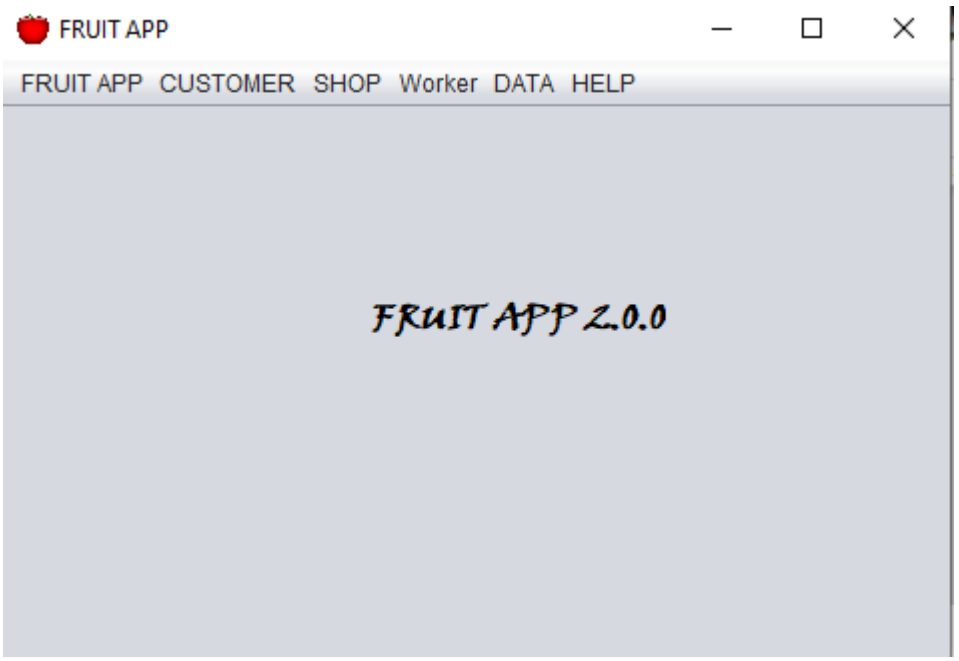


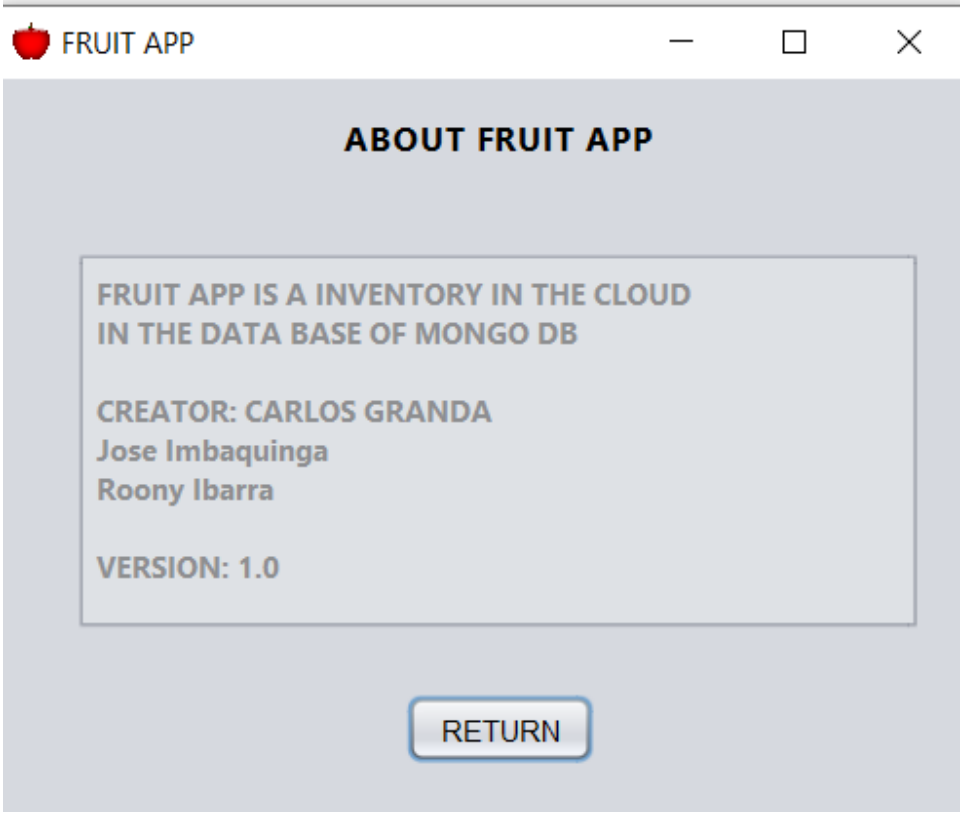
GUI

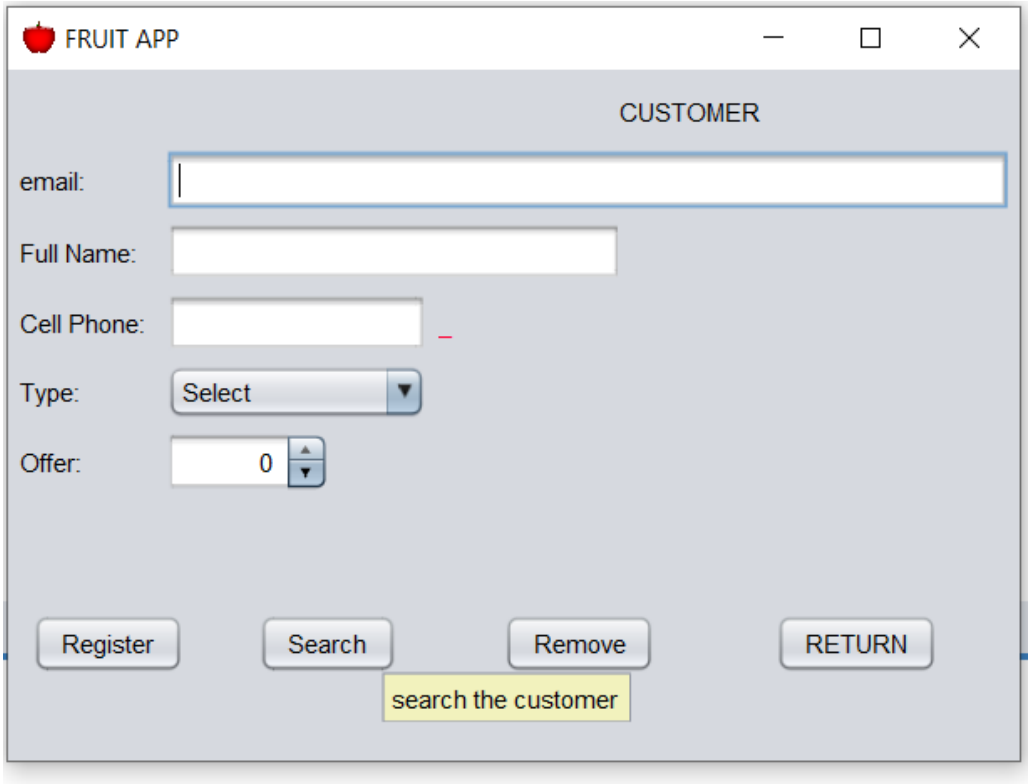
Splash Window			
Authentication	2/2	(it should work)	
Application (Menu)	1.6/2	(requirements)	(-0.2)
Forms	5/6	(requirements)	Navigability -0.2


LOGGIN
Name:
David
Cell Phone:
0987179083
*Enter only digits (0
Email:
davidmontufar@gmail.com
Password
DELETE
OK


FRUIT APP
FRUIT APP CUSTOMER SHOP Worker DATA HELP
Message
Welcome to Fruit App David
OK


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 JTextFields) -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.

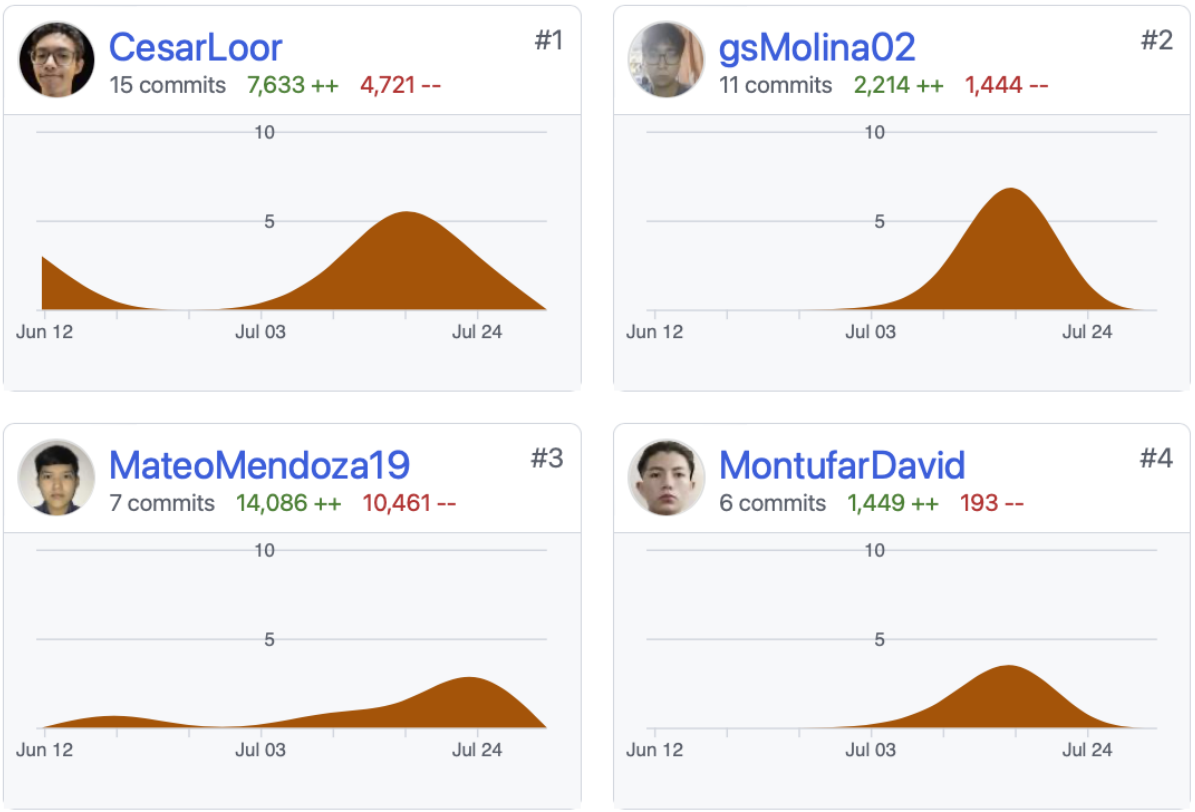
  every mistake: -1
  less than 5 forms -> /50%

Unit Test

TEAM 04 "GADC.MSI" **INSPECTOR: TEAM 05 DEES Dev's (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%

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

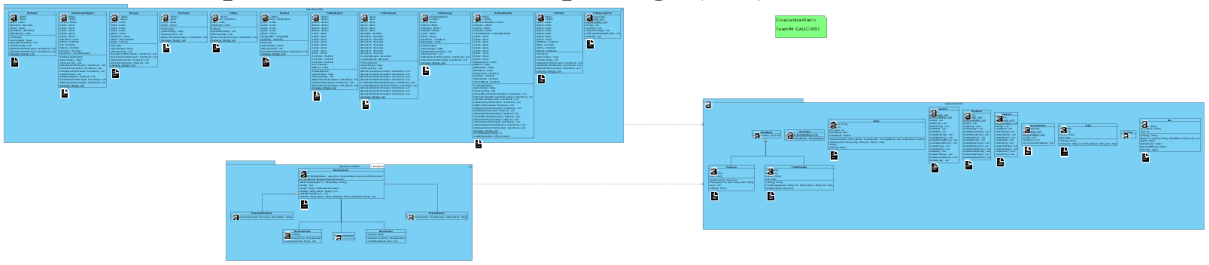
Requirements

- 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 marianas, 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 lblIdErrorKeyPressed(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;
import sun.security.krb5.internal.Ticket;
```

d) Spanish -0.1

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

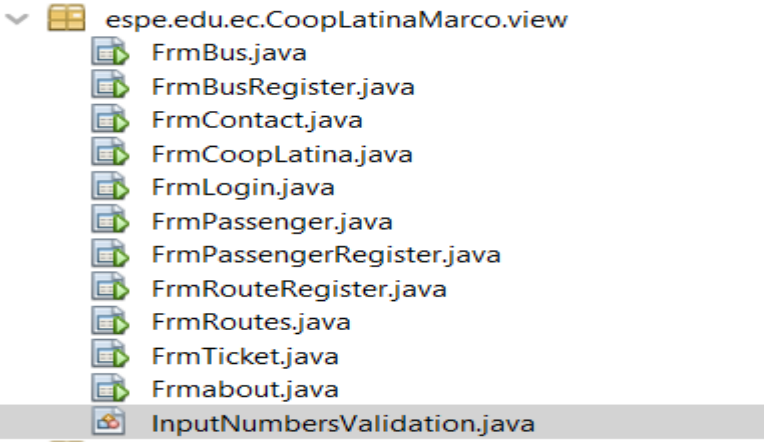
e) Redundant or Meaningless Comments again

```
//route.setPriceOfRoute(JTValidation(txtPriceOfRoute.getText()));
route.setAvalibleRoute(availeRoute(cmbAvaibleRoute.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();
}
```

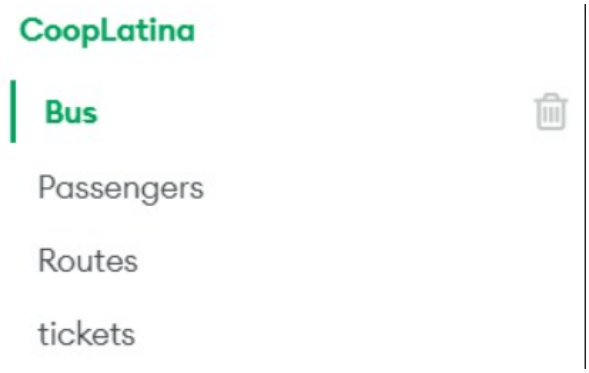
g) A java class in the view package -0.1



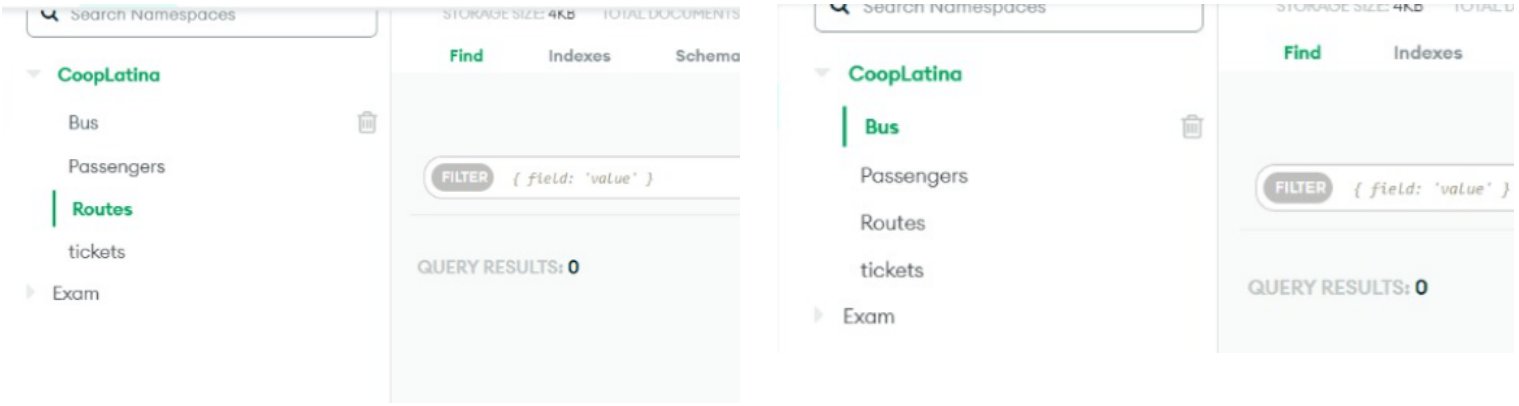
MONGODB ATLAS:

Database Name -0.5

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



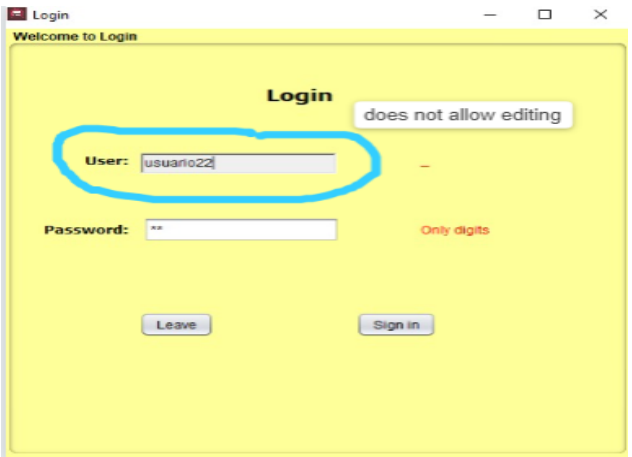
Collections Routes and tickets are voids (-0.5)



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:

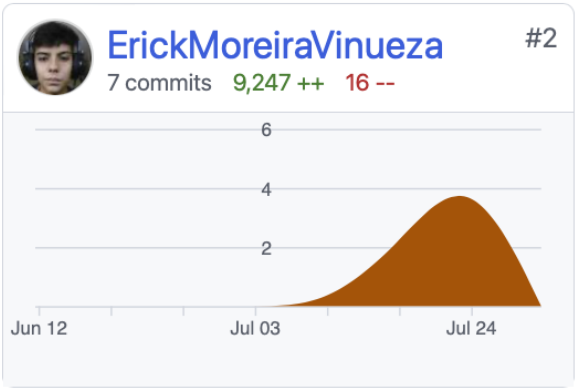
at least, there must be 6 different forms, every form is worth 5 points

- 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 JTextFields) -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 **INSPECTOR: TEAM 6 Codex++ (Juan Quimbiulco)**
GitHub: https://github.com/stevenpozo/DEES-developers
NameProject: LeatherFlowerSystem

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



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

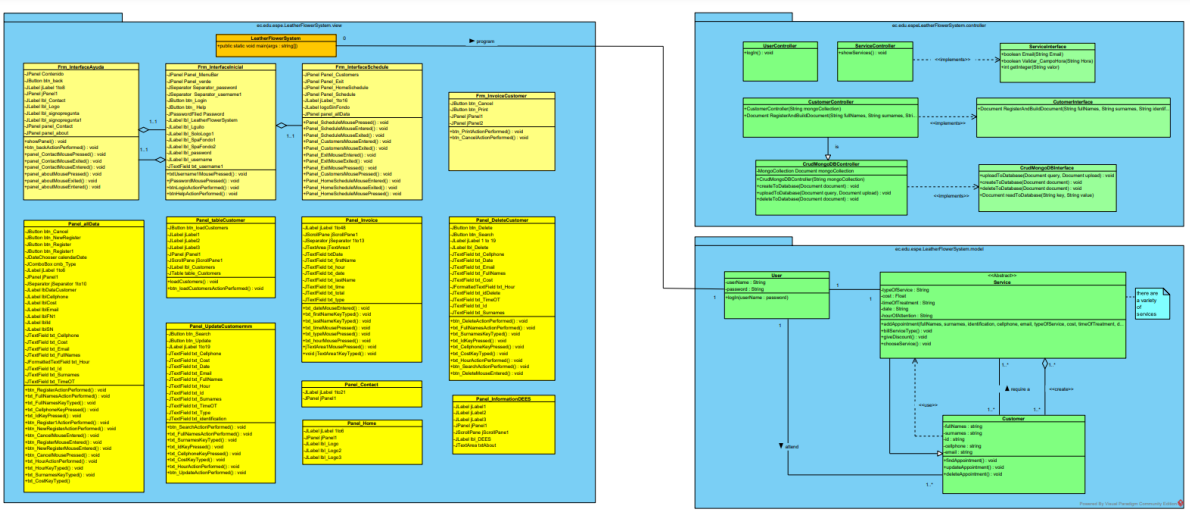
- Log in
- Save customer data
- Show the amount to be charged for the service provided
- Edit customer data
- Delete customer data

HU Framework Matrix

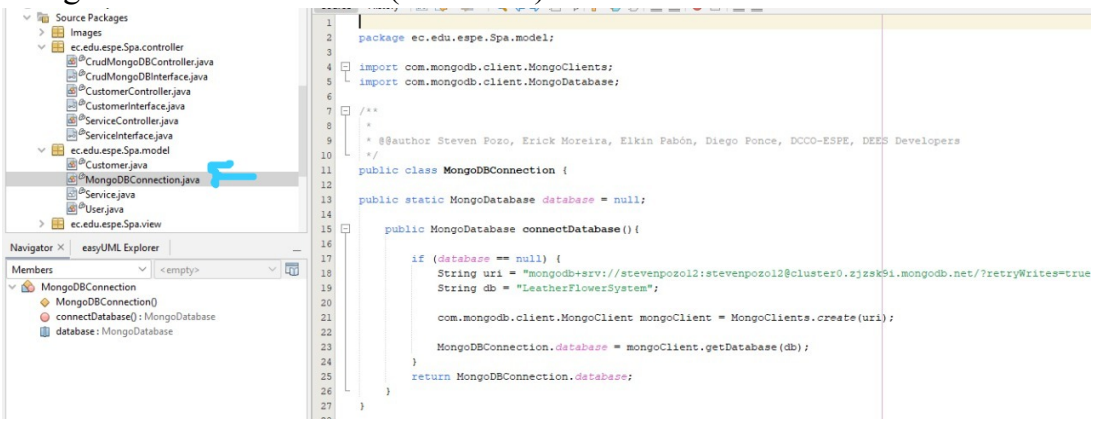
ITEM	PROBLEM	WHAT (REQUEST)	WHAT FOR (SOLUTION)	FOR WHO (USUARIO)	HOW (DESCRIPTION OF TASKS)	MADE BY (PROG. RESP.)	HOW LONG (HOURS)	DATE OF DELIVERY	PRIORITY	STATUS	TEST (HOW TO VERIFY)	COMENTARIES
REQ001	Options menu	Log in	So you can schedule, postpone or cancel appointments	Customer (owner)	In the 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 Pozo	In process	August	hight	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, ID number and email of the client that will be requested.	Steven Pozo	In process	August	hight	In process	Once the data is saved, the appointment already scheduled can be viewed later in a section.	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 or you can pay in installments, whatever the method, at the beginning of this you must invoice the amount that the service that will be provided acquires.	Elkin Pabón - Diego Ponce	In process	August	hight	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	hight	In process	Once the change is saved, the day already differed can be viewed later in a section 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	hight	In process	Once the data is delete, 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)



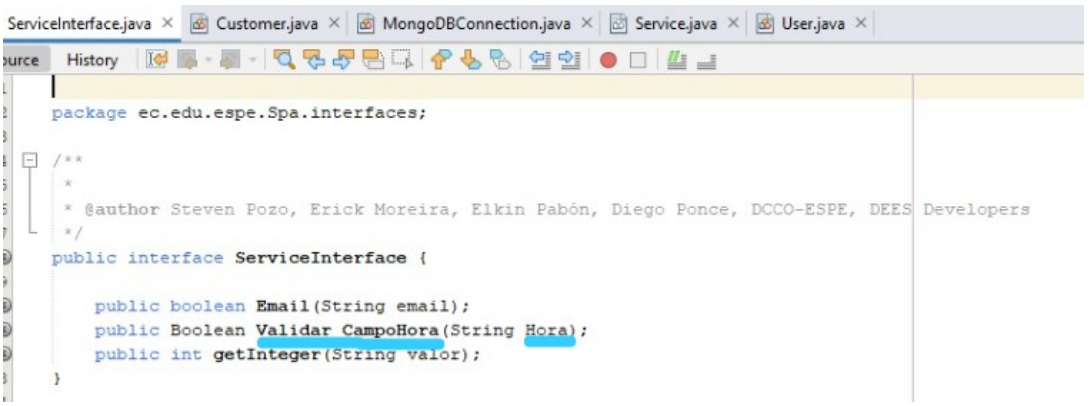
Spanish

```
@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;
}
}
```

```java
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.

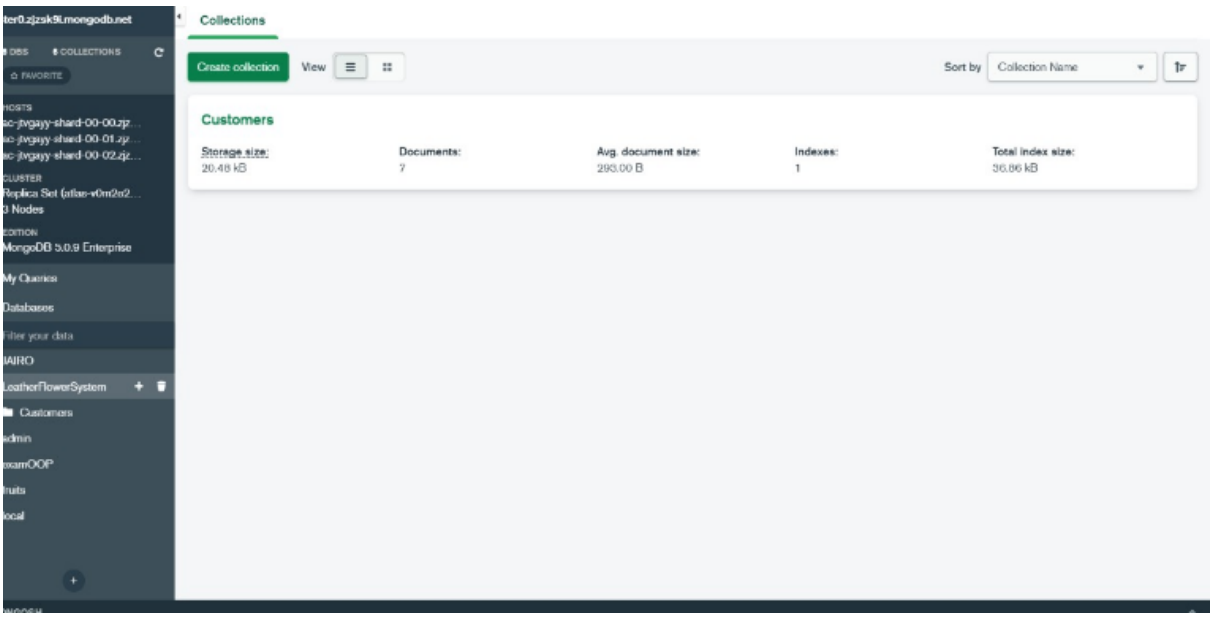
```java
protected String fullNames;
protected String surnames;
protected String identification;
protected String cellphone;
protected String email;
```



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)	(-0.2)
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 FormQuilumbaquin 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 JTextFields) -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%

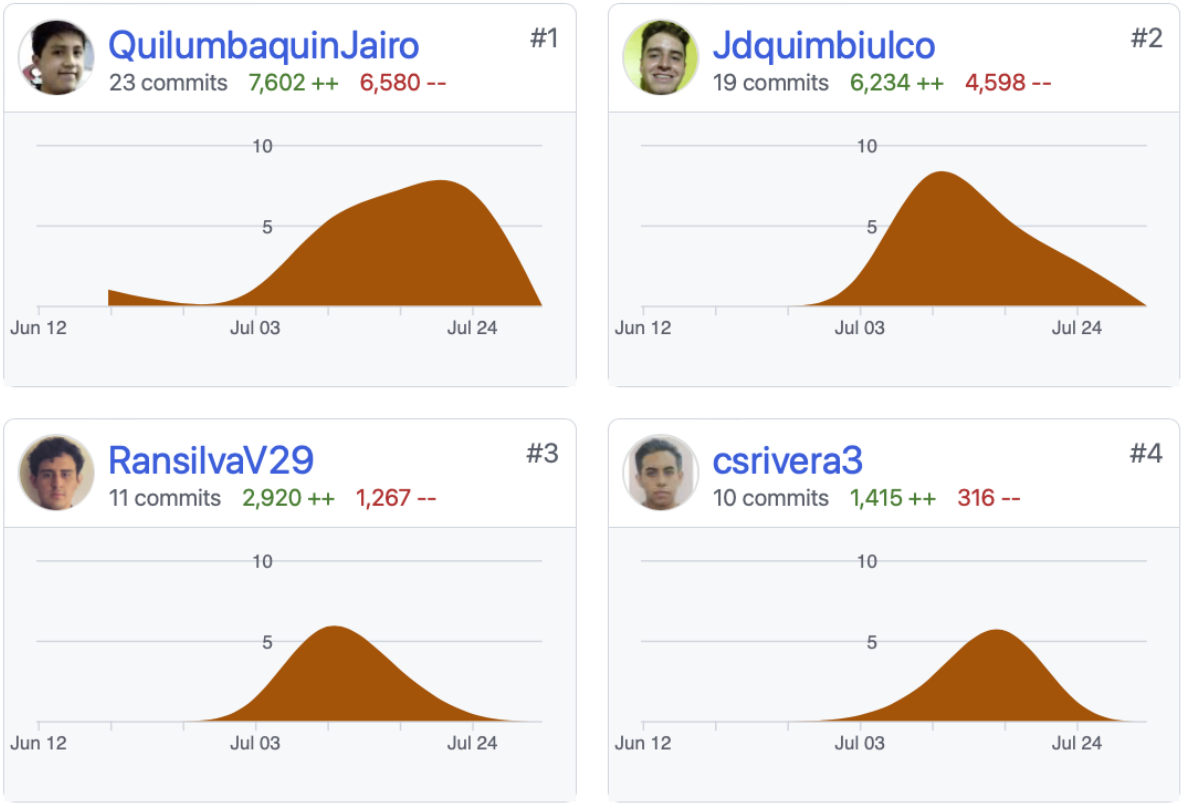
Unit Test 0.08

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) **INSPECTOR: TEAM 07 MyWayCode**

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



Github:https://github.com/Jdquimbiulco/TC_BooKify.git
Project name: BookiFiy

21.

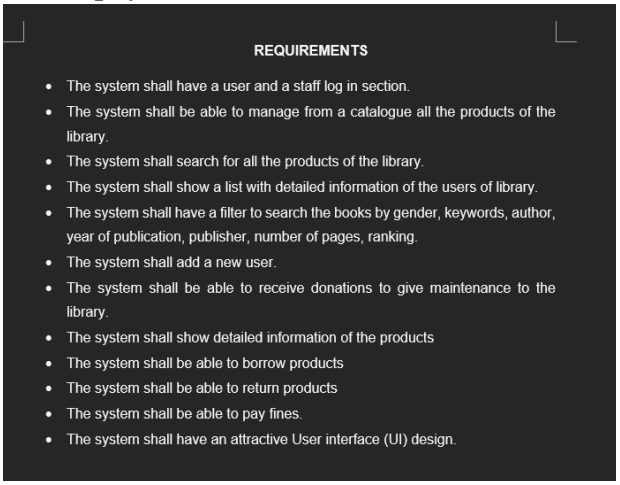
Rubric:

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

TOTAL	62.1/100
GitHub	/100
Avegarge	/100

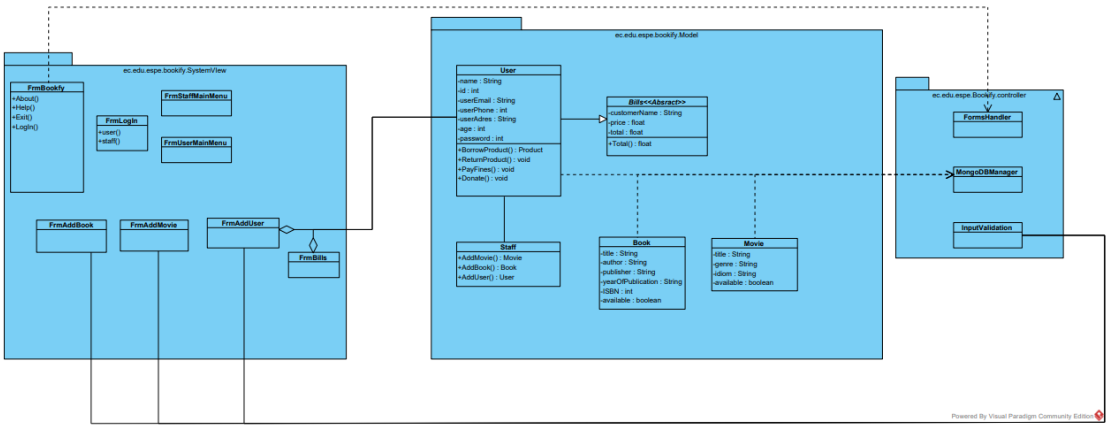
1. Requirements (Features/items)
   - ★ log in section
   - ★ user list
   - ★ search product
   - ★ delete information
   - ★ product catalog
   - ★ user information
   - ★ product information
   - ★ book filter
   - ★ payment of fines



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 int StaffPasword;
    private boolean StaffBlackList;
```

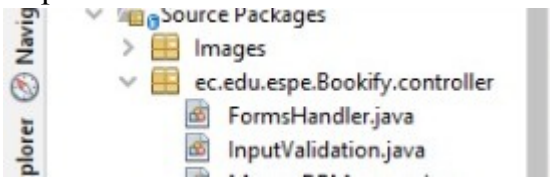
variables that start with a capital letter -0.1

```
Toolkit miPantalla
Image miIcono = miF
```

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);
```

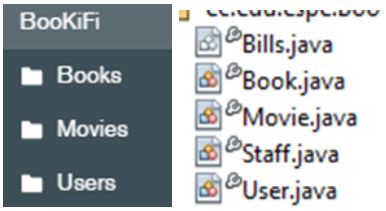
Duplicate code



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);
}
```

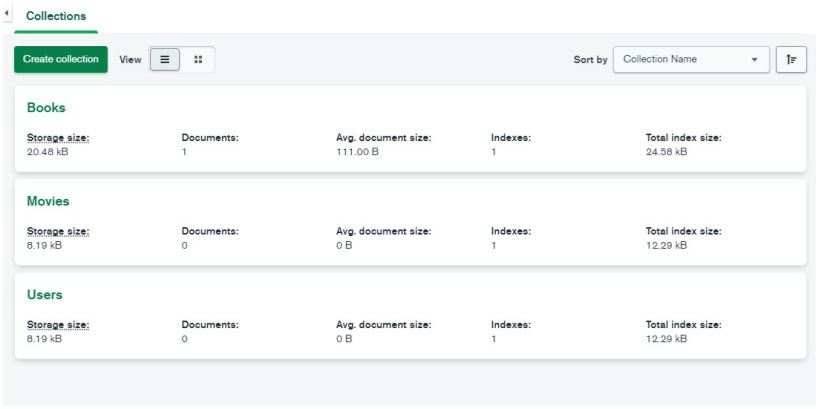
dataOfReturn is missing from class diagram customerNames does not match diagram.

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;
```

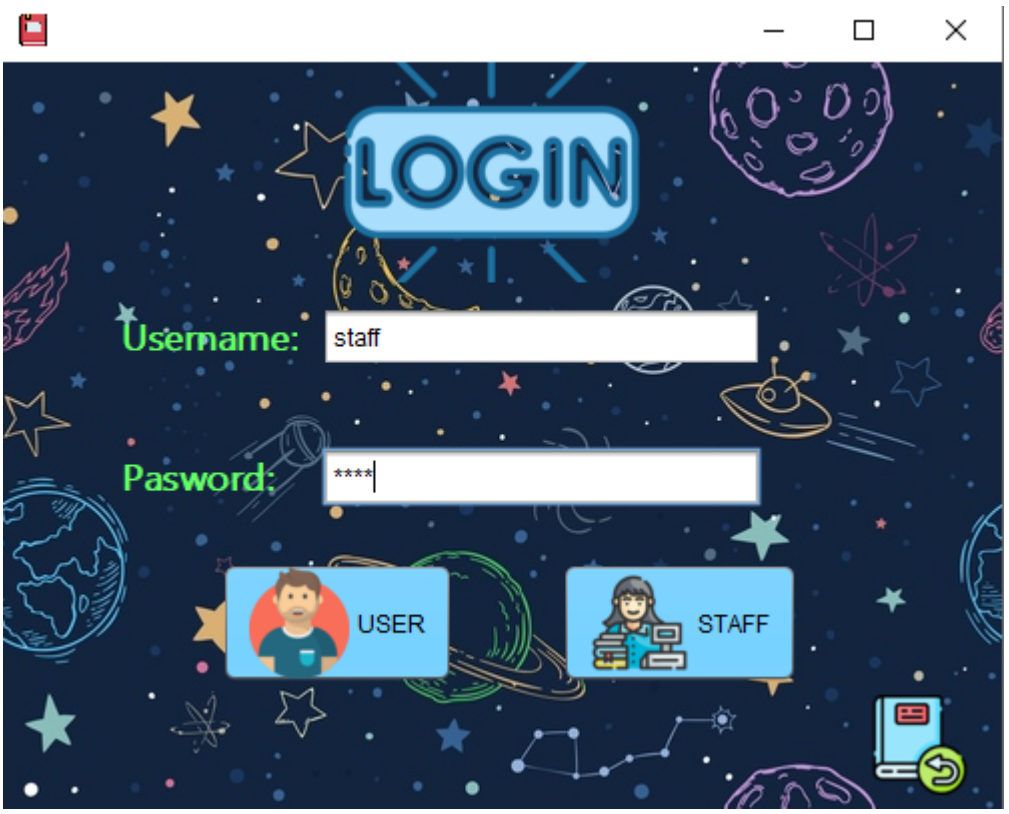
```
ublic class User extends Bills{
    private String UserName;
    private int UserId;
    private String UserEmail;
    private int UserPhone;
    private String UserAddress;
    private int UserAge;
    private int UserPassword;
    int Pasuwu;
    int Password = 1945;
```

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



5.GUI

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

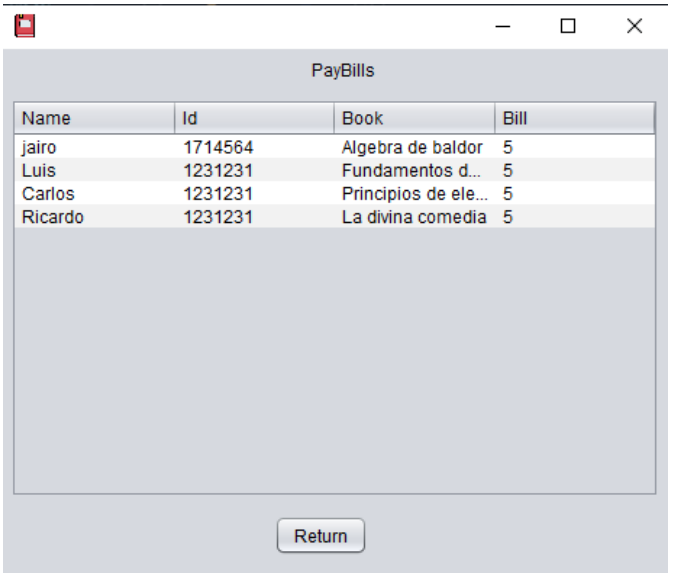


LogIn information it´s a predesignated password and user (-0.1)

6.Tables:

data come from DB 5

print data to printer 5



```
Object[] userRow = new Object[]{"jairo", "1714564","Algebra de baldo
table.addRow(userRow);
userRow = new Object[]{"Luis", "1231231","Fundamentos de Circuitos E
table.addRow(userRow);
userRow = new Object[]{"Carlos", "1231231","Principios de electronic
table.addRow(userRow);
userRow = new Object[]{"Ricardo", "1231231","La divina comedia","5"]
```

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 JTextFields) -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

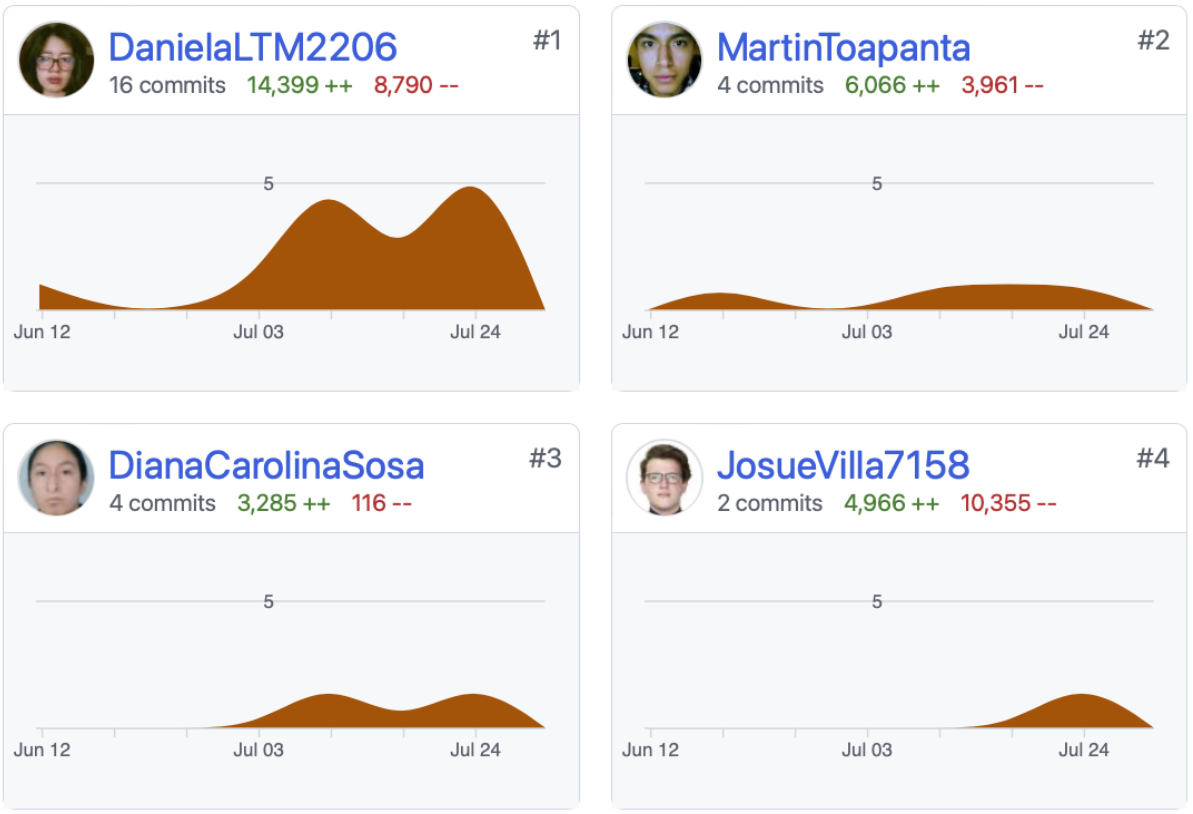
TEAM 07 MyWayCode INSPECTOR: TEAM 01 BettaCoders(Luis Burbano)

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

Proyect: 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

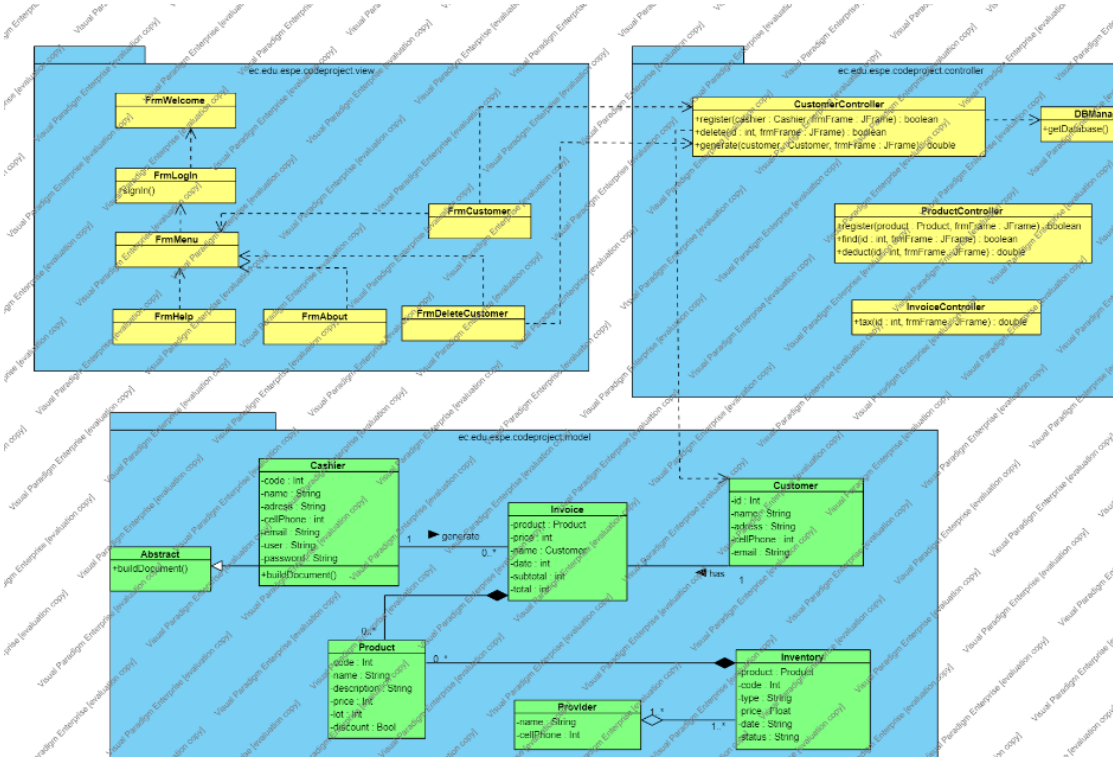
- Login Form
- 5 Forms for Information (CRUD operations, 4 business rules)

  Every form
  - Every input (JTextField) is validated
  - There are different types of inputs (not everything is a JTextFields) -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%

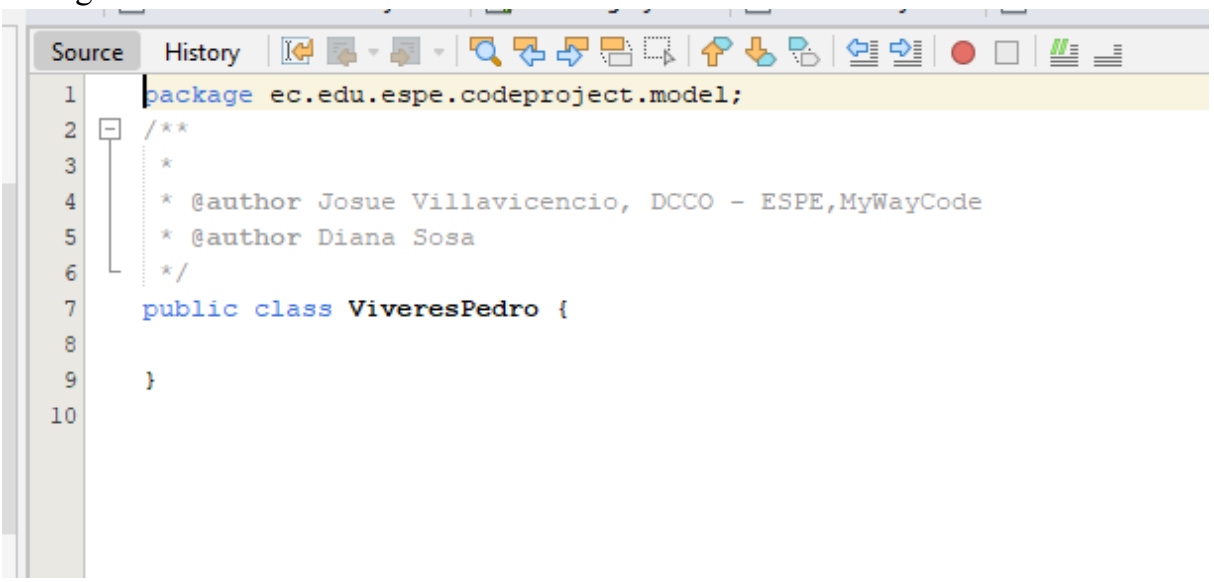
Unit Test



Clean Code

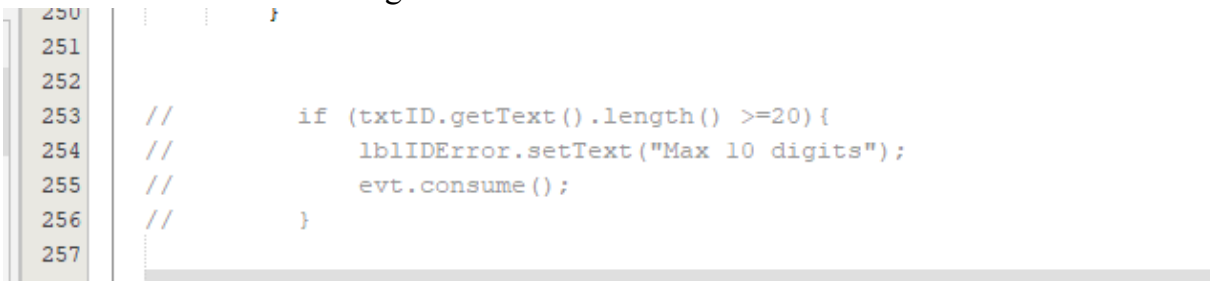
```
    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
        String Username="oop";
        String Password="1234";

        String Pass=new String(this.Password.getPassword());

        if(txtUsername.getText().equals(Username)&&Pass.equals(Password)){
```

Wrong name of button variables



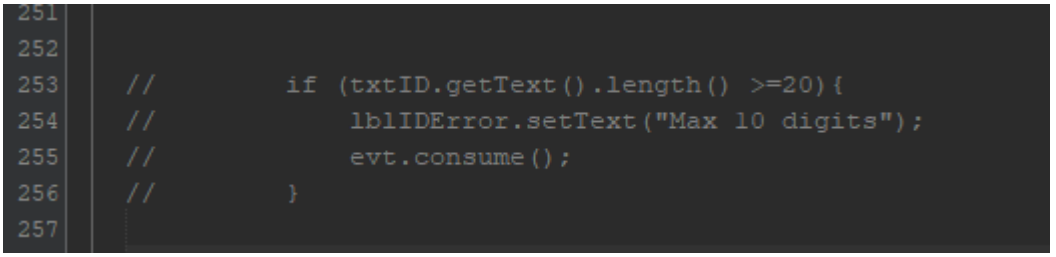
```
Source  History
1  package ec.edu.espe.codeproject.model;
2  /**
3   *
4   * @author Josue Villavicencio, DCCO - ESPE,MyWayCode
5   * @author Diana Sosa
6   */
7  public class ViveresPedro {
8
9  }
10
```

classes created without using



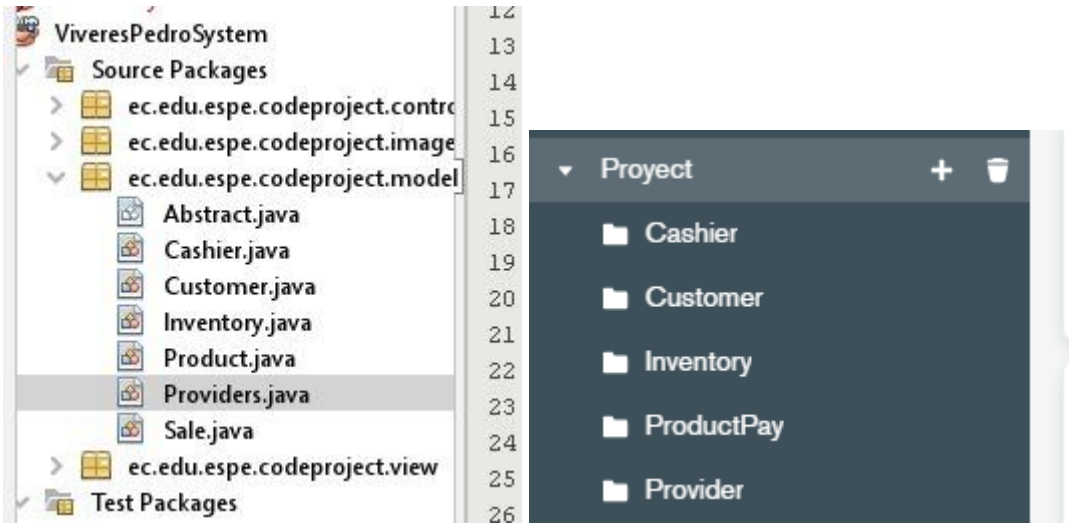
```
251
252
253  //          if (txtID.getText().length() >=20){
254  //              lblIDError.setText("Max 10 digits");
255  //              evt.consume();
256  //          }
257
```

unused code



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

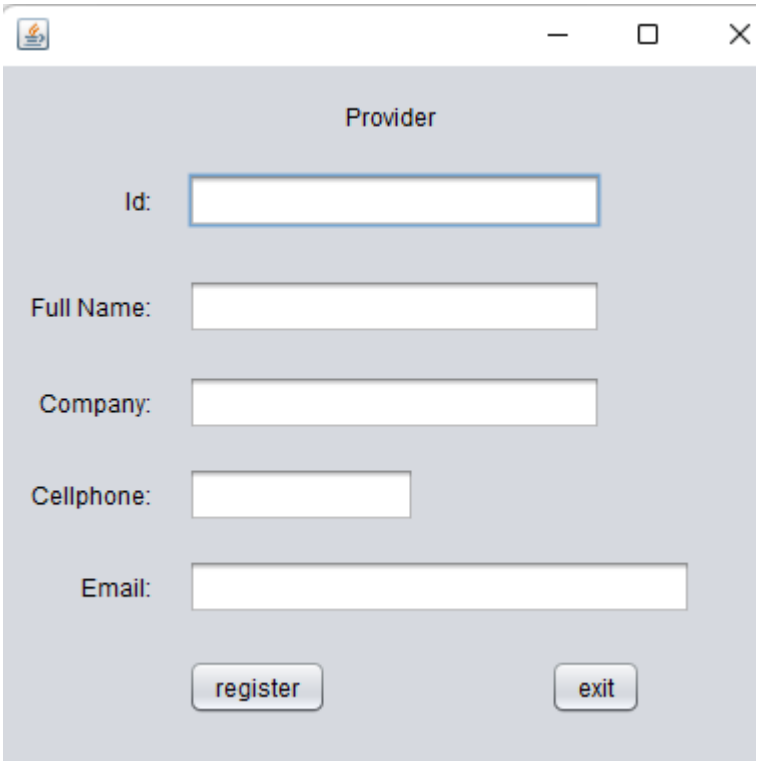
MONGO



Authentication(With mongoDB not with burned data)

```
    }

    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)){

        FrmMenu EP=new FrmMenu();
        EP.setVisible(true);
        dispose();

        }
        else{
         JOptionPane.showMessageDialog(this,"Incorrect Username / Password");
       }
    }
```

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


Provider
Id:
Full Name:
Company:
Cellphone:
Email:
register
exit