University Distrital Francisco Jose De Caldas

Nelson de Jesús Navarro de la Rosa 20242020116

Jefferson David Rico Ruiz 20242020108

Engineering of Systems

Report Workshop 3

Development of a buying and selling app based of TEMU Application

Introduction

An application inspired by the Temu model was developed, replicating its main features and adapting them to an academic and experimental approach. Developed as part of an academic exercise, this application incorporates features such as product navigation, category filtering, detailed item viewing, purchasing process simulation, and shipping status.

Motivation

The main motivation for carrying out this project arises from the constant changes brought about by the digital age, which have significantly transformed the way people carry out their daily activities. In particular, the way we shop, access products, and consume services has rapidly evolved thanks to mobile technologies and e-commerce platforms. This change in consumption habits has generated new needs, such as immediacy, ease of use, and accessibility from anywhere. Considering this reality, we decided to develop a Temu-like application, with the goal of exploring how to build a modern digital solution that responds to these new behaviors and expectations.

1. Requirements

In this section we define the objectives or functional requirements that describe behaviors or actions that the application should have and the non-functional requirements that seek to show characteristics that the application should have at a general level.

1.1 Functional requirements:

1. User Registration: The system must allow users to register by providing their name, email address, and password.
2. User login: The system must allow users to log in using their registered email and password.
3. Product exploration: Users must be able to browse products categorized by type and view individual product details.
4. Order Tracking: Users must be able to view the status of their orders (pending, shipped, delivered).
5. Product Management (Seller): The system must allow sellers to upload products with a name, description, image, and price.
6. Sales Overview (Seller): Sellers must be able to view a list of products sold and basic sales statistics.
7. Buy product: The app must allow the user to purchase the desired product.
8. Record payments and sales: The system must have a section where the seller can view payments and sales for their products.
9. Implement Promotions: The seller can set up promotions for a specific product.
10. Display offers: The user can view product offers upon entering the app.

1.2 No functional requirements:

1. An interactive and easy-to-use interface: The application's user and seller interfaces must be clear and intuitive.
2. Security: Ensure the security of the user's information and confidential data, such as their password or payment information.
3. Performance: The system must respond efficiently to user interaction, with agile processes and minimal response times.
4. Modularity: The code must be divided into independent modules to facilitate maintenance and scalability.

1.3 Changes

We analyzed the functional and non-functional requirements considering the solid principles and decided that it is not necessary to add more or make any modifications since they are the objectives or characteristics that our app must have.

2. User Stories

Here we create user stories that seek to briefly but specifically describe a functionality that the application will have, including aspects such as who it is aimed at, what it should do, with what objective, and when it will do it.

2.1 Changes

We analyzed the user stories and decided to create more of them, where we sought to ensure that each story had a clear functionality, explaining it more briefly, and adding new functionalities such as buttons and interfaces.

USER STORY		
TITLE: Login Interface	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As an app user, I want a login interface where I can choose whether to log in as a customer or seller to log into the app with my account.		
Acceptance Criteria: Given a previous registration in the app, I can enter data such as username and password, and if I click Log In, I'll be redirected to the main interface for either the seller or customer.		

USER STORY		
TITLE: User registration	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks
User Story: As an app user, I want one interface to register and provide information such as name, age, residential address, etc., to create an account.		
Acceptance Criteria: When I access the app, if I click "Register," I will be redirected to the corresponding interface.		

USER STORY

TITLE: Main interface (Customer)	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As an app user, I want to have a main interface that allows me to easily access the products I need through categories.		
Acceptance Criteria: Given a series of product categories, when the user interacts with them, he or she will be able to access the section of the selected category.		

USER STORY

TITLE: Products on Sale	**PRIORITY:** Low	**ESTIMATE:** 3 weeks
User Story: As a user, I want a window to appear with a discounted product upon entering the app. This is to attract attention and encourage people to buy.		
Acceptance Criteria: Given a few discounted products, choose one at random to display upon app launch. If the user interacts with the offer window, they'll be redirected to the discounted product. Otherwise, they'll be redirected to the initial interface.		

USER STORY

TITLE: Product Purchase Interface	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As a user, I want to be redirected to an interface that displays the product image, price, description, user reviews, and a "Buy Now" button . This allows the user to choose the product they want to purchase and obtain more information.		
Acceptance Criteria: Once a product is selected, an interface will be displayed with the product and its information.		

USER STORY

TITLE: Pay for product	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As a buyer, I would like to be redirected to the payment methods when selecting the "Buy Now" button.		
Acceptance Criteria: Once the product is selected, I can click the "Buy Now" button and then be redirected to the payment methods section.		

USER STORY

TITLE: Payment methods	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As an app user, I would like to be able to choose between several payment methods when paying for a product, so I can use the one that's most convenient for me.		
Acceptance Criteria: After choosing the payment method, I will be asked for the necessary information to complete the purchase.		

USER STORY

TITLE: Return to home (button)	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As a user, I want the "Return" button to redirect me to the main interface to continue interacting with the products.		
Acceptance Criteria: Once the interface is deployed, when the user interacts with the button, it will redirect them to the main interface.		

USER STORY

TITLE: Main (Seller) Interface	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As a seller, I want a space where the products I sell are displayed, along with their price, image, and name. I also want a "Add Product" button and a "Sold Products" button to view them.		
Acceptance Criteria: Once the user has registered as a seller, they will be redirected to this interface upon logging in.		

USER STORY

TITLE: Add Product(button)	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As a seller, I want a button on the main seller interface where I can add products to sell.		
Acceptance Criteria: With the button implemented in the interface, if the seller interacts with it, they will be redirected to the section.		

USER STORY

TITLE: Add Product Interface	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks
User Story: As a seller, I want a space where I can add the product to the application and enter data such as the name, price, image, and description, and assign it to the category to put the product for sale.		
Acceptance Criteria: Given the add product button, when the seller interacts with it, they should be redirected to that interface.		

USER STORY

TITLE: Publish Product(button)	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As a seller, I want a "Publish Product" button to complete the product addition process.		
Acceptance Criteria: With the "Add Product" interface, when the user interacts with the button, the product is published in the app, added to the seller's main interface, and redirected to that interface.		

USER STORY

TITLE: Products sold (Button)	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks
User Story: As a seller user, I want the 'Sold Products' button to take me to the interface where all my sold products are so I can manage my sales.		
Acceptance Criteria: Given the main interface of the seller, when the user interacts with the button, they will be directed to the respective sold products interface.		

USER STORY

TITLE: Products sold Interface	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks
User Story: As a seller, I want an interface where I can see all the products I have sold in the application to manage my sales.		
Acceptance Criteria: Given the main interface of the seller, when the seller interacts with the button, they will be directed to that interface.		

3. Crcs Cards

In this section we present the main classes that the application will have, where we mention what roles or responsibilities they will perform, as well as the classes with which they collaborate and help fulfill their responsibilities and how they achieve this.

3.1 Changes

We analyzed CRC cards and sought to add more responsibilities to the classes. We decided to create a class registration and profile offering to comply with the single responsibility principle where each class has a clear responsibility.

CRC CARDS

Class: Seller

Responsabilities	Collaborators
- The seller has to publish his products on the application. - Ensure a correct purchase and sale process with the user. - Coordinate the shipment or delivery of the product as established in the app. - Follow the rules and terms of use of the app.	Product - The seller publishes and adds numerous products with which they interact to achieve their goal, allowing them to establish contact with the customer. Payment - Recieve data of monitor sales Profile - Can collect data such as name, age, etc. to validate each seller

CRC CARDS

Class: Customer

Responsabilities

- Order the product and finish paying for it.
- Have the ability to choose the desired product(s) across categories.
- Provide information when paying for a product

Collaborators

Products
-You can view and buy many products, each from different sellers.

Payment
- The customer can use this class to make a purchase.

Profile

- The customer can enter login information such as their name, age, etc. to validate each user.

Offer
- The customer can interact with the displayed offe

CRC CARDS

Class: offer

Responsabilities

- This class seeks to display the offer window with some discounted product

Collaborators

Customer
- The offer window is displayed to the customer.

Product
- To display the offer, the class must select the product.

CRC CARDS

Class: Product

Responsabilities

- Have a photo, description, and price to provide information to the buyer
- Know what type of item it is and, therefore, what category it should be associated with.
- Going on offer eventually.

Collaborators

Seller
- This person creates and manages product features. They can implement offers, modify prices, and modify images.

Customer
- The customer can view and purchase the product.

Payment
- The product can pass data to the Payment class, such as its price and image.

Offer
- The offer select a product

CRC CARDS

Class: Payment

Responsabilities

- Record the sale of a certain product
- Validate the data according to the card number, security number and amount of money on the card and discount the value of the product.
- Calculate the new value to be paid depending on the discount percentage, if there is one.

Collaborators

Customer
- Can provide information such as their name, home address, etc., for the payment process

Product
- Can provide the price and image of the item

Seller
- Receives payments made for this class and stores them in the sales record

CRC CARDS

Class: Register	
Responsabilities	**Collaborators**
- Collect information from the user	Profile - The register class passes user data to the profile class

CRC CARDS

Class: Profile	
Responsabilities	**Collaborators**
- Store the data supplied by the user.	Seller - Stores the data provided by the seller in the class. A registry is used to store it. Customer - Stores the data provided by the customer in the class. A registry is used to store it.

CRC CARDS

Class: App	
Responsabilities	**Collaborators**
- Show the main interface	Seller - Displays the main interface of the seller Customer - Displays the main interface of Customer

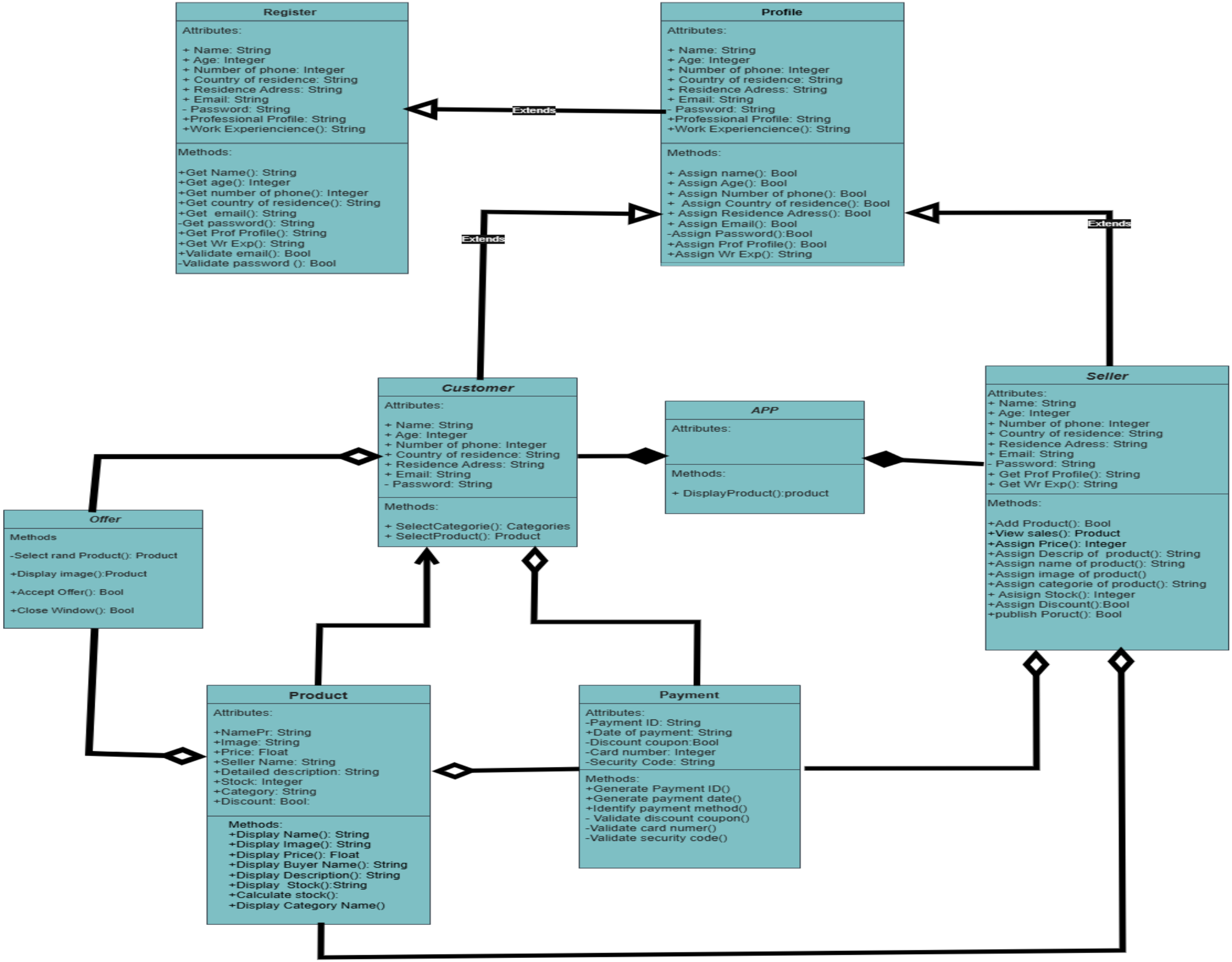
4. UML diagrams

In this section, we will show class, activity, and sequence diagrams.

In class diagrams, we seek to show the relationships between different classes, with the methods and attributes expected for each class, in addition to ensuring and displaying relationships such as association, composition, aggregation, inheritance, or implementation.

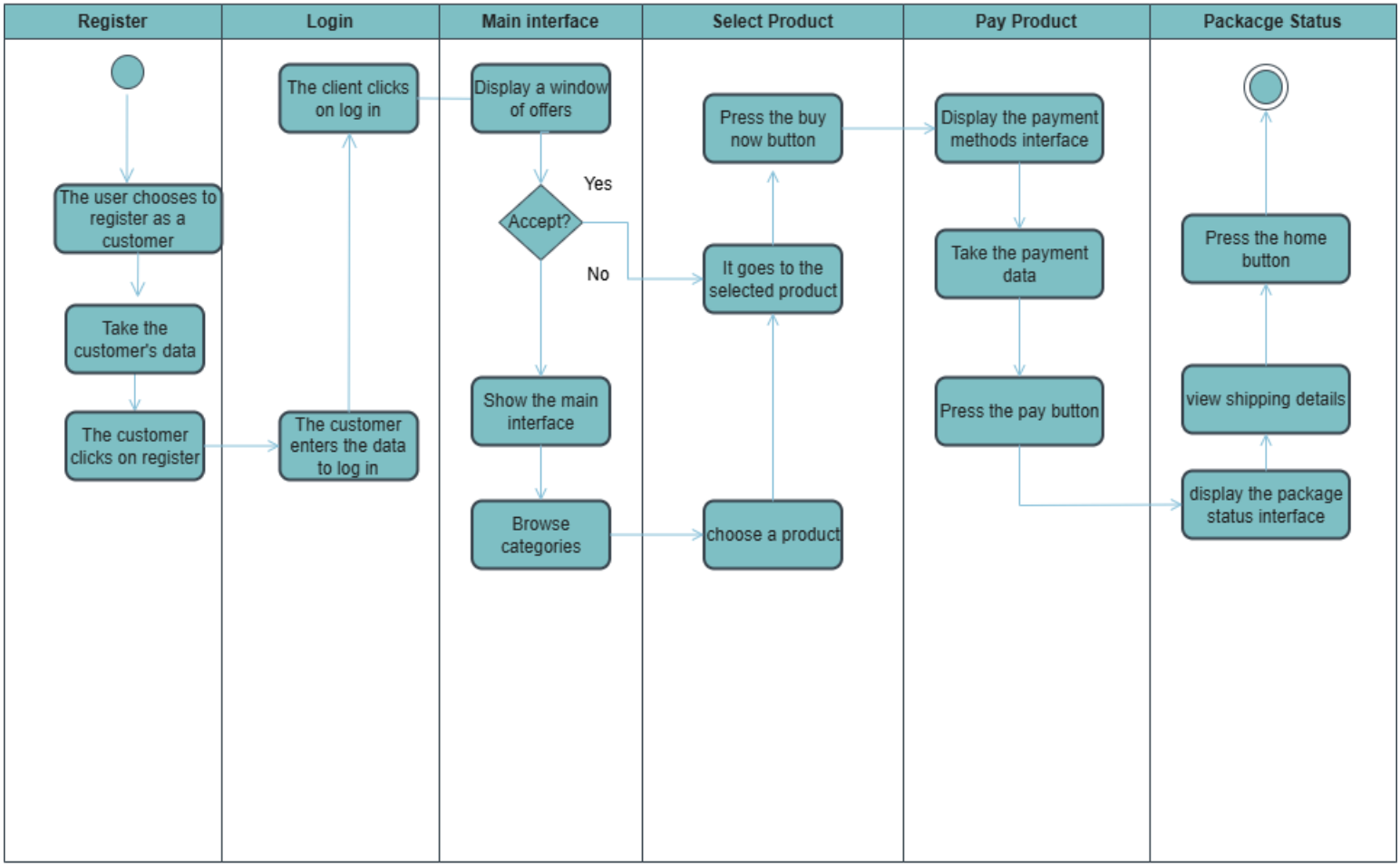
4.1 Changes

For the UML diagram, we place the arrows that indicate the type of relationship between the classes. We add a new class to try to guarantee the principle of single responsibility and that the classes also allow adding new functionalities if required, complying with the principle of open to extension, closed to modification.

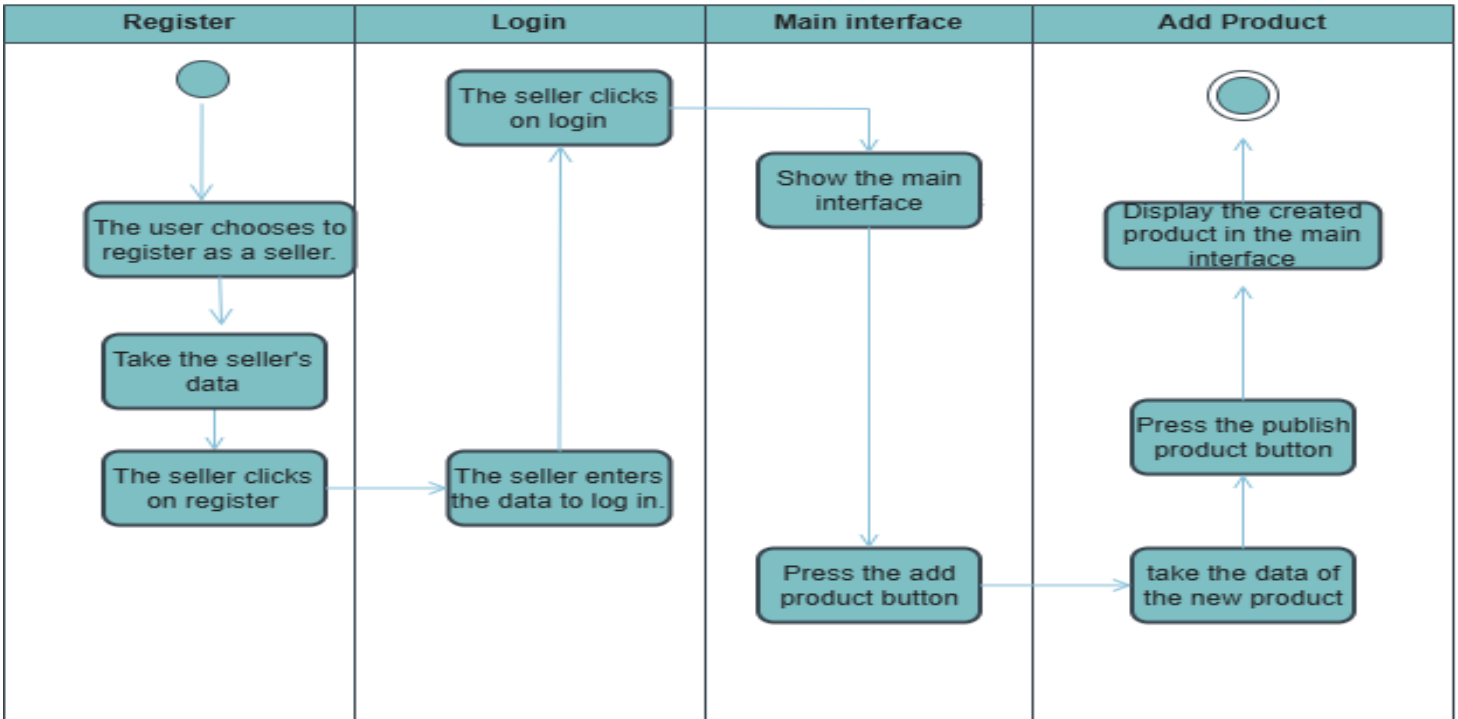

Register
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
+Professional Profile: String
+Work Experiencience(): String
Methods:
+Get Name(): String
+Get age(): Integer
+Get number of phone(): Integer
+Get country of residence(): String
+Get email(): String
-Get password(): String
+Get Prof Profile(): String
+Get Wr Exp(): String
+Validate email(): Bool
-Validate password (): Bool
Extends
Profile
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
+Professional Profile: String
+Work Experiencience(): String
Methods:
+ Assign name(): Bool
+ Assign Age(): Bool
+ Assign Number of phone(): Bool
+ Assign Country of residence(): Bool
+ Assign Residence Adress(): Bool
+ Assign Email(): Bool
-Assign Password():Bool
+Assign Prof Profile(): Bool
+Assign Wr Exp(): String
Extends
Extends
Customer
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
Methods:
+ SelectCategorie(): Categories
+ SelectProduct(): Product
APP
Attributes:
Methods:
+ DisplayProduct():product
Seller
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
+ Get Prof Profile(): String
+ Get Wr Exp(): String
Methods:
+Add Product(): Bool
+View sales(): Product
+Assign Price(): Integer
+Assign Descrip of product(): String
+Assign name of product(): String
+Assign image of product()
+Assign categorie of product(): String
+ Asisign Stock(): Integer
+Assign Discount():Bool
+publish Poruct(): Bool
Offer
Methods
-Select rand Product(): Product
+Display image():Product
+Accept Offer(): Bool
+Close Window(): Bool
Product
Attributes:
+NamePr: String
+Image: String
+Price: Float
+Seller Name: String
+Detailed description: String
+Stock: Integer
+Category: String
+Discount: Bool:
Methods:
+Display Name(): String
+Display Image(): String
+Display Price(): Float
+Display Buyer Name(): String
+Display Description(): String
+Display Stock():String
+Calculate stock():
+Display Category Name()
Payment
Attributes:
-Payment ID: String
+Date of payment: String
-Discount coupon:Bool
-Card number: Integer
-Security Code: String
Methods:
+Generate Payment ID()
+Generate payment date()
+Identify payment method()
- Validate discount coupon()
-Validate card numer()
-Validate security code()

For activity diagrams, we seek to show the flow of information or the sequence of steps to carry out a certain activity in the application, such as actions and processes.

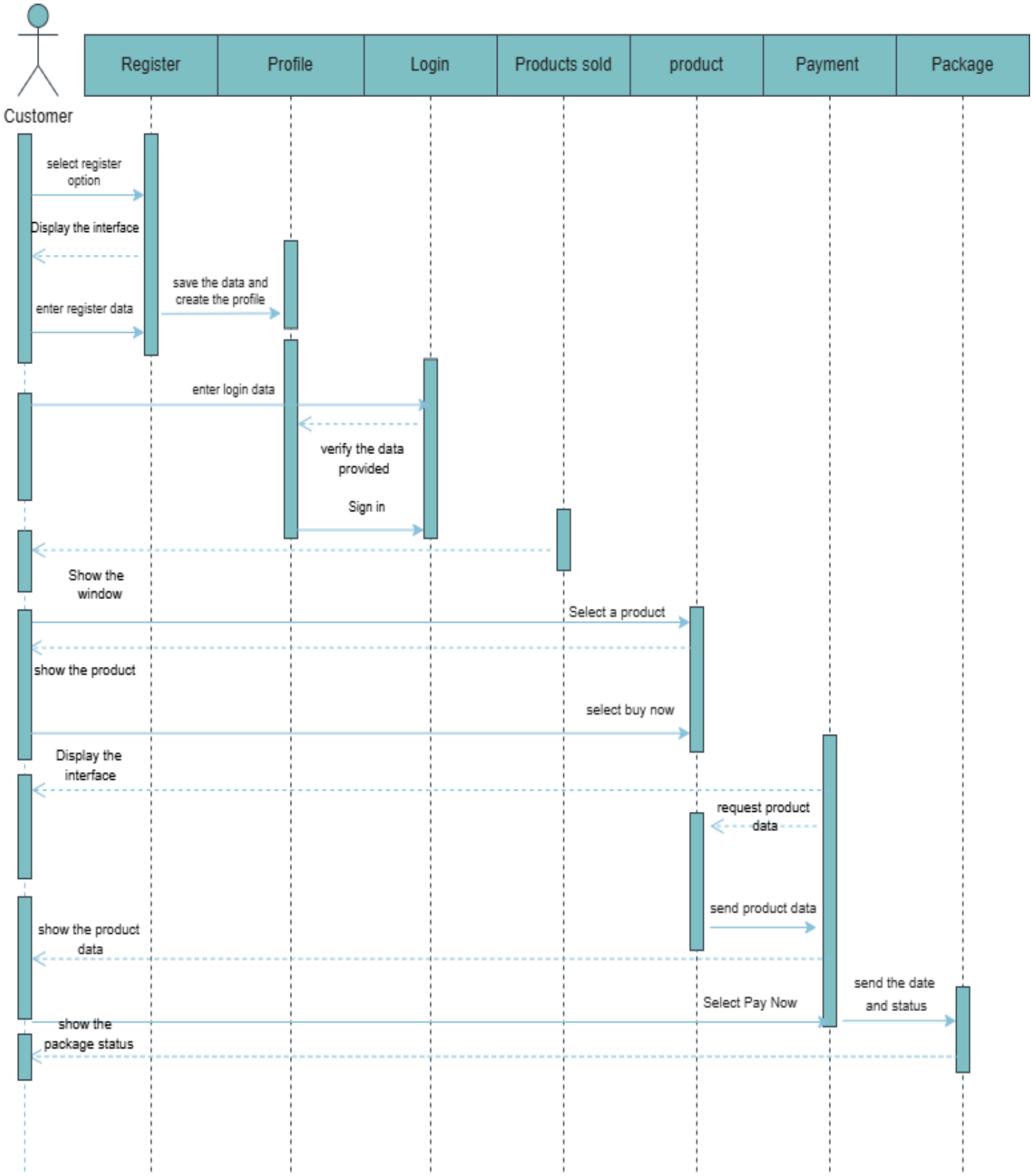
Activity Diagram Product Purchase

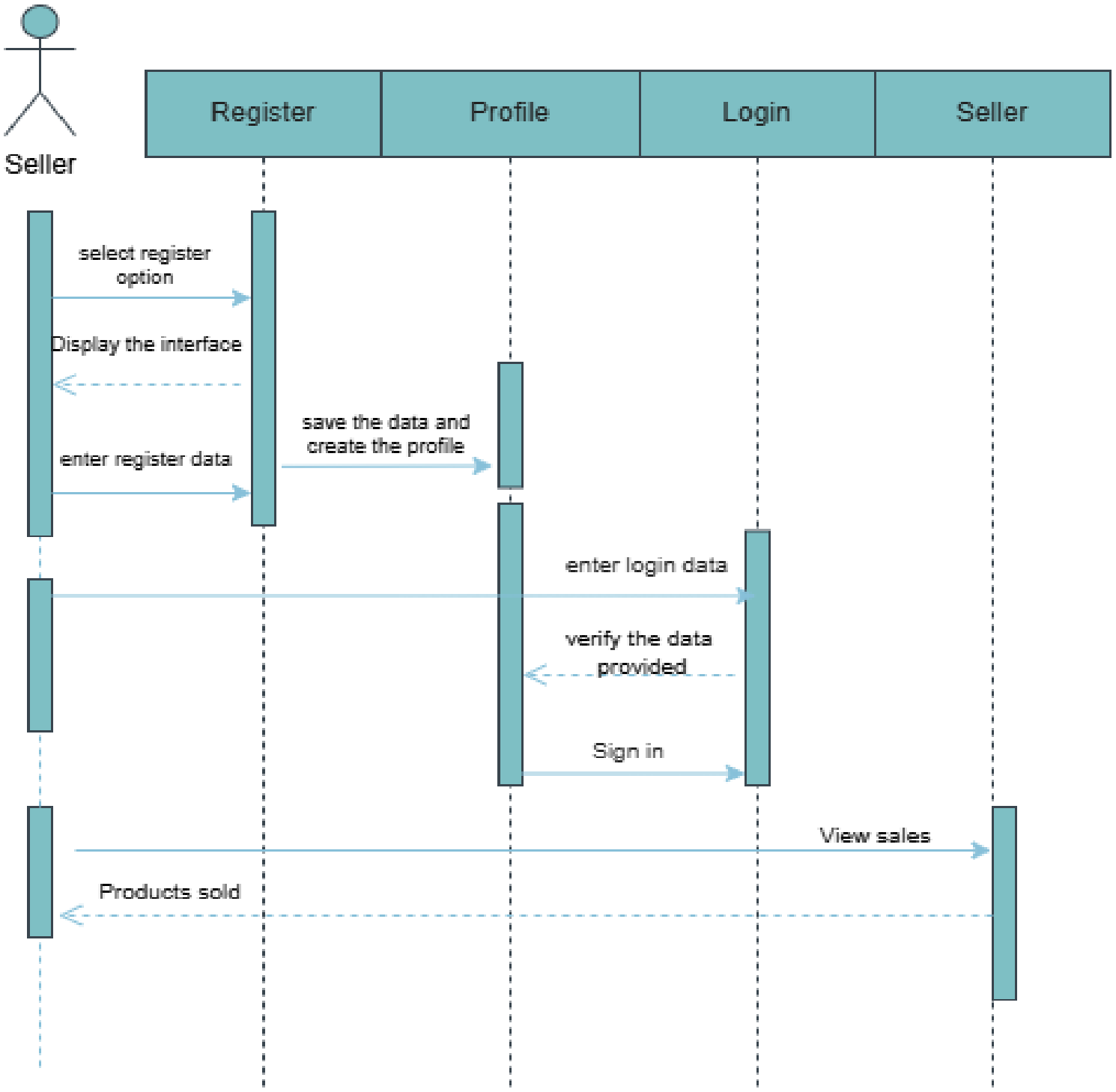


Activity Diagram Add Product



And in sequence diagrams, we seek to show the relationships between classes sequentially, where importance is given to the order in which information flows.





5. Solid Implementation

In this section we explain how we apply each solid principle to our design

1 Single Responsibility:

This means that classes should have a single responsibility. This can be seen in our design, for example, in the product class, which is only responsible for displaying product information, and in the package class, which is only responsible for package data.

2 Open/Closed Principle:

This principle is used to add new functionality without altering the behavior of the base class. This principle is applied in our design, as we use classes such as Payment, Product, Seller, and Customer, which allow us to extend behavior without modifying specific classes. For example, if we want to add product reviews, we could add this functionality to the product class.

3 Liskov Substitution Principle:

This principle seeks to ensure that a subclass or child class can replace the parent class without affecting its functionality. In our design, without clearly defined child classes, we cannot apply this principle.

4 Interface Segregation Principle:

This principle seeks to ensure that each application's functionality or client has its own interface. This implies having multiple interfaces for a specific purpose or client, rather than having interfaces with multiple overloaded functions. In our design, since we don't have interfaces or more than one class that can implement the same interface, we can't apply this principle.

5 Dependency Inversion Principle:

This principle states that high-level classes should not depend on low-level classes, but rather on abstractions.

In our design, we don't have high-level and low-level classes, so we can't apply this principle.

6. Code Snippets

In this section we show code advances related to the logic of some of the relevant classes in the registration and creation of the user profile.

```java
/**
 * @method:This method asks the user for the email and if it does not contain
 *              the extensions of an email it will delete it and return the
 *              value of email.
 *
 * @return: the email of user
 */

public String getemail() {
    System.out.println("Enter your email");
    email = inputs.nextLine();
    if (email.contains("@gmail.com") || email.contains("@hotmail.com")) {
    } else {
        email = null;
    }
    return email;
}

/**
 *@method: This method asks the user for the password and, once verified, if it does not
 *              have a minimum length of 6 characters, it deletes it.
 *
 * @return: the password of user
 */

public String getpassword() {
    System.out.println("Enter your password");
    password = inputs.nextLine();
    if (password.length() < 6) {
        password = null;
    }
    return password;
}
```

```java
/**
 *
 * @method:This method asks the user for the name, receives it and validates
 *              that it has a length greater than 3, otherwise it deletes it.
 * @return:the name of user
 */
public String getname() {
    System.out.println("Enter your name");
    name = inputs.nextLine();
    if (name.length() < 3) {
        name = null;
    }
    return name;
}

/**
 * @method:This method asks the user's age and if it is negative or less than
 *              18, it deletes the data.
 * @return: the age of user
 */
public Integer getage() {
    System.out.println("Enter your age");
    age = inputs.nextInt();
    if (age < 0 || age < 18) {
        age = null;
    }
    return age;
}

/**
 *
 * @method:This method asks for the user's phone number and if it has less than
 *              10 digits, it deletes the data and return the value.
 *
 * @return: the number of phone of user
 */
public Long getnumberofphone() {
    System.out.println("Enter your number of phone");
    numphon = inputs.nextLong();
    inputs.nextLine();
    String numStr = Long.toString(numphon);
    if (numStr.length() != 10) {
        numphon = null;
    }
    return numphon;
}
```

```java
public Profile(String name, Integer age, Long numphon, String country, String resiadd, String email,
            String password) {
        this.name = name;
        this.age = age;
        this.numphon = numphon;
        this.country = country;
        this.resiadd = resiadd;
        this.password = password;
        // TODO Auto-generated constructor stub
    }

    /**
     * @method:The method takes the entered name and checks if the data was deleted.
     *             If it was, the user is notified and prompted to enter it
     *             correctly.
     *
     * @return true if the data meets the requirements for assignment
     * @return false if the data does not meet the requirements for assignment
     */
    public Boolean assignname() {
        if (name == null) {
            while (name == null) {
                System.out.println("Ingrese un nombre valido");
                super.getname();
            }
            return false;
        } else {
            return true;
        }

        /**
         * @method:The method takes the entered age and checks if the data was deleted.
         *             If it was, the user is notified and prompted to enter it
         *             correctly.
         *
         * @return true if the data meets the requirements for assignment
         * @return false if the data does not meet the requirements for assignment
         */

    }

    public Boolean assignage() {
        if (age == null) {
            while (age == null) {
                System.out.println("Ingrese una edad valida");
                super.getage();
            }
            return false;
        } else {
            return true;
        }
    }

    /**
     * @method:The method takes the entered number of phone and checks if the data
     *             was deleted. If it was, the user is notified and prompted to
     *             enter it correctly.
     *
     * @return true if the data meets the requirements for assignment
     * @return false if the data does not meet the requirements for assignment
     */

    public Boolean assignnumphon() {
        if (numphon == null) {
            while (numphon == null) {
                System.out.println("Ingrese un numero de telefono valido");
                super.getnumberofphone();
            }
            return false;
        } else {
            return true;
        }
    }
```