IMPLEMENTACIÓN EN JAVA
CUSTOMER UNION MARKET

PARTE 1: IMPLEMENTACIÓN EN JAVA

CASO DE USO 1 - LOGIN

Hemos desarrollado el caso de uso “login.java” que consiste en el inicio de sesión de nuestros clientes. Se ha llevado a cabo el desarrollo de la siguiente clase en java:

```java
public class Login {
    private Map<String, String> users;

    public Login() {
        users = new HashMap<>();
        // Agregar usuarios y contraseñas
        users.put( key:"usuario1",  value:"contrasena1");
        users.put( key:"usuario2",  value:"contrasena2");
        users.put( key:"usuario3",  value:"contrasena3");
    }

    public boolean authenticate(String username, String password) {
        // Verificar si el usuario existe y la contraseña es correcta
        if (users.containsKey( key:username) && users.get( key:username).equals( anObject:password)) {
            return true; // Autenticación exitosa
        }
        return false; // Autenticación fallida
    }

    public static void main(String[] args) {
        Login login = new Login();

        // Ejemplo de uso
        String username = "usuario1";
        String password = "contrasena1";
        boolean isAuthenticated = login.authenticate(username, password);

        if (isAuthenticated) {
            System.out.println( x:"Inicio de sesión exitoso");
        } else {
            System.out.println( x:"Inicio de sesión fallido");
        }
    }
}
```

La clase ***Login*** tiene un mapa (***users***) que almacena los nombres de usuario como claves y las contraseñas correspondientes como valores. En el constructor de la clase, se agregan algunos usuarios de ejemplo al mapa.

El método ***authenticate*** toma un nombre de usuario y una contraseña como parámetros y verifica si el usuario existe en el mapa y si la contraseña coincide. Devuelve true si la autenticación es exitosa y false en caso contrario.

En el método ***main***, se crea una instancia de la clase ***Login*** y se realiza un ejemplo de uso llamando al método ***authenticate*** con un nombre de usuario y contraseña específicos. Dependiendo del resultado, se muestra un mensaje indicando si el inicio de sesión fue exitoso o no.

CASO DE USO 2 - PEDIR PRESUPUESTO

Hemos desarrollado el caso de uso “presupuesto.java” que consiste en recibir un presupuesto tras elegir una serie de servicios. Se ha llevado a cabo el desarrollo de la siguiente clase en java:

```java
public class Presupuesto {
    private Map<String, Double> preciosServicios;
    private List<String> serviciosSeleccionados;

    public Presupuesto() {
        preciosServicios = new HashMap<>();
        serviciosSeleccionados = new ArrayList<>();

        // Configurar los precios de los servicios
        preciosServicios.put( key:"Marketing",  value:500.0);
        preciosServicios.put( key:"Publicidad",  value:1000.0);
        preciosServicios.put( key:"Community Manager",  value:800.0);
        preciosServicios.put( key:"Diseño gráfico",  value:600.0);
        preciosServicios.put( key:"Desarrollo web",  value:1500.0);
    }

    public void seleccionarServicio(String servicio) {
        // Verificar si el servicio está disponible y agregarlo a la lista de servicios seleccionados
        if (preciosServicios.containsKey( key:servicio)) {
            serviciosSeleccionados.add( e:servicio);
            System.out.println("Servicio \"" + servicio + "\" seleccionado.");
        } else {
            System.out.println("El servicio \"" + servicio + "\" no está disponible.");
        }
    }

    public void mostrarServiciosSeleccionados() {
        System.out.println( x:"Servicios seleccionados:");
        for (String servicio : serviciosSeleccionados) {
            System.out.println("- " + servicio);
        }
    }
```

```
    public double obtenerPresupuesto() {
        double presupuestoTotal = 0.0;
        for (String servicio : serviciosSeleccionados) {
            if (preciosServicios.containsKey( key:servicio)) {
                presupuestoTotal += preciosServicios.get( key:servicio);
            }
        }
        return presupuestoTotal;
    }

    public static void main(String[] args) {
        Presupuesto presupuesto = new Presupuesto();

        // Ejemplo de uso
        presupuesto.seleccionarServicio( servicio:"Marketing");
        presupuesto.seleccionarServicio( servicio:"Publicidad");
        presupuesto.seleccionarServicio( servicio:"Desarrollo web");
        presupuesto.seleccionarServicio( servicio:"SEO"); // Servicio no disponible

        presupuesto.mostrarServiciosSeleccionados();

        double totalPresupuesto = presupuesto.obtenerPresupuesto();
        System.out.println("Presupuesto total: $" + totalPresupuesto);
    }
}
```

La clase ***Presupuesto*** es una representación simple de un sistema de solicitud de presupuestos en una tienda de marketing digital. Permite a los clientes seleccionar servicios disponibles y calcular el presupuesto total en función de los servicios seleccionados.

La clase ***Presupuesto*** contiene los siguientes atributos:

preciosServicios: Un mapa que almacena los precios de los servicios disponibles. La clave es el nombre del servicio y el valor es el precio asociado.
serviciosSeleccionados: Una lista que almacena los servicios seleccionados por el cliente.
La clase tiene los siguientes métodos:

Presupuesto(): El constructor de la clase. Inicializa los atributos y configura los precios de los servicios disponibles.

seleccionarServicio(String servicio): Permite al cliente seleccionar un servicio especificado por su nombre. Verifica si el servicio está disponible y lo agrega a la lista de servicios seleccionados.
mostrarServiciosSeleccionados(): Muestra por pantalla la lista de servicios seleccionados.
obtenerPresupuesto(): Calcula y devuelve el presupuesto total sumando los precios de los servicios seleccionados.
El método ***main*** es un ejemplo de uso de la clase ***Presupuesto***. En este ejemplo, se crea una instancia de la clase, se seleccionan algunos servicios y se muestra la lista de servicios seleccionados. Luego, se llama al método ***obtenerPresupuesto*** para calcular el presupuesto total y se muestra en la consola.

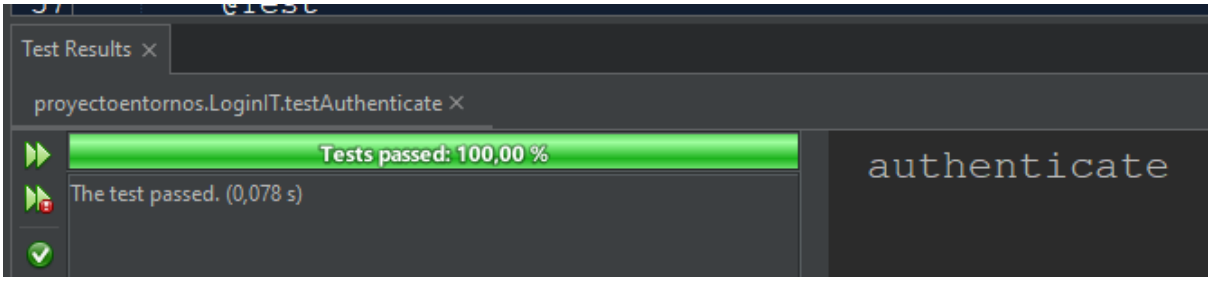
PARTE 2: PRUEBAS

Para comprobar que nuestros métodos y clases son efectivas hemos desarrollado gracias a JUnit una serie de tests. Los resultados se muestran a continuación:

- **CASO DE USO 1: LOGIN**

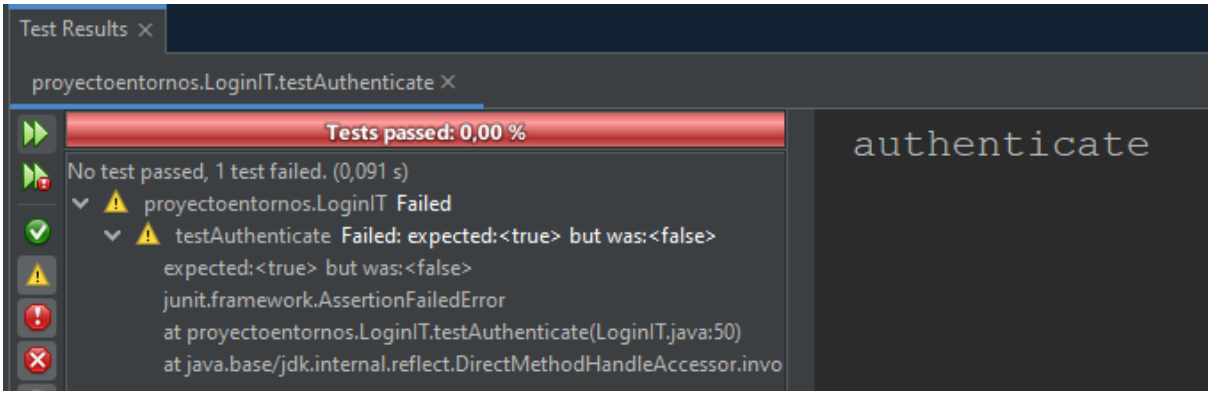
```
/**
 * Test of authenticate method, of class Login.
 */
@Test
public void testAuthenticate() {
    System.out.println( x:"authenticate");
    String username = "usuario1";
    String password = "contrasena1";
    Login instance = new Login();
    boolean expResult = true;
    boolean result = instance.authenticate(username, password);
    assertEquals( expected:expResult,  actual:result);
    // TODO review the generated test code and remove the default call to fail.
}
```

Hemos ejecutado el test con el usuario: *usuario1* y *contrasena1* que previamente estaba creado en la clase como puede comprobarse más arriba. El resultado ha sido el siguiente:



Para terminar de comprobar la eficiencia del método hemos comprobado con *usuario1* y *contrasena2* y este ha sido el resultado:

```
public void testAuthenticate() {
    System.out.println( x: "authenticate");
    String username = "usuario1";
    String password = "contrasena2";
    Login instance = new Login();
    boolean expResult = true;
    boolean result = instance.authenticate(username, password);
    assertEquals( expected: expResult,  actual: result);
    // TODO review the generated test code and remove the default call to fail.
}
```



De esta manera afirmamos que nuestro método actúa correctamente y se comporta como deseamos. Finalizamos las pruebas de este método con satisfacción.

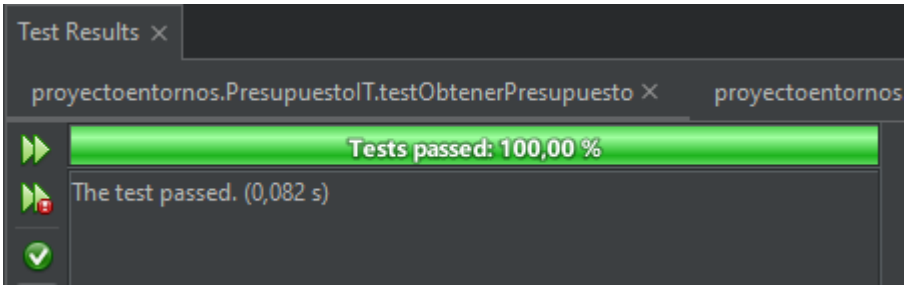
- **CASO DE USO 2: PRESUPUESTO**

A continuación hemos comprobado la eficiencia de nuestra segunda clase ***Presupuesto***.

Para ello vamos a comprobar el método ***obtenerPresupuesto()*** y lo hemos ejecutado sin valores, sin servicios seleccionados, por lo que el resultado esperado debería de ser 0.

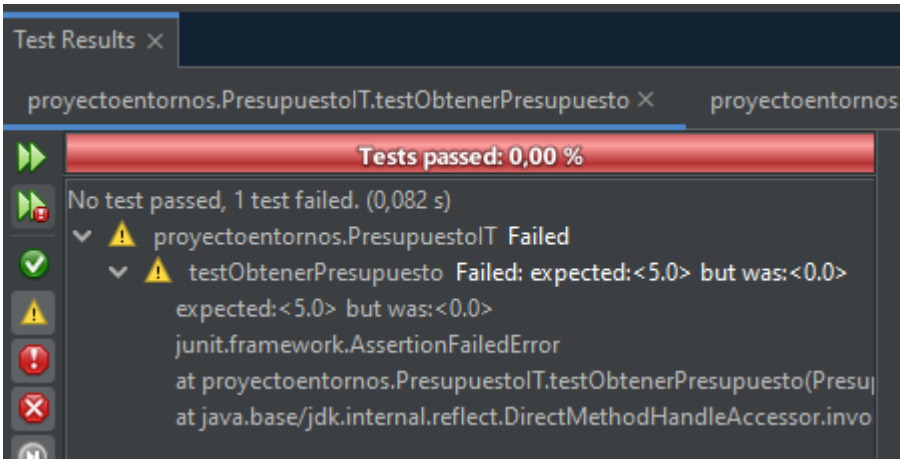
```
public void testObtenerPresupuesto() {
    System.out.println( x: "obtenerPresupuesto");
    Presupuesto instance = new Presupuesto();
    double expResult = 0.0;
    double result = instance.obtenerPresupuesto();
    assertEquals( expected: expResult,  actual: result,  delta: 0);
    // TODO review the generated test code and remove the default call to fail.
}
```

Y este es el resultado:



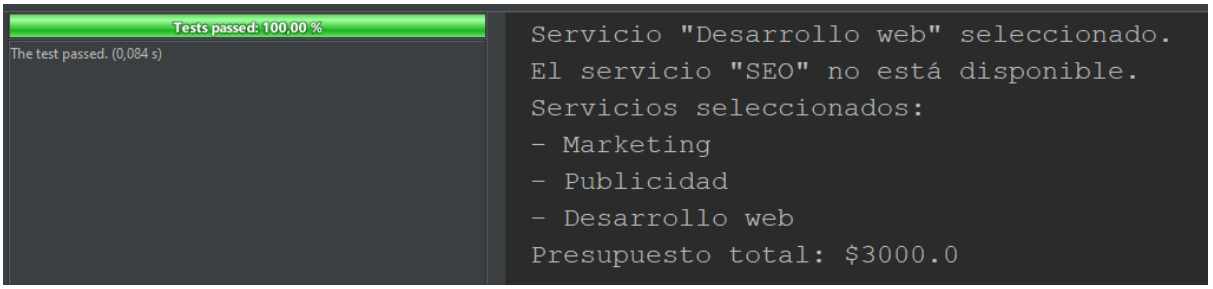
Si cambiamos el valor esperado a “5” el resultado es el siguiente:

```
public void testObtenerPresupuesto() {
    System.out.println( x: "obtenerPresupuesto");
    Presupuesto instance = new Presupuesto();
    double expResult = 5;
    double result = instance.obtenerPresupuesto();
    assertEquals( expected: expResult, actual: result, delta: 0);
    // TODO review the generated test code and remove the default call to fail.
}
```



Por lo tanto afirmamos y comprobamos que nuestro método ***ObtenerPresupuesto()*** funciona correctamente.

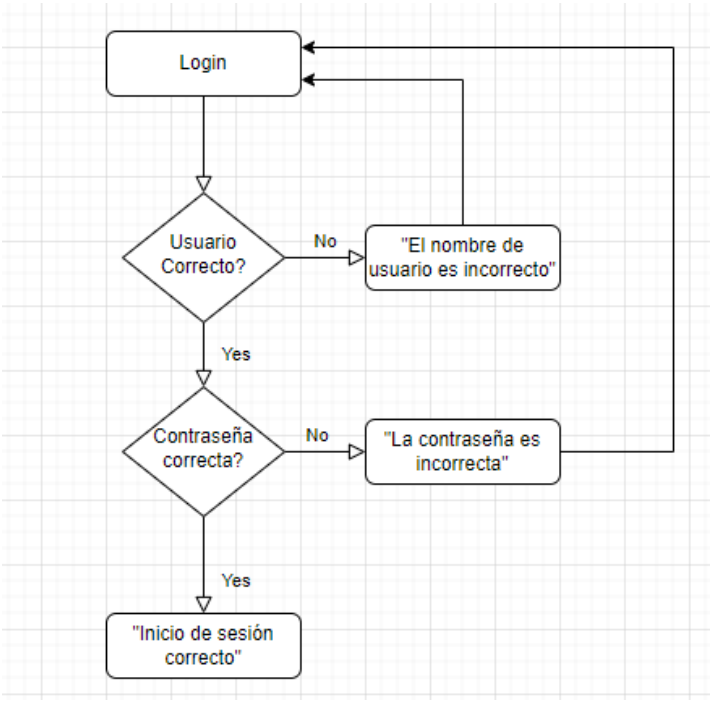
Para terminar de probar el programa hemos ejecutado el método ***main()*** con unos previos valores ya seleccionados en el método ***main()***:



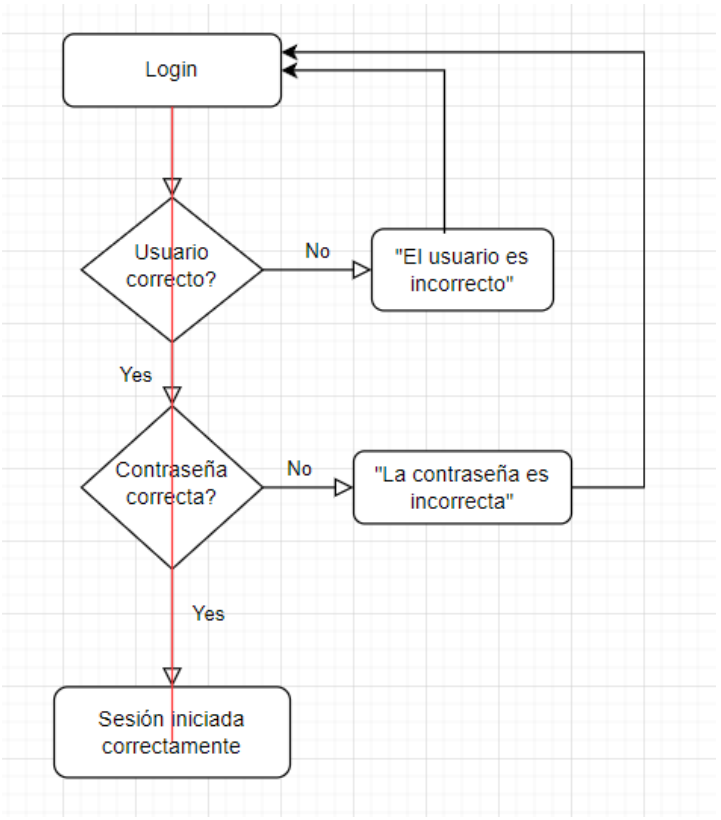
Finalmente afirmamos que la clase funciona perfectamente y a medida.

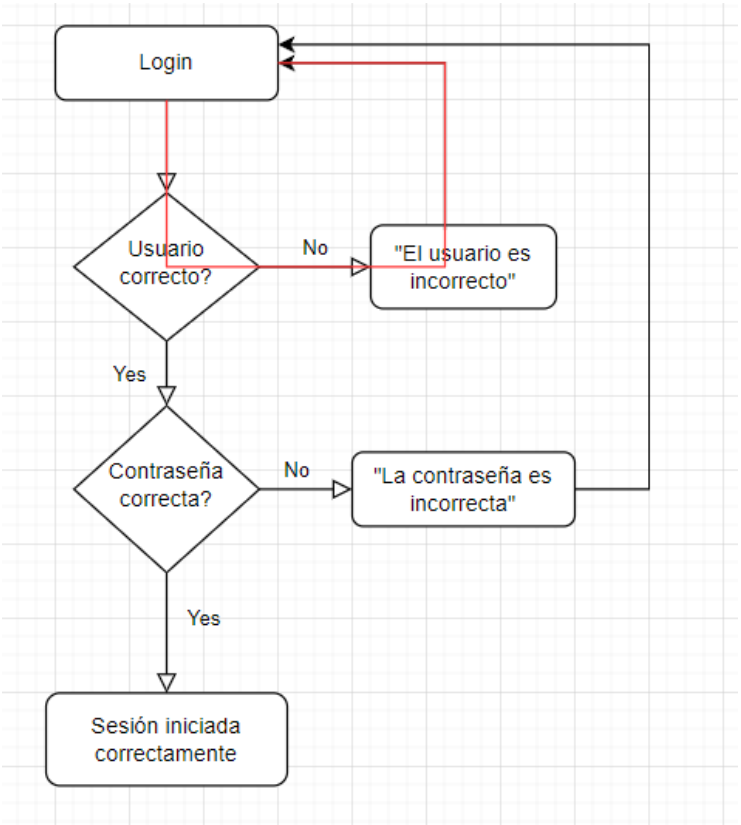
Diagramas de flujo

Diagrama de flujo (clase Login)



- **Caminos Básicos**


Login
Usuario correcto?
No
"El usuario es incorrecto"
Yes
Contraseña correcta?
No
"La contraseña es incorrecta"
Yes
Sesión iniciada correctamente


Login
Usuario correcto?
No
"El usuario es incorrecto"
Yes
Contraseña correcta?
No
"La contraseña es incorrecta"
Yes
Sesión iniciada correctamente

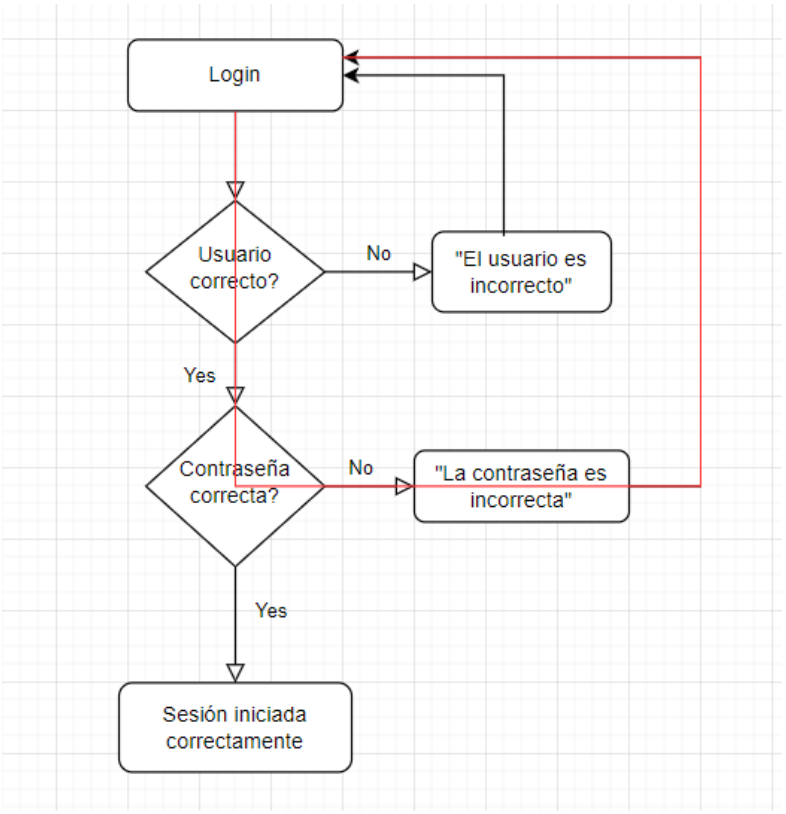
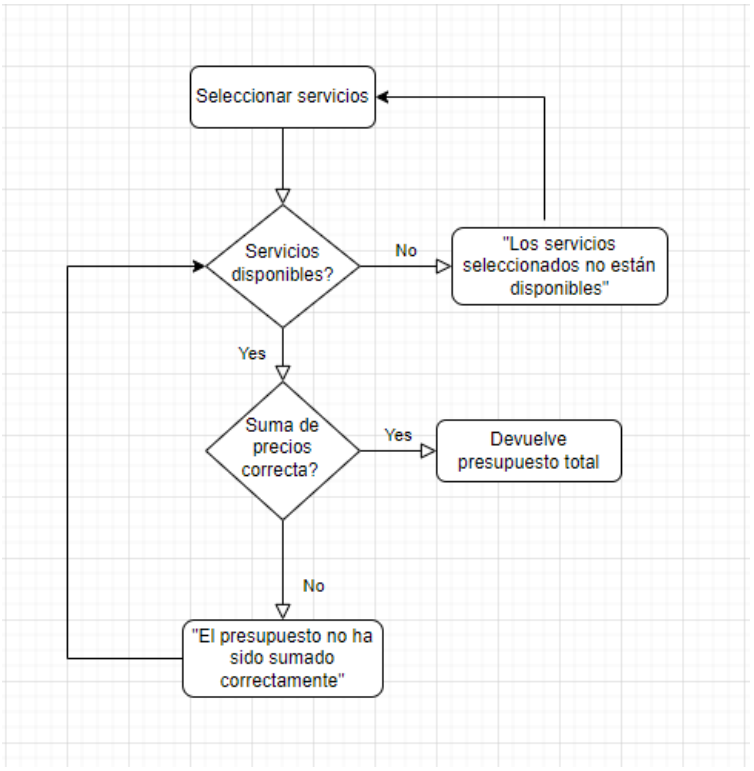
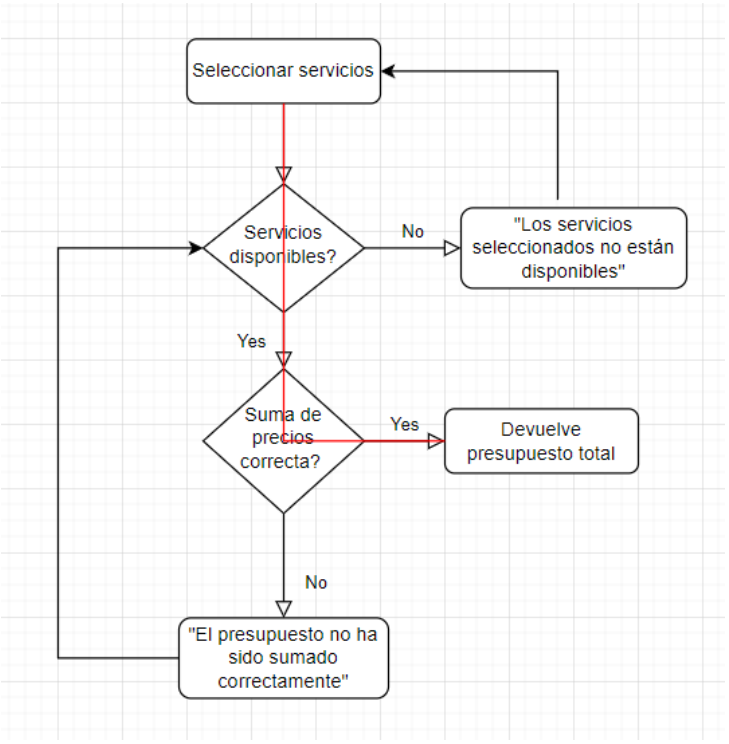
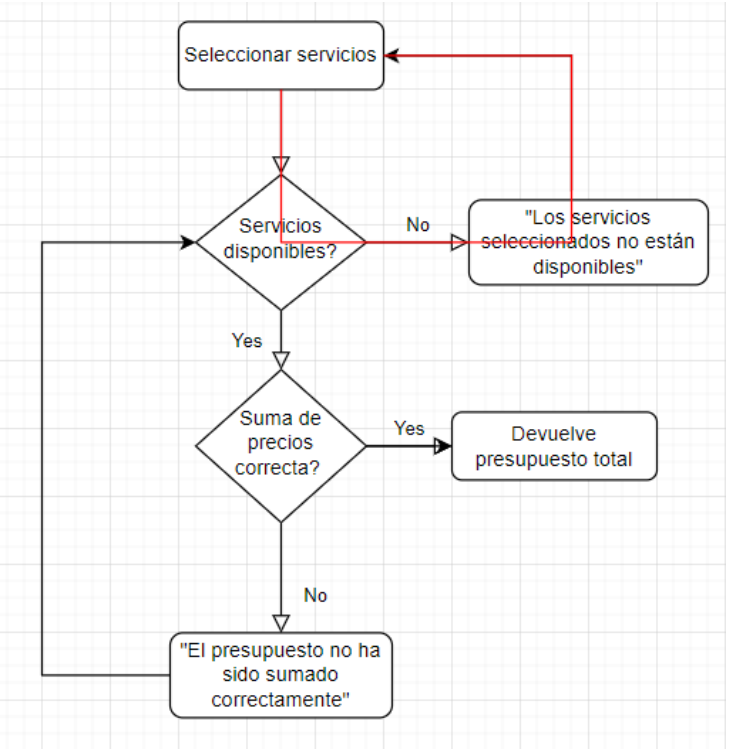

Login
Usuario correcto?
No
"El usuario es incorrecto"
Yes
Contraseña correcta?
No
"La contraseña es incorrecta"
Yes
Sesión iniciada correctamente

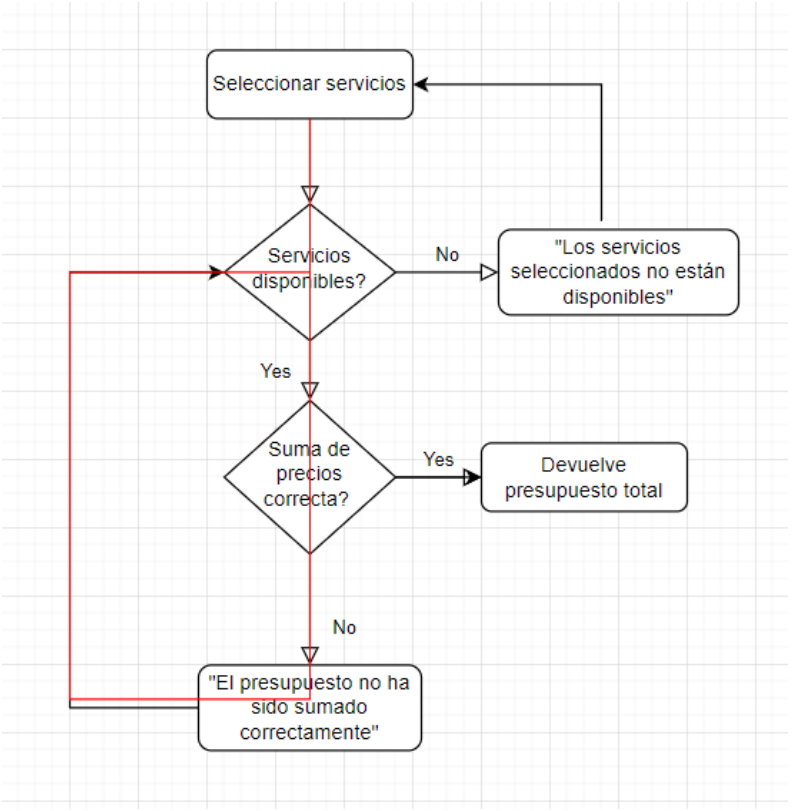
Diagrama de flujo (Clase Pedir Presupuesto)



- **Caminos básicos**




Seleccionar servicios
Servicios disponibles?
No
"Los servicios seleccionados no están disponibles"
Yes
Suma de precios correcta?
Yes
Devuelve presupuesto total
No
"El presupuesto no ha sido sumado correctamente"


Seleccionar servicios
Servicios disponibles?
No
"Los servicios seleccionados no están disponibles"
Yes
Suma de precios correcta?
Yes
Devuelve presupuesto total
No
"El presupuesto no ha sido sumado correctamente"