

# Testing - Tarefa

Sánchez Seoane, Pablo

Ciclo Superior de Desenvolvemento de Aplicacións Web (DAW)

## Índice

<b>1. Actividad Propuesta: Parte I</b>	<b>2</b>
1.1. Diseño de pruebas . . . . .	4
<b>2. Actividad Propuesta: Parte II</b>	<b>10</b>
2.1. Enlaces menú . . . . .	10
2.2. Funcionalidad Comprar . . . . .	12
2.3. Funcionalidad Comprar (Alternativas) . . . . .	14
2.4. Número de unidades a adquirir . . . . .	16
2.5. Seguridad Login: Inyección SQL . . . . .	18

## 1. Actividad Propuesta: Parte I

En esta actividad vamos a tratar de plantear una serie de pruebas de testing sobre una página web. En este caso he elegido Rakayo Clothing (1) porque lo he visto la recomendación en el foro. No se si algún compañero ha elegido la misma página, pero considero que puede ser interesante que los que estamos en el ciclo podamos poner a prueba a alumnos que han pasado por aquí.

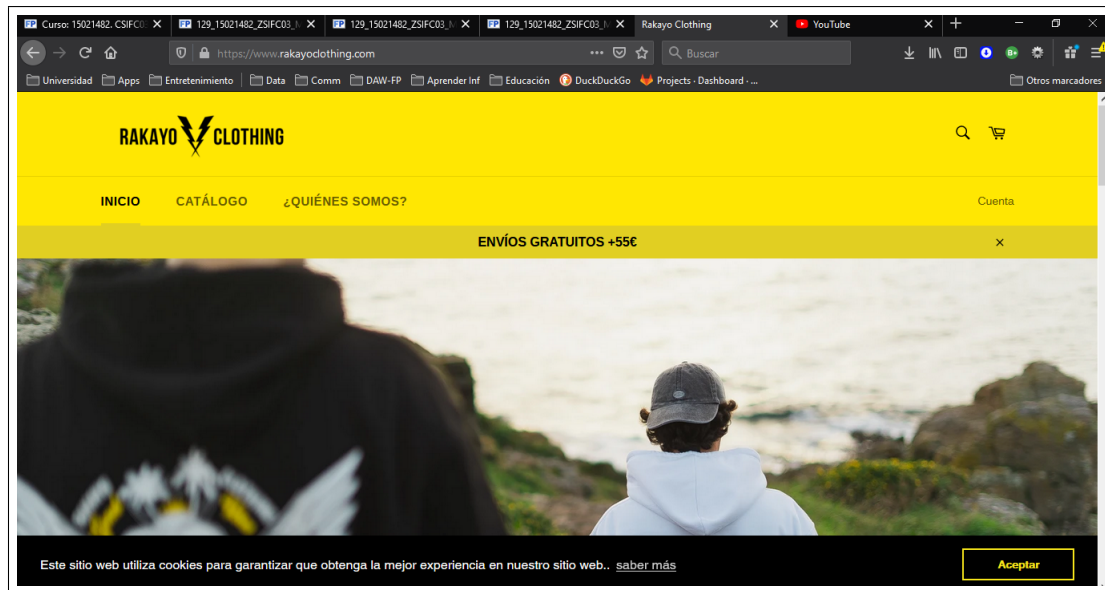


Figura 1: Página principal de Rakayo Clothing.

Como se puede observar, esta página dispone de un buscador de artículos localizado en la esquina superior derecha (2). Comprobando un poco el funcionamiento de esta he podido observar que las búsquedas realizadas no autocompletan palabras (3), ya sea porque utilizan etiquetas predefinidas o porque está diseñado para buscar palabras completas para cada artículo (separadas por espacios).



Figura 2: Herramienta de búsqueda.

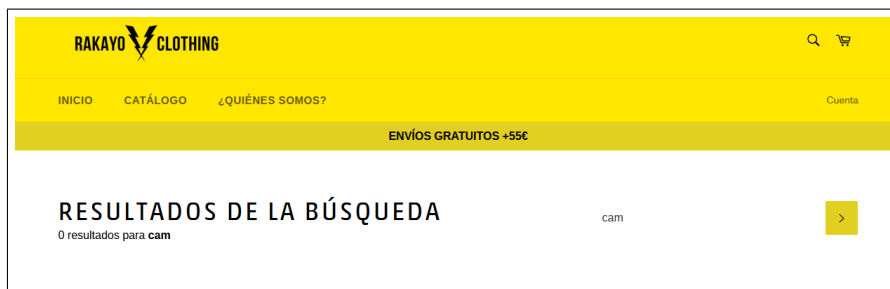


Figura 3: Limitaciones de la herramienta de búsqueda.

Otro elemento importante es el carrito de la compra (4). Como es habitual en esta sección se pueden observar todos los objetos añadidos a la compra que se desea realizar.

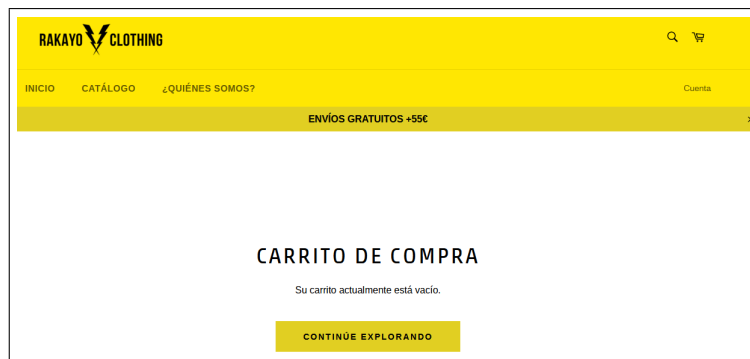


Figura 4: Carrito de la compra.

Además de esto otras secciones que aparecen son las de iniciar sesión, el catálogo y la información de la empresa. Otro de los aspectos más importantes es la sección de compra de artículos (5) que permite añadir al carrito o realizar la compra directamente.

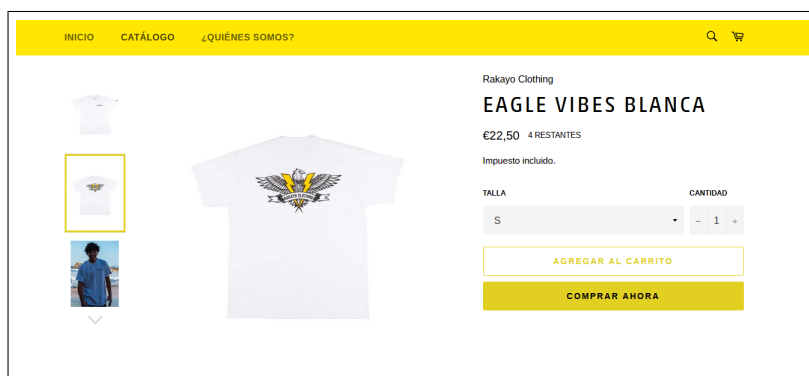


Figura 5: Comprar artículo.

## 1.1. Diseño de pruebas

En esta sección se mencionan todas las pruebas planteadas para probar la página web mencionada anteriormente. Para ello, se trata de plantear pruebas de diferentes tipos para poner a prueba diferentes aspectos. Las pruebas planteadas para esta aplicación web son las que se mencionan a continuación:

### ■ Pruebas Funcionales

- **Funcionalidad Comprar:** Pone a prueba la función de compra de un producto.
- **Funcionalidad Comprar (Alternativas):** Comprueba que la compra de productos individuales funcione igual utilizando la función del carrito que por compra directa.
- **Número de unidades a adquirir:** Comprueba los valores límite del número de unidades a adquirir, tanto cuando toma valores iguales o inferiores a cero como cuando supera el stock máximo.
- **Número de unidades a adquirir (no numérico):** Comprueba que la aplicación no acepte valores no numéricos (o números flotantes) como número de unidades a adquirir.
- **Eliminar productos (Admin) guardados en cestas:** Comprueba que sucede si un usuario añade productos a su cesta y antes de comprarlos el administrador elimina el producto o reduce su stock a 0.
- **Enlaces menú:** Esta prueba comprueba que todos los botones del menú, independientemente de la página de la aplicación, enlacen a la misma página.

### ■ Pruebas no funcionales

- **Desconexión de la red:** Pone a prueba desconexiones involuntarias de los usuarios comprobando si se mantiene el estado o se cierra la sesión.
- **Seguridad Login - Inyección SQL:** Con esta prueba se pretende comprobar que el sistema de autenticación es resistente a ataques que utilizan inyección de código SQL para obtener datos de la base de datos.

### ■ Pruebas de rendimiento

- **Funcionabilidad Comprar (Acceso múltiple):** Con esta prueba se pretende poner a prueba la aplicación cuando múltiples usuarios (de 10 - 1000) realizan una compra simultáneamente.
- **Tiempo Transacción:** Calcular el tiempo promedio que tarda en procesarse una solicitud de compra.

Caso de prueba: <b>Funcionalidad Comprar</b>
<b>Tipo de Prueba</b> Funcional - Basada en el usuario final
<b>Descripción</b> La aplicación debe permitir comprar productos directamente (sin utilizar la funcionalidad de la cesta).
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Comprar el producto.
<b>Resultado Esperado</b> Se realiza la transacción económica correctamente y se actualiza el stock.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Funcionalidad Comprar (Alternativas)</b>
<b>Tipo de Prueba</b> Funcional - Test Alternativas
<b>Descripción</b> La aplicación debe permitir comprar productos directamente desde un producto o bien añadirlo al carro y adquirirlo sin que existan diferencias.
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Comprar el producto. 4. Añadir el producto al carrito 5. Comprar el producto utilizando la función del carrito.
<b>Resultado Esperado</b> En ambos casos se realiza la transacción económica correctamente y se actualiza el stock, sin que existan diferencias entre ambos procesos.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Número de unidades a adquirir</b>
<b>Tipo de Prueba</b> Funcional - Test de valor límite
<b>Descripción</b> La aplicación no debe realizar transacciones cuando las unidades a adquirir toma valor igual o inferior a 0, y de igual forma cuando toma un valor superior al stock para ese producto.
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Insertar el número de unidades (igual o inferior a 0, o valores iguales o superiores al stock máximo de ese producto). 4. Comprar el producto.
<b>Resultado Esperado</b> La transacción no se puede realizar.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Número de unidades a adquirir (no numérico)</b>
<b>Tipo de Prueba</b> Funcional - Test de equivalencia
<b>Descripción</b> Al insertar como número de unidades caracteres no numéricos (o números flotantes) la transacción no se debe poder realizar.
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Insertar el número de unidades con valores alfabéticos o números flotantes. 4. Comprar el producto.
<b>Resultado Esperado</b> La transacción no se puede realizar.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Eliminar productos (Admin) guardados en cestas</b>
<b>Tipo de Prueba</b> Funcional - Test Ad-Hoc
<b>Descripción</b> Después de que un usuario añada un producto al carrito el administrador elimina este producto (o reduce el stock a 0), no debería ser posible realizar la transacción.
<b>Prerequisitos</b> Debe existir stock del producto a adquirir y ser modificable.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Añadir el producto al carrito. 4. Con permisos de administrador eliminar el producto (o reducir el stock a 0). 5. Comprar el producto desde la cesta.
<b>Resultado Esperado</b> La transacción no se puede realizar.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Enlaces menú</b>
<b>Tipo de Prueba</b> Funcional - Test Alternativas
<b>Descripción</b> El menú de la aplicación web debe remitir siempre a los mismos enlaces, independientemente de la página en que se encuentre.
<b>Prerequisitos</b> N/A
<b>Pasos</b> 1. Entrar en la página. 2. Para cada botón del menú, comprobar si desde las diferentes páginas de la aplicación (son pocas páginas) todas enlazan correctamente a la misma página.
<b>Resultado Esperado</b> Cada botón está enlazado a la misma dirección.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Desconexión de la red</b>
<b>Tipo de Prueba</b> No funcional - Test de Fiabilidad
<b>Descripción</b> Ante una desconexión de la red por parte de un usuario logeado (desconexión no deseada por el usuario), el sistema debería recuperar la conexión (en caso de ser cortes cortos) o desloguear al usuario.
<b>Prerequisitos</b> Cuenta de usuario.
<b>Pasos</b> 1. Entrar en la página. 2. Iniciar una sesión con una cuenta de usuario. 3. Generar un corte en la conexión que impida la comunicación con el servidor de la aplicación. 4. Recuperar la conexión con el servidor.
<b>Resultado Esperado</b> Si se trata de una desconexión corta el usuario debería recuperar la conexión con el servidor en el mismo punto sin problemas. En caso de una desconexión larga el sistema debería cerrar la sesión iniciada por el usuario.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Seguridad Login: Inyección SQL</b>
<b>Tipo de Prueba</b> No funcional - Test de Seguridad
<b>Descripción</b> Al iniciar sesión en la aplicación la entrada de datos debe analizar el tipo de datos que se están solicitando para evitar que se realicen solicitudes SQL en la base de datos sin que estas sean autorizadas.
<b>Prerequisitos</b> N/A
<b>Pasos</b> 1. Entrar en la página 2. Acceder a la sesión de iniciar la sesión. 3. Inyectar código SQL en alguno de los campos para iniciar sesión.
<b>Resultado Esperado</b> Acceso denegado.
<b>Resultado Obtenido</b>



Caso de prueba: <b>Funcionabilidad Comprar (Acceso múltiple)</b>
<b>Tipo de Prueba</b> Rendimiento - Test de picos
<b>Descripción</b> La aplicación debe ser capaz de responder al usuario en caso de realizar numerosas ventas simultáneas (entre 10 - 1000).
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto 3. Adquirir ese producto (simultaneamente por los usuarios).
<b>Resultado Esperado</b> Soporte de la aplicación para todos los usuarios y priorización del proceso de venta hasta fin de existencias.
<b>Resultado Obtenido</b>

Caso de prueba: <b>Tiempo Transacción</b>
<b>Tipo de Prueba</b> Rendimiento - Prueba de resistencia
<b>Descripción</b> Comprobar el tiempo que tarda la aplicación en realizar una transacción cuando un usuario realiza un pedido.
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entra en la página. 2. Seleccionar un producto. 3. Adquirir el producto. 4. Repetir el proceso y calcular el tiempo promedio que tarda en procesar la solicitud de compra.
<b>Resultado Esperado</b> Tiempo promedio bajo e incremento constante o lineal al adquirir un mayor número de productos.
<b>Resultado Obtenido</b>

## 2. Actividad Propuesta: Parte II

Para esta segunda tarea relacionada con el tema de testing vamos a tratar de implementar alguna de las pruebas que han sido planteadas en la parte anterior. Para ello utilizaremos la herramienta **Selenium WebDriver** utilizando como entorno de trabajo **Eclipse**, con lo que el código con el que trabajaremos será *Java*. De las propuestas anteriores se han seleccionado las 5 que son posibles realizar desde la posición actual.

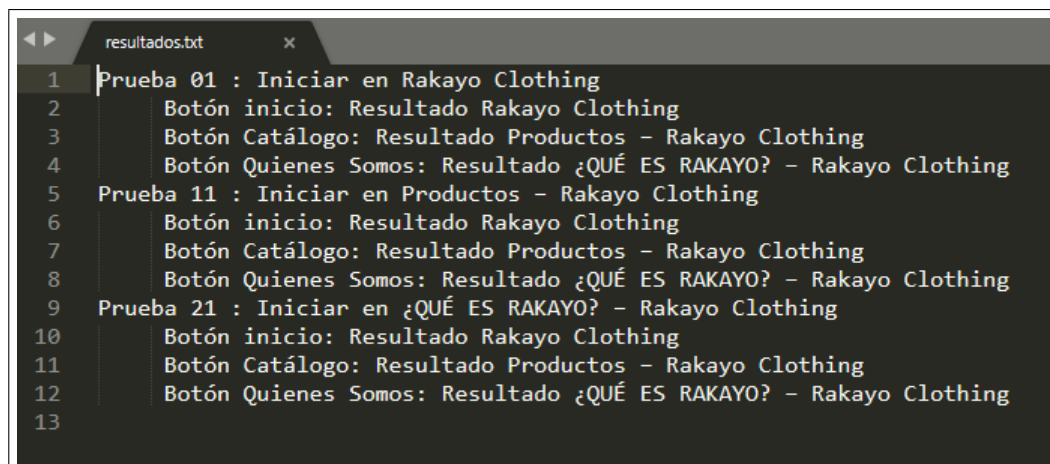
### 2.1. Enlaces menú

- **Enlaces menú:** Esta prueba comprueba que todos los botones del menú, independientemente de la página de la aplicación, enlacen a la misma página.

Esta prueba comprueba que las direcciones de los enlaces en la barra de menú estén correctamente vinculados. Para ello esta prueba se sitúa en cada una de las páginas que deberían estar adjuntadas en este menú (son 3 páginas) y pulsa sobre cada uno de los botones. Esta prueba realiza de forma secuencial las pruebas para cada una de las páginas y cada uno de los botones. Los resultados son almacenados en un documento de texto para poder analizar los resultados (6).

**Nota:** A pesar de que el código podría ser simplificado utilizando un segundo array con los nombres de los botones del menú, desconozco el motivo, siempre pulsaba el botón de *Inicio* independientemente de la llamada. Mis conocimientos en Java no son suficientemente extensos como para poder desarrollar una solución (al menos sin dedicar más tiempo del que dispongo en este momento).

Caso de prueba: <b>Enlaces menú</b>
<b>Tipo de Prueba</b> Funcional - Test Alternativas
<b>Descripción</b> El menú de la aplicación web debe remitir siempre a los mismos enlaces, independientemente de la página en que se encuentre.
<b>Prerequisitos</b> N/A
<b>Pasos</b> 1. Entrar en la página. 2. Para cada botón del menú, comprobar si desde las diferentes páginas de la aplicación (son pocas páginas) todas enlazan correctamente a la misma página.
<b>Resultado Esperado</b> Cada botón está enlazado a la misma dirección.
<b>Resultado Obtenido</b> Todos los enlaces llevan a la página esperada.



```
1 Prueba 01 : Iniciar en Rakayo Clothing
2     Botón inicio: Resultado Rakayo Clothing
3     Botón Catálogo: Resultado Productos - Rakayo Clothing
4     Botón Quienes Somos: Resultado ¿QUÉ ES RAKAYO? - Rakayo Clothing
5 Prueba 11 : Iniciar en Productos - Rakayo Clothing
6     Botón inicio: Resultado Rakayo Clothing
7     Botón Catálogo: Resultado Productos - Rakayo Clothing
8     Botón Quienes Somos: Resultado ¿QUÉ ES RAKAYO? - Rakayo Clothing
9 Prueba 21 : Iniciar en ¿QUÉ ES RAKAYO? - Rakayo Clothing
10     Botón inicio: Resultado Rakayo Clothing
11     Botón Catálogo: Resultado Productos - Rakayo Clothing
12     Botón Quienes Somos: Resultado ¿QUÉ ES RAKAYO? - Rakayo Clothing
13
```

Figura 6: Resultado de la prueba de los menús.

## 2.2. Funcionalidad Comprar

- **Funcionalidad Comprar:** Pone a prueba la función de compra de un producto.

Esta prueba trata de comprar una unidad de producto directamente desde la pestaña del artículo (sin añadirlo al carrito). Esta prueba no realizar una compra real, así que se considerará un éxito si muestra la pestaña de datos de compra.

Caso de prueba: <b>Funcionalidad Comprar</b>
<b>Tipo de Prueba</b> Funcional - Basada en el usuario final
<b>Descripción</b> La aplicación debe permitir comprar productos directamente (sin utilizar la funcionalidad de la cesta).
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Comprar el producto.
<b>Resultado Esperado</b> Se realiza la transacción económica correctamente y se actualiza el stock.
<b>Resultado Obtenido</b> Accede a la pantalla de pago correctamente (no ha sido posible testear la compra completa)

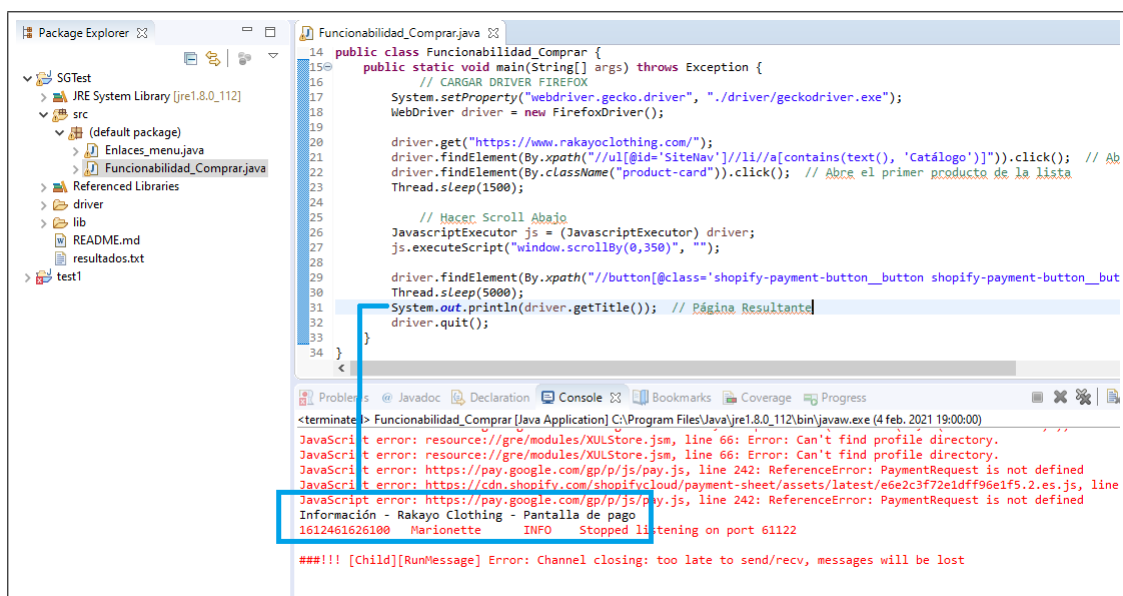


Figura 7: Resultado de la compra desde la pestaña de artículos.

### 2.3. Funcionalidad Comprar (Alternativas)

- **Funcionalidad Comprar (Alternativas):** Comprueba que la compra de productos individuales funcione igual utilizando la función del carrito que por compra directa.

Esta prueba trata de añadir una unidad de producto al carrito para posteriormente acceder a la pestaña del carrito y realizar la compra. Esta prueba no realiza una compra real, así que se considerará éxito si muestra la pestaña de datos de la compra.

Caso de prueba: <b>Funcionalidad Comprar (Alternativas)</b>
<b>Tipo de Prueba</b> Funcional - Test Alternativas
<b>Descripción</b> La aplicación debe permitir comprar productos directamente desde un producto o bien añadirlo al carro y adquirirlo sin que existan diferencias.
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Comprar el producto. 4. Añadir el producto al carrito 5. Comprar el producto utilizando la función del carrito.
<b>Resultado Esperado</b> En ambos casos se realiza la transacción económica correctamente y se actualiza el stock, sin que existan diferencias entre ambos procesos.
<b>Resultado Obtenido</b> Accede a la pantalla de pago correctamente (no ha sido posible testear la compra completa)

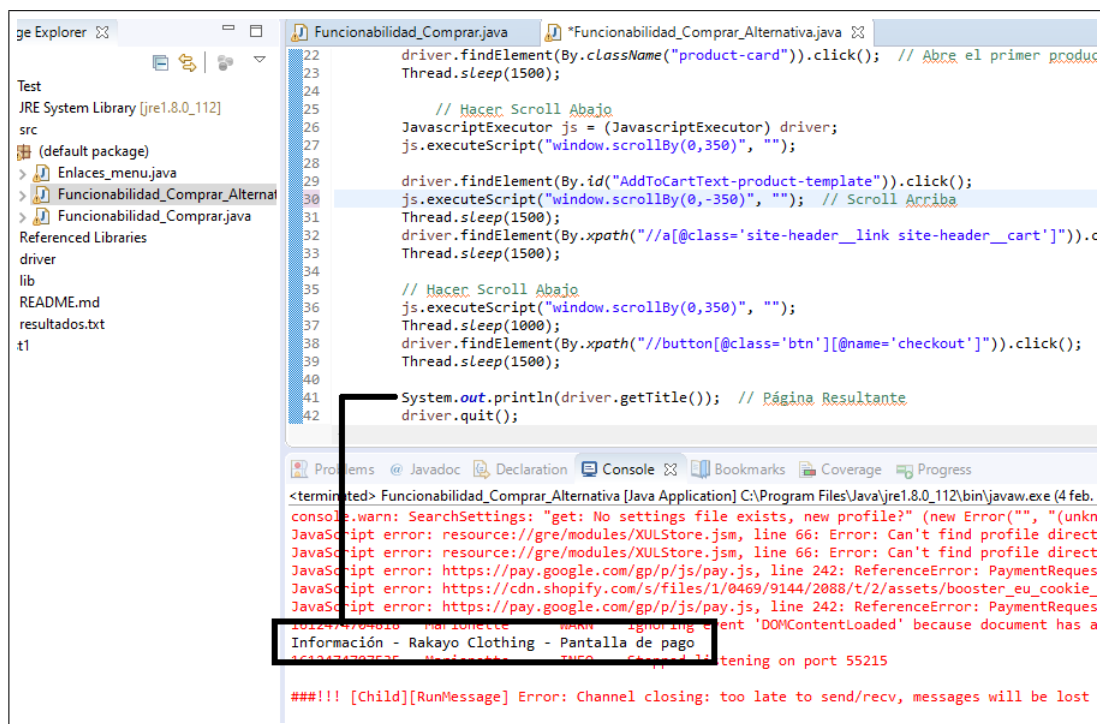


Figura 8: Resultado de la compra desde la pestaña de carrito.

## 2.4. Número de unidades a adquirir

- **Número de unidades a adquirir:** Comprueba los valores límite del número de unidades a adquirir, tanto cuando toma valores iguales o inferiores a cero como cuando supera el stock máximo.

Para esta prueba es necesario volver a acceder a una página de un producto. En esta ocasión vamos a añadir un producto al carrito, pero asignando un valor de 0, -1 y 10000 (muy superior al stock) en el número de unidades.

Caso de prueba: <b>Número de unidades a adquirir</b>
<b>Tipo de Prueba</b> Funcional - Test de valor límite
<b>Descripción</b> La aplicación no debe realizar transacciones cuando las unidades a adquirir toma valor igual o inferior a 0, y de igual forma cuando toma un valor superior al stock para ese producto.
<b>Prerequisitos</b> Debe existir stock del producto a adquirir.
<b>Pasos</b> 1. Entrar en la página. 2. Seleccionar un producto. 3. Insertar el número de unidades (igual o inferior a 0, o valores iguales o superiores al stock máximo de ese producto). 4. Comprar el producto.
<b>Resultado Esperado</b> La transacción no se puede realizar.
<b>Resultado Obtenido</b> No es posible realizar la compra porque no se añaden productos al carrito.

**Nota\*:** Durante la verificación de este proceso sucede algo curioso con el caso del valor 0. A pesar de que el carrito no almacena el producto, aparece una alerta de "Producto añadido al carrito". Esto es un fallo visual, pero no altera la funcionabilidad de la aplicación.



```
15 public class Numero_Unidades_Adquirir {
16     public static void main(String[] args) throws Exception {
17         // CARGAR DRIVER FIREFOX
18         System.setProperty("webdriver.gecko.driver", "./driver/geckodriver.exe");
19         WebDriver driver = new FirefoxDriver();
20
21         // PRECARGAR VALORES QUE VAN A SER UTILIZADOS
22         String [] unitValue = {"0", "-1", "10000"};
23
24         // PRUEBA
25         for(int i=0; i<unitValue.length; i++){ // El bucle permite probar con el valor 0 y con el valor negativo
26             // DESPLAZARSE A LA PÁGINA DE INTERÉS
27             driver.get("https://www.rakayoclothing.com/"); // Iniciar instancia del navegador (página Inicio)
28             driver.findElement(By.xpath("//ul[@id='SiteNav']/li//a[contains(text(), 'Catálogo')]")).click(); // A
```

<terminated> Numero\_Unidades\_Adquirir [Java Application] C:\Program Files\Java\jre1.8.0\_112\bin\javaw.exe (5 feb. 2021 11:37:35)

console.warn: SearchSettings: "get: No settings file exists, new profile?" (new Error("", "(unknown module)"))

JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile directory.

JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile directory.

JavaScript error: https://pay.google.com/gp/p/is/pay.js, line 236: ReferenceError: PaymentRequest is not defined

No hay elementos en el carrito, no se puede comprar.

Tu carrito - Rakayo Clothing

JavaScript error: https://pay.google.com/gp/p/is/pay.js, line 236: ReferenceError: PaymentRequest is not defined

No hay elementos en el carrito, no se puede comprar.

Tu carrito - Rakayo Clothing

JavaScript error: https://pay.google.com/gp/p/is/pay.js, line 236: ReferenceError: PaymentRequest is not defined

No hay elementos en el carrito, no se puede comprar.

Tu carrito - Rakayo Clothing

1012321499711 Marionette INFO Stopped listening on port 58465

JavaScript error: resource://activity-stream/lib/ASRouter.jsm, line 987: NS\_ERROR\_ILLEGAL\_VALUE: Component returned fail

####!!! [Child][RunMessage] Error: Channel closing: too late to send/recvd, messages will be lost

Figura 9: Resultado de añadir números no válidos al comprar unidades de producto.

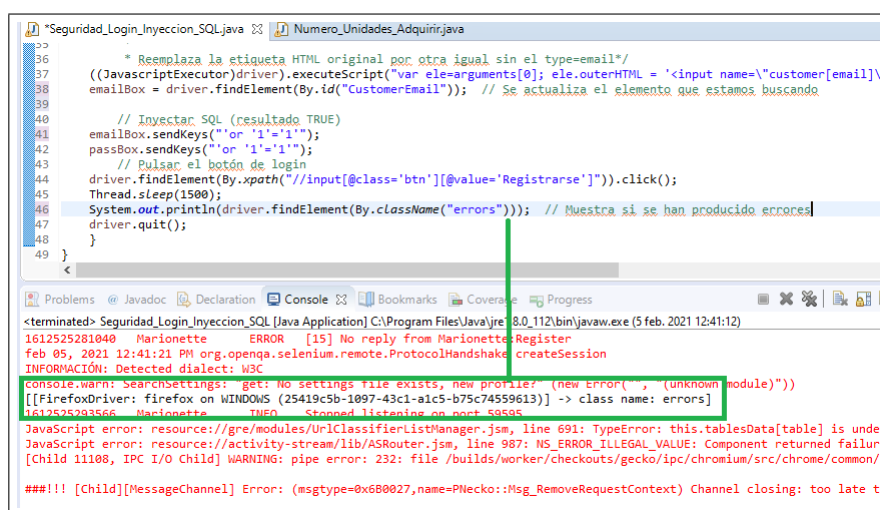
## 2.5. Seguridad Login: Inyección SQL

- **Seguridad Login - Inyección SQL:** Con esta prueba se pretende comprobar que el sistema de autenticación es resistente a ataques que utilizan inyección de código SQL para obtener datos de la base de datos.

En esta ocasión puede ser necesario una mayor explicación, ya que el código no es tan sencillo como los anteriores. De forma similar a los casos anteriores el programa comienza cargando el driver de *Firefox* y posteriormente inicia una instancia accediendo a la página de inicio de la plataforma. Navegamos hasta alcanzar la pestaña de **cuenta**.

- En primer lugar identificamos los elementos de interés (los *inputs* para el correo electrónico y la contraseña).
- En el caso del **correo electrónico** la etiqueta *input* incorpora un atributo 'type=email' lo que impide enviar el formulario si no se utiliza un formato de correo electrónico.
- La solución a este problema pasa por modificar la etiqueta HTML para eliminar el atributo *type*. A pesar de mis limitaciones en conocimiento de java he podido encontrar una respuesta en Stackoverflow que requiere de llamadas a JS durante la ejecución. Con algunas adaptaciones he podido modificar la etiqueta por otra idéntica sin el atributo *type*.
- Finalmente es necesario actualizar el objeto *emailBox* para que detecte los nuevos cambios realizados en la etiqueta.

El resto del código es relativamente sencillo. Para cada uno de los campos se escribe 'or '1'='1'. En este caso lo que se trata es de que realice la consulta e incorpore este elemento. En caso afirmativo 1 = 1 sería **True**, por lo que independientemente del usuario o la contraseña se aceptaría el login. En caso de que se muestre un error de logueo aparece un error, que es el que trata de identificar al buscar el resultado.



```
36 * Reemplaza la etiqueta HTML original por otra igual sin el type=email/*
37 ((JavascriptExecutor)driver).executeScript("var ele=arguments[0]; ele.outerHTML = '<input name='\"customer[email]\"'
38 emailBox = driver.findElement(By.id(\"CustomerEmail\")); // Se actualiza el elemento que estamos buscando
39
40 // Inyectar SQL (resultado TRUE)
41 emailBox.sendKeys("'or '1'='1'");
42 passBox.sendKeys("'or '1'='1'");
43 // Pulsar el botón de login
44 driver.findElement(By.xpath("//*[@class='btn'][@value='Registrarse']")).click();
45 Thread.sleep(1500);
46 System.out.println(driver.findElement(By.className("errors"))); // Muestra si se han producido errores
47 driver.quit();
48
49 }
```

Problems | Javadoc | Declaration | Console | Bookmarks | Coverage | Progress

<terminated> Seguridad\_Login\_Inyeccion\_SQL [Java Application] C:\Program Files\Java\jre-8.0.112\bin\javaw.exe (5 feb. 2021 12:41:12)

1612525281040 Marionette ERROR [15] No reply from MarionetteRegister  
feb 05, 2021 12:41:21 PM org.openqa.selenium.remote.ProtocolHandshake createSession  
INFORMACIÓN: Detected dialect: W3C

console.warn: SearchSettings: GET: NO settings file exists, new provider (new Error(" (unknown module)"))  
[[{"FirefoxDriver": "firefox on WINDOWS (25419c5b-1097-43c1-a1c5-b75c74559613)"}] -> class name: errors]  
1612525293566 Marionette INFO Stopped listening on port 5956

JavaScript error: resource://gre/modules/UrlClassifierListManager.jsm, line 691: TypeError: this.tablesData[table] is undefined  
JavaScript error: resource://activity-stream/lib/ASRouter.jsm, line 987: NS\_ERROR\_ILLEGAL\_VALUE: Component returned failure  
[Child 11108, IPC I/O Child] WARNING: pipe error: 232: file /builds/worker/checkouts/gecko/ipc/chromium/src/chrome/common/i  
###!!! [Child][MessageChannel] Error: (msgtype=0x6B0027,name=PNecko::Msg\_RemoveRequestContext) Channel closing: too late to

Figura 10: Resultado de añadir números no válidos al comprar unidades de producto.

Caso de prueba: <b>Seguridad Login: Inyección SQL</b>
<b>Tipo de Prueba</b> No funcional - Test de Seguridad
<b>Descripción</b> Al iniciar sesión en la aplicación la entrada de datos debe analizar el tipo de datos que se están solicitando para evitar que se realicen solicitudes SQL en la base de datos sin que estas sean autorizadas.
<b>Prerequisitos</b> N/A
<b>Pasos</b> 1. Entrar en la página 2. Acceder a la página de iniciar la sesión. 3. Inyectar código SQL en alguno de los campos para iniciar sesión.
<b>Resultado Esperado</b> Acceso denegado.
<b>Resultado Obtenido</b> Acceso denegado, no se permite la inyección de código.