



DESCRIPCIÓN BREVE

Se realiza el informe de la automatización de 4 escenarios del sistema

[Ariagna Albuerne](#)

Taller de automatización del Testing Funcional

Contenido

INTRODUCCIÓN	2
DESARROLLO	3
Configuración POM Del proyecto	8
Parametrización.....	8
Indicaciones para ejecución con clave y contraseña.....	9
Herramientas Utilizadas	12
CONCLUSIONES	12

INTRODUCCIÓN

En el informe se muestra los elementos que se tuvieron en cuenta para darle solución a la consigna propuesta 'Implementar los scripts de prueba de la aplicación eGroupWare para los siguientes escenarios:


- Dar de alta un proyecto
- Consultar los datos de un proyecto
- Asignar recursos a un proyecto
- Agregar contactos a la agenda

DESARROLLO

Se agregan los guiones a grandes rasgos para que se entiendan mejor los flujos automatizados.

1. Dar de alta un proyecto

Guión

1. **IniciarSesion ()**
2. Clic en el menú project Manager
3. Clic en el botón  (Add a new project)
4. Verificar que estamos en la ventana Egroupware [project Manager - Add project]
5. Ingresar título al proyecto "Proyecto de prueba"
6. GenerarIdProyecto()
7. Ingresar id del proyecto
8. Clic en Save
9. Verificar el mensaje "Project saved"
10. Verificar el mensaje Project saved
11. BuscarProyectoCreado()
12. EliminarProyectoCreado()
13. Verificar que se haya eliminado
14. Cerrar la pestaña Project Manager

2. Consultar los datos de un proyecto

Guión

1. **IniciarSesion ()**
2. CrearProyecto()
3. BuscarProyectoCreado()
4. Verificar que todos los datos coincidan con el proyecto creado para consulta
5. EliminarProyectoCreado()
6. Cerrar la pestaña Project Manager


3. Asignar recursos a un proyecto

Guión

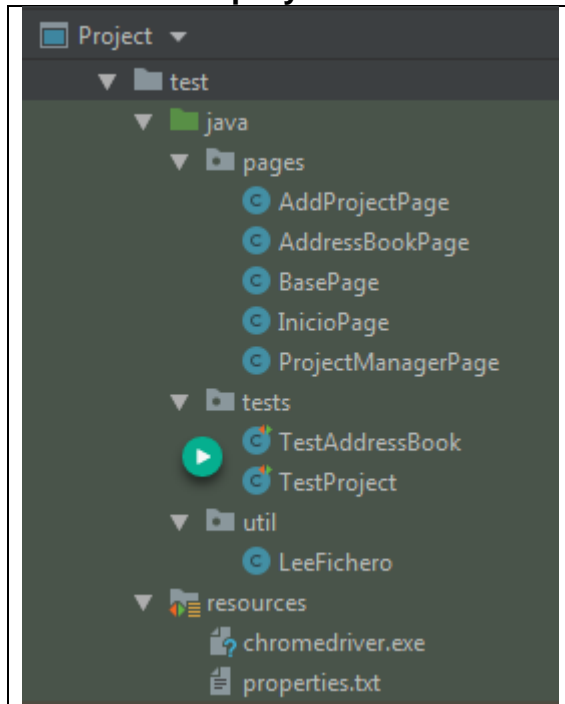
1. **IniciarSesion ()**
2. CrearProyecto()
3. BuscarProyectoCreado()
4. Clic derecho en el proyecto
5. Seleccionar la opción open
6. Verificar que estamos en la ventana de edición
7. Ir hasta miembros y agregar un <miembro >
8. Clic en el botón sabe
9. BuscarProyectoCreado()
10. Verificar que se le haya agregado el recurso miembro
11. EliminarProyectoCreado()
12. Cerrar la pestaña Project Manager

4. Agregar contactos a la agenda

Guion:

1. **IniciarSesion ()**
2. En la barra lateral clic en el menú Address Book
3. Clic en el botón  (Add a new contact)
4. Verificar que estamos en la ventana Egroupware [Address Book]
5. Escribir datos nombre <nombre>; <prefix>
6. Clic en el botón ok
7. Escribir <movil>;<mail>;<prefix>;<bussPhones>;<job>;<organizations>;<pais>;
8. Clic en save
9. Verificar el mensaje Contact saved
10. **BuscarContacto ()**
11. En el resultado
verificar >;<movil>;<mail>;<prefix>;<bussPhones>;<job>;<organizations>;<pais>;
12. EliminarContacto()
13. Cerrar la pestaña **Address Book**

Estructura del proyecto



Se realizó una clase Base donde se inicializa el Webdriver una única vez así como objetos de selenium WebDriver, se crean métodos genéricos de acciones para emplear en las clases que extienden de ella.

```
package pages;
```

```
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.support.ui.Select;
import java.util.Set;
import org.openqa.selenium.Keys;
```

```
public class BasePage {
```

```
    protected static WebDriver driver;
    private static WebDriverWait wait;
    private static By messageSave;
```

```
    static{
        System.setProperty("webdriver.chrome.driver", "../src/test/resources/chromedriver.exe");
        ChromeOptions chromeOptions = new ChromeOptions();
        driver = new ChromeDriver(chromeOptions);
        wait = new WebDriverWait(driver, 20);
        messageSave = By.id("egw_message");
    }
```

```
    public BasePage ( WebDriver driver ){
        BasePage.driver = driver;
        wait = new WebDriverWait(driver, 20);
    }
```

```
    public static void navigateTo ( String url ){
        driver.get(url);
    }
```

```

}

public static void closedBrowser () {
    driver.quit();
}

private WebElement find ( By locator ) {
    WebElement elementWithWait = wait.until( ExpectedConditions.visibilityOfElementLocated( locator ) );
    return elementWithWait;
}

public void write ( By locator , String textToWrite ) {
    find( locator ).clear();
    find( locator ).sendKeys( textToWrite );
}

public void click ( By locator ) {
    find( locator ).click();
}

public String getText ( By locator ) {
    return find( locator ).getText();
}

public void clickJs ( By locator ) {
    WebElement waitUntil = wait.until( ExpectedConditions.visibilityOfElementLocated( locator ) );
    JavascriptExecutor js = (JavascriptExecutor) driver;
    js.executeScript( "arguments[0].click();" , waitUntil );
}

public String winHandles ( int positionWindow ) {
    Set<String> windows = driver.getWindowHandles();
    return (String) windows.toArray()[ positionWindow ];
}

public String getTitle () {
    return driver.getTitle();
}

public void switchToVentana ( String IdVentana ) {
    driver.switchTo().window( IdVentana );
}

public boolean elementIsDisplayed ( By locator ) {
    return find( locator ).isDisplayed();
}

public void enter ( By selector ) {

```

```

new WebDriverWait( driver , 20 ).until( ExpectedConditions.elementToBeClickable( selector ) ).click();
find( selector ).sendKeys( Keys.ENTER );
}

public void clicRigthandMove ( By locator , By locatorMove ){

    WebElement b = find( locator );
    Actions action = new Actions( driver );
    action.contextClick( b ).build().perform();
    WebElement move = find( locatorMove );
    action.moveToElement( move );
    move.click();
}

public String messageGetText (){

    String mensaje = find( messageSave ).getText();
    click( messageSave );
    return mensaje;
}

public void dobleClick ( By locator ){

    Actions action = new Actions( driver );
    WebElement doblec = find( locator );
    action.doubleClick( doblec ).build().perform();
}

public void selectOption ( By locator , String value ){

    Select lista = new Select( find( locator ) );
    lista.selectByValue( value );
}
}

```

Se creó clases para realizar los métodos sobre los elementos web de cada clase:

ProjectManagerPage

AddProjectPage

AddressBookPage

InicioPage

Se agregó un archivo properties donde se carga las credenciales y la url.

En el paquete tests

Se crearon una clase para los test relacionados con proyecto y otra para los test de los contactos.

TestAddressBook

TestProject

Luego existe una clase útil para la lectura de properties.txt

Configuración POM Del proyecto

Se realiza la configuración del POM para definir las dependencias a utilizarse:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>tarea4</groupId>
  <artifactId>tarea4</artifactId>
  <version>1.0-SNAPSHOT</version>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependencies>

    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>3.141.59</version>
    </dependency>
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-chrome-driver</artifactId>
      <version>3.141.59</version>
    </dependency>

    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>5.8.0-M1</version>
      <scope>test</scope>
    </dependency>

  </dependencies>
</project>
```

Estructura de las clases Test Paso el ejemplo de la de TestAddressBook

Parametrización

Con respecto a la parametrización se realizó la función crearContacto pasándole los datos de contacto, y estos se cargan desde el archivo properties de esta manera se puede cambiar los datos de prueba y las validaciones no están realizadas hardcode sino comparando con esas variables que se cargan desde el archivo.

Se parametrizó también el agregar recursos, pudiéndose agregar siempre que cumpla con el formato y se encuentre en el sistema Apellido, Nombre, Así como los nombres de los proyectos para los demás casos.

Esta solución es un poco estática porque se trabaja con la posición de la lista en otras versiones se piensa agregar validaciones de la clave antes de usar el valor.

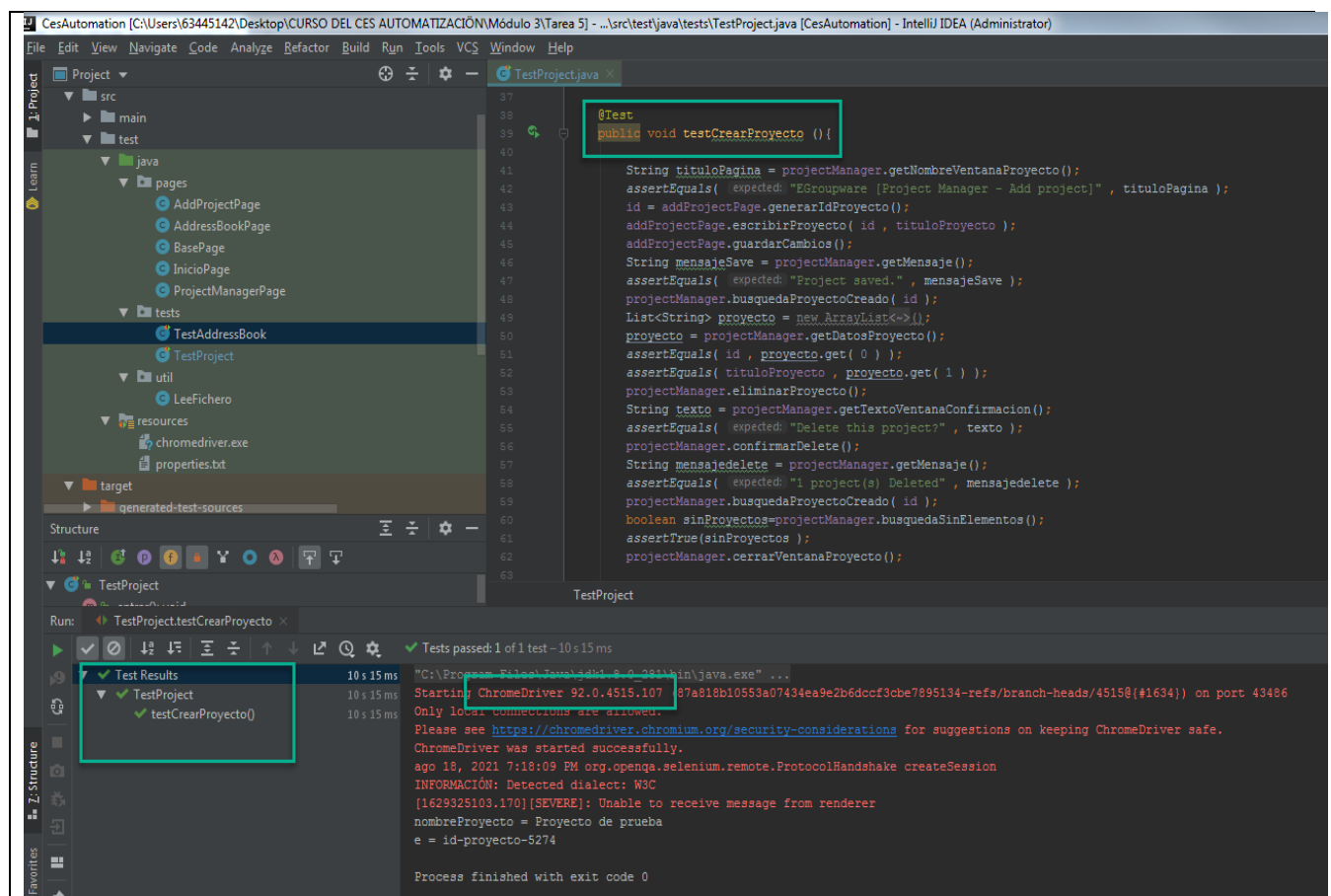
Indicaciones para ejecución con clave y contraseña.

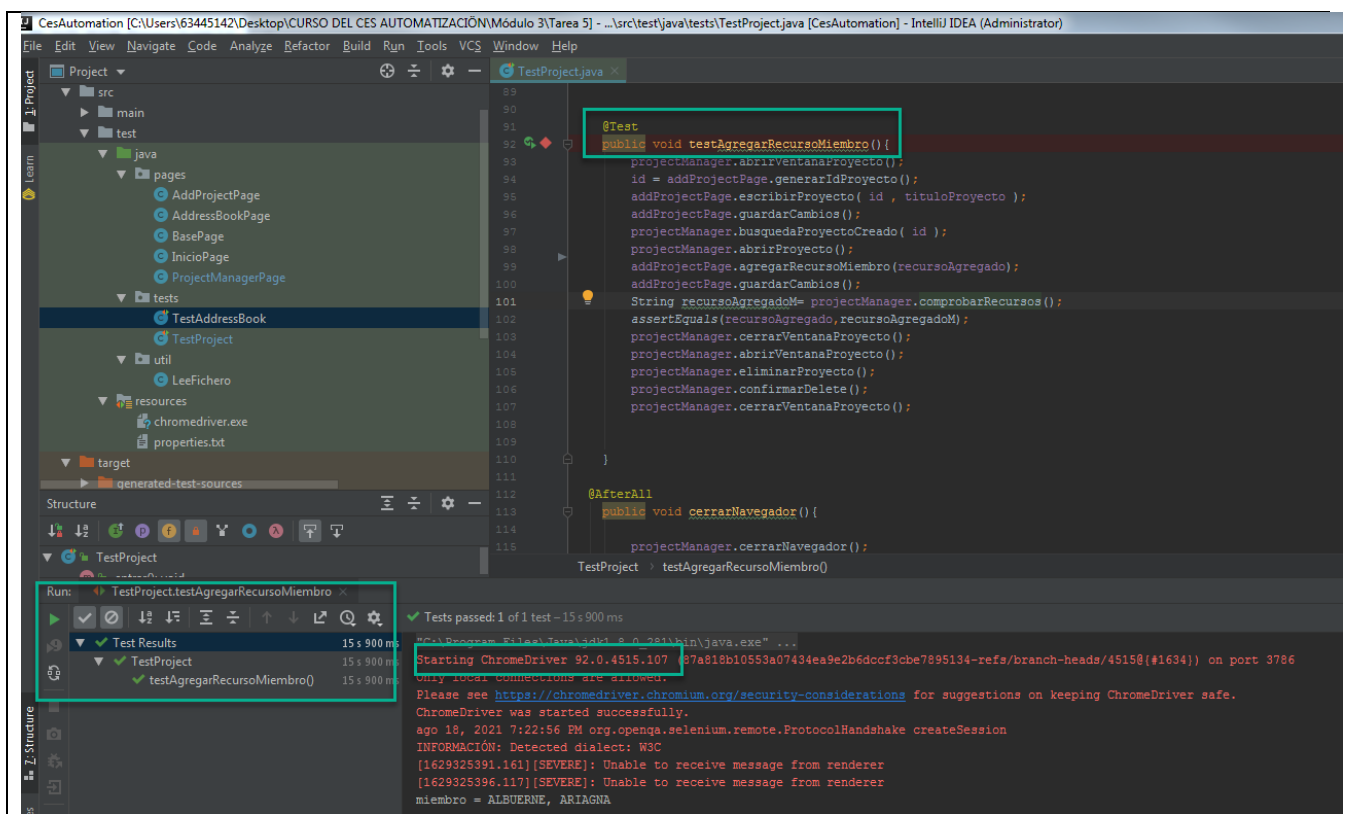
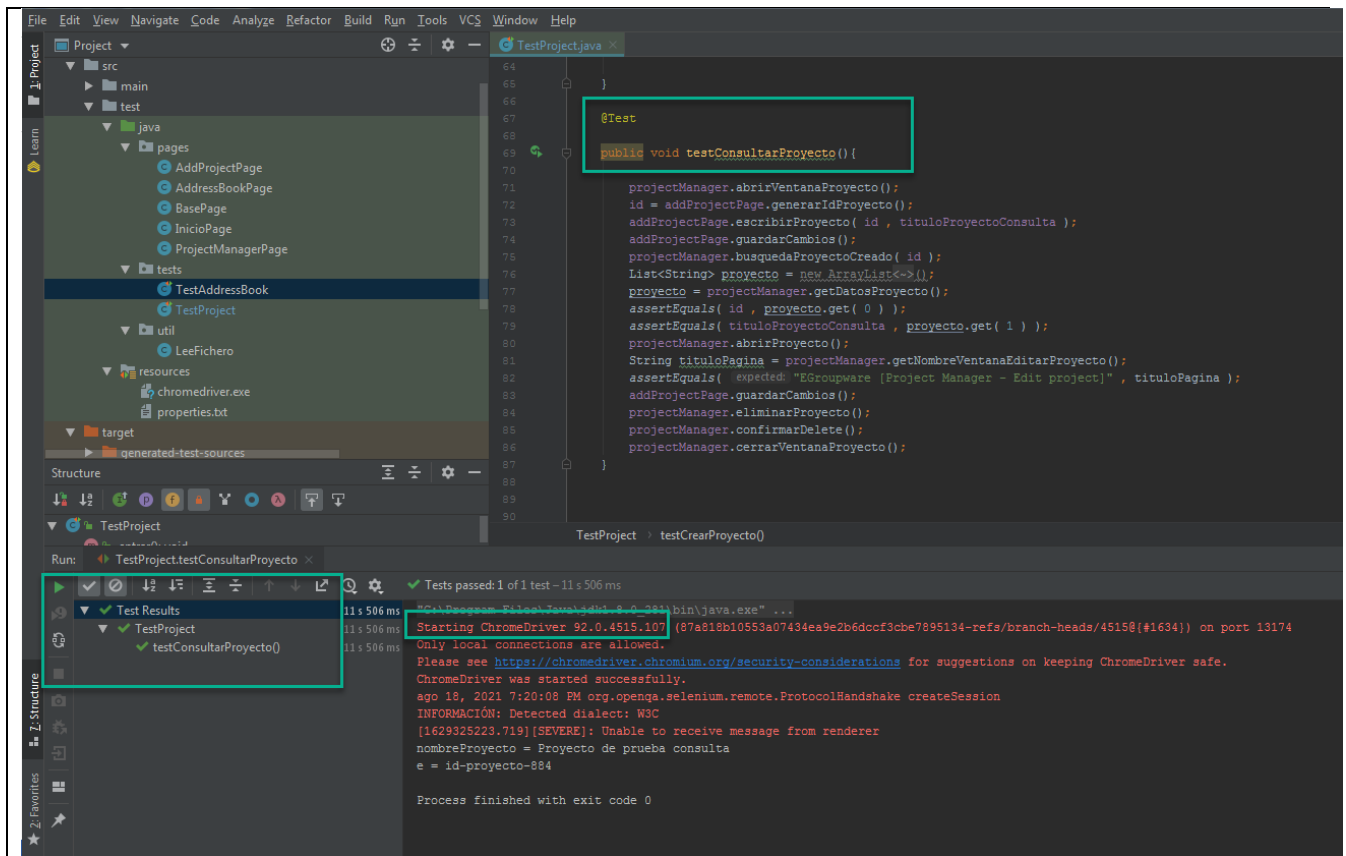
Se entrega un fichero leeme.txt con las indicaciones.

Estructura del proyecto

Código completo se encuentra en git: <https://github.com/aalbuerner87/tareaCesSelenium>

Captura del test pasando y con la versión de Chrome utilizada.





Herramientas Utilizadas

- IntelliJ IDEA Community Edition
- Java versión 1.8.0_281
- Maven
- JUnit 5 (junit-jupiter-engine 5.8.0-M1)
- Selenium (3.141.59)

CONCLUSIONES

Con la realización de este ejercicio se pudo comprobar que:

- 1- Que si bien es útil la herramienta Selenium IDE para obtener selectores, hay que usar las herramientas dev tool del navegador comprobar la unicidad del selector o en mucho de los casos construirlos, y depende de la situación a veces es mejor usar un tipo de selector u otro, más complejo cuando son dinámicos o están en elementos que desaparecen.
- 2- La estructura, organización y nombre de los métodos te das cuenta de la importancia cuando el proyecto va creciendo un poco y puedes reutilizarlos y encontrar de forma rápida lo que necesitas cuando tienes que hacer un cambio.