|  |  |  |  |
| --- | --- | --- | --- |
| Autor | Fecha | Descripción Cambio | Versión |
| Gabriel Herrera Zarate | 27-04-2018 | Creación documento | 1.0 |

**AUTOMATIZACIÓN DE PRUEBAS**

**TODAS LOS ARCHIVOS Y DRIVER DE LOS QUE SE HABLA EN EL DOCUMENTO SE ENCUENTRA EN LA SIGUIENTE RUTA** [**http://co1p222a/sitios/Core/GERSISOP/HerrmientasDF/\_layouts/15/start.aspx#/**](http://co1p222a/sitios/Core/GERSISOP/HerrmientasDF/_layouts/15/start.aspx#/) **DENTRO DE LA CARPETA Utilitarios/Automatizacion,**

Para realizar la automatización de los casos de pruebas en las aplicaciones se utilizar la librería de Selenium disponible para Visual Studio Community 2015, en el siguiente documento se explicará cómo implementar dicha librería y también, cómo utilizar el proyecto estándar creado con el fin de que se pueda utilizar en futuras aplicaciones.

**Cargar origen de datos**

Este proyecto se encarga de cargar un archivo plano el cual contiene toda la información que será usada en los formularios de la aplicación en estudio, dicho archivo, consiste en un CSV separado por comas, y tiene la siguiente estructura:

Campo1,Campo2,Campo3,Campo4…CampoN

Valor1,Valor2,Valor3,Valor4…ValorN

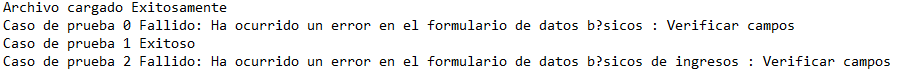
Donde campo representa el nombre de la columna y en la parte inferior se encuentras los valores de dichos campos, lo que quiere decir, que cada fila que se encuentre por debajo de los nombres de los campos corresponden a un caso de prueba.



**Figura 1. Estructura archivo origen de datos**

**Escribir log**

Este proyecto se encarga de generar un archivo txt en una ruta parametrizada en el archivo de configuración, este archivo contiene el resultado de los casos de prueba ejecutados en la aplicación y tiene la siguiente estructura:



**Figura 2. Estructura archivo de log**

**Funcionalidades**

Cuando se esté automatizando los casos de prueba de una página se usarán muchas veces las funcionalidades encontradas en este proyecto, cada una de ellas ayudará a que la automatización se realice de manera más fácil.

**MÉTODO IniciarPrueba():** Es el método por el cual empieza toda la automatización, se encarga de cargar el archivo plano, crear el driver de chrome por cada caso de prueba encontrado en el origen de datos.

**MÉTODO Retardo():** Es el método encargado de agregar tiempos de espera, generalmente se usa para cuando se hace cambio de formulario, estos tiempos vienen dados en segundos.

Por ejemplo:

Metodo.Retardo(2); 🡪 Retardo de de 2 segundos

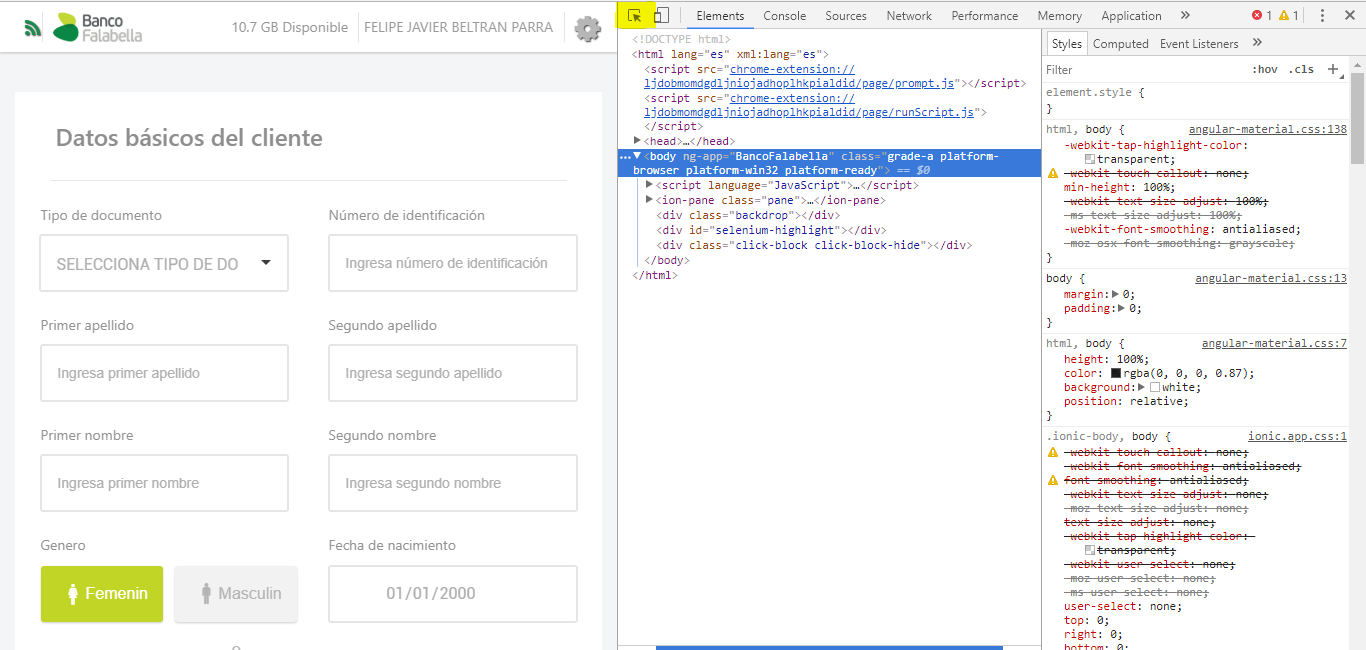
Si se desea agregar el multiplicador para que varíe en función del valor parametrizado en el archivo de configuración (Ver archivo de configuración en la parte inferior) se pone la siguiente manera:

Metodo.Retardo(2\*TiempoEspera); 🡪 Retardo de de 2 segundos \* el multiplicador

**MÉTODO Controlador():** Se encarga de ejecutar los métodos dentro de una sección de try-catch para controlar los errores que puedan ocurrir.

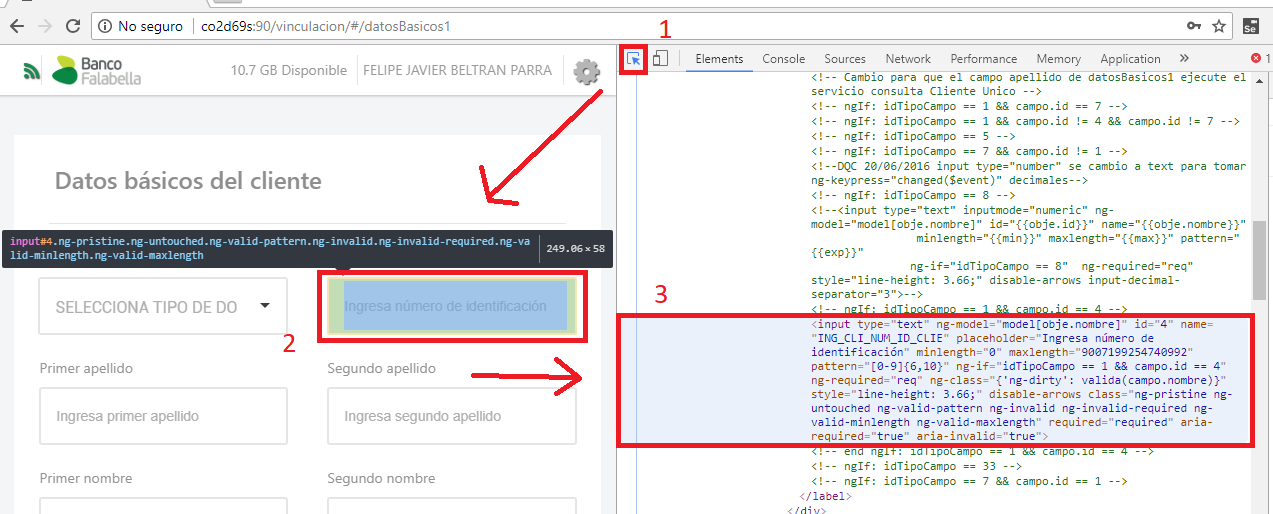
**MÉTODO EncontrarElemento():** Para automatizar una aplicación, siempre se requiere ubicar los elementos del HTML de la página o formulario, ya que así es la única forma de poder simular un usuario, el cual normalmente inserta valores en un input, escoge un valor de una lista desplegable, desplaza la página hasta la parte inferior, oprime un botón , etc.

De esta manera, se hace necesario establecer una relación entre el código y el HTML, para esto, el método EncontrarElemento() creará esta conexión, lo primero que se debe hacer es inspeccionar los elementos de la página oprimiendo F12, el cual mostrará una ventana como la de la figura 3.



**Figura 3. Inspección de elementos de la página**

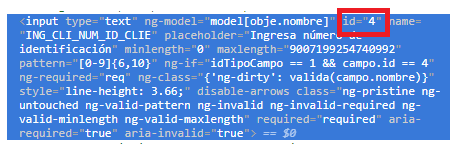
Para ubicar el elemento se debe seleccionar el botón que se resalta con amarillo en la figura 3, una vez seleccionado se da clic en el elemento al cual se quiera obtener información, por ejemplo, si se quiere encontrar información sobre el input Número de identificación se ejecutan los pasos de la figura 4.



**Figura 4. Inspección de elementos de la página**

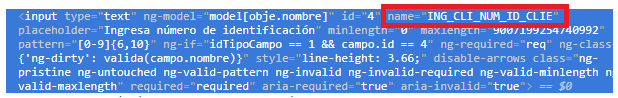
Se debe tener en cuenta la información del recuadro 4, ya que ahí se puede encontrar la forma de identificar cada elemento, los cuales puede ser por:

**Id ->**



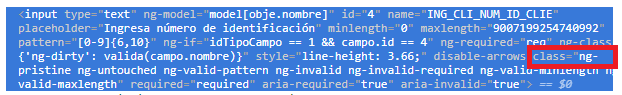
**Figura 5 Encontrar elemento por Id**

**Name ->**



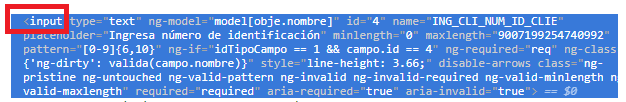
**Figura 6. Encontrar elemento por name**

**ClassName ->** Los ClassName no deben ser compuestos



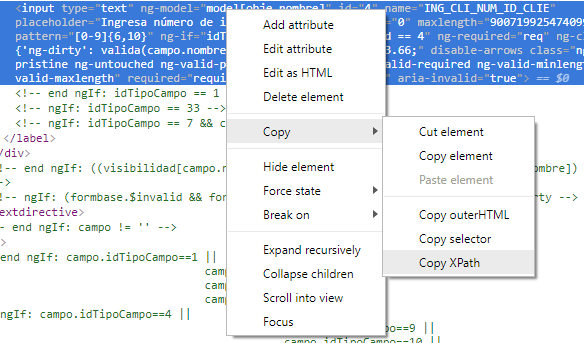
**Figura 7. Encontrar elemento por ClassName**

**TagName ->**



**Figura 8. Encontrar elemento por TagName**

**XPath ->**



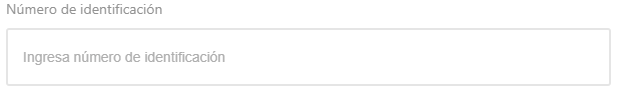
**Figura 9. Encontrar elemento por XPath**

Se da click derecho->Copy->Copy XPath, el cual vendría siendo el siguiente:

//\*[@id="4"]

Una vez se conoce como identificar los elementos en le HTML, se procede a explicar la forma de encontrarlos mediante el método ubicado en funcionalidades.

**Insertar datos en input:**



**Figura 10. Input entrada de datos**

Para insertar datos en un input el método se utiliza de la siguiente manera:

**EncontrarElemento(Atributo, Nombre atributo, valor)**

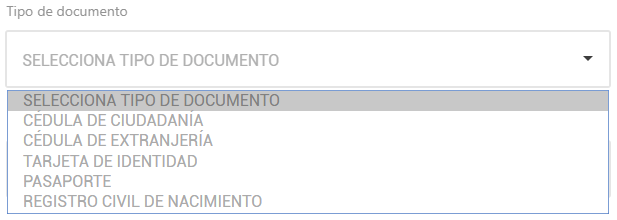
Dónde:

* **Atributo:** Id, Name, ClassName, TagName o XPath dependiendo por donde lo quiera identificar.
* **Nombre atributo:** Nombre que aparece en comillas para los Id, name y ClassName, nombre del TagName o el XPath copiado.
* **Valor:** Campo numérico o varchar, dependiendo la información que solicite el input.

**Ejemplo:**

EncontrarElemento(“Id”,”4”,”1096216579”) y el método se encarga de poner el valor en el formulario de la página.

**Escoger valor de lista desplegable:**



**Figura 11. Lista desplegable**

Pueden existir varias listas desplegables en el formulario, para escoger el valor cada una, se debe enviar la posición en la que se encuentre dicha lista.

**Ejemplo:**

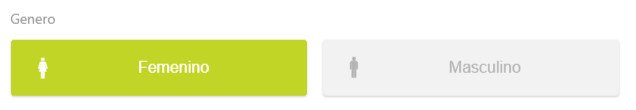
* Si existe solo una lista desplegable en el formulario:

EncontrarElemento(“Select”,”0”,”PASAPORTE”)

* Si existe otra lista desplegable se debe poner el siguiente número consecutivo y así sucesivamente.

EncontrarElemento(“Select”,”1”,”PASAPORTE”)

**Encontrar Botón:**



**Figura 12. Botones**

Para encontrar botones sucede lo mismo que con las listas desplegables, pueden haber muchos en el formulario entonces también se debe definir la posición en la que se encuentra cada botón.

**IWebElement NombreBoton =EncontrarElemento(“TagName”,”Button”,”Posicion”);**

**Ejemplo:**

Para seleccionar el botón “Femenino” de la figura 12 se hace lo siguiente:

IWebElement BotonFemenino =EncontrarElemento(“TagName”,”Button”,”0”);

El cero indica la posición en la que está en el HTML, para el caso del botón “Masculino” quedaría de la siguiente manera:

IWebElement BotonMasculino =EncontrarElemento(“TagName”,”Button”,”1”);

**MÉTODO Accion():** Cuando se navega por una página web se realizan acciones como, dar click, deslizar elementos, hacer scroll, etc. Estas acciones se realizan sobre elementos del HTML.

**Click en botón o elemento:**

Cuando se quiere realizar una acción sobre un elemento se usa el método acción y se envían los parámetros de la siguiente manera:

**Ejemplo:**

* Si se busca por un selector, Id, Name, ClassName,XPath:

Accion(“Atributo”,”NombreAtributo”,”Acción”,null);

Por ejemplo:

Accion(“Name”,”Boton\_Inicio”,”Click”,Null);

* Si se busca por el TagName

Accion(“TagName”,”NombreTagName”,”Acción”,”Posición”);

Por ejemplo:

Accion(“TagName”,”Button”,”Click”,”0”);

**Tab en elemento:**

Muchas veces es necesario simular el Tab del teclado para seleccionar un valor de una lista observable.

**Ejemplo:**

* Se debe encontrar el atributo y el nombre del atributo del elemento al que se desea realizar Tab.

Accion(“Atributo”,”NombreAtributo”,”Tab”,null);

Por ejemplo:

Accion(“Name”,”Numero\_Identificacion”,”Tab”,null);

**Scroll fin de página:**

Para realizar scroll al final de la página se realiza de la siguiente manera.

**Ejemplo:**

* Se declara el elemento, el cual debe ser el Scroll de la página.

Accion(“Atributo”,”NombreAtributo”,”Acción”,null);

Por ejemplo:

Accion(“Name”,”Scroll\_Pagina”,”SCROLL-FIN”,null);

**Scroll elemento:**

Para realizar el scroll a la parte superior, se debe encontrar el Id del elemento y se usa de la siguiente manera.

**Ejemplo:**

* Una vez se conozca el Id del elemento se envía al método Accion():

Accion(null,”Id”,”SCROLL-ELEMENT”,Null);

**Scroll Posición de página:**

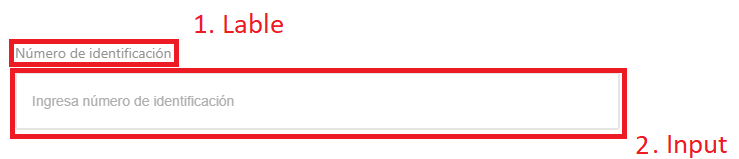
Para realizar el scroll en algún punto específico de la página se hace de la siguiente manera:

**Ejemplo:**

* Una vez se conozca el Id del elemento se envía al método Accion():

Accion(null,null,”SCROLL”,”Posicion”);

**MÉTODO ValorElemento ():** Este método se encarga de encontrar el texto dentro de un label o un input del HTML y se usa de la siguiente manera:



**Figura 13. Label y Input**

Este método se usa cuando hay integraciones entre páginas de diferente navegador, y el dato de salida de la primera es dato de entrada de la segunda página, cuando sucede esto, es posible que el dato de entrada aparezca una vez se inicie la segunda página, entonces se debe capturar ese valor y compararse contra el que se encuentra en el archivo de origen de datos.

ValorElemento(“IdLable”,”ValorIdLable”,”IdInput”,”ValorIdInput”);

Ejemplo:

ValorElemento(“Name”,”NumeroIdentificacion”,”Name”,”ING\_CLI\_NUM\_ID\_CLIE”);

Este método retorna el texto que aparece en el label y el valor insertado en el input o el que apareza, en el caso de la figura 13 retornaría lo siguiente:

Label: Número de Identificación (Debe existir una columna en el archivo origen)

Input: 1096216579 (Este valor debe estar en el archivo Origen para que sea comparado y se ejecute ese caso de prueba de la segunda aplicación).

**EMPEZAR UN PROYECTO DE AUTOMATIZACIÓN DESDE CERO UTILIZANDO LA APLICACIÓN DE CONSOLA**

**Para ejecutar las aplicaciones en el navegador de internet Explorer se debe configurar el zoom en 100%.**

Se debe diligenciar el archivo de configuración (csv) el cual se encarga de especificar el orden que debe llevar la aplicación al momento de ejecutar las pruebas automatizadas, a continuación se muestra la estructura del archivo:



**Dónde:**

**RutaArchivo:** Corresponde a la ruta donde está el archivo que contiene la información en los formularios.

**Url:** Dirección de la página web.

**App:** Nombre de la aplicación donde se quiere dejar la evidencia.

**Query:** Si la página necesita como entrada información de la base de datos, acá se deja la consulta encargada de realizar la obtención de esos datos y debe tener la siguiente estructura:

**Ejemplo:**

SELECT ad.id\_admision,ad.estado FROM ADMISION.dbo.admision ad, ADMISION.dbo.admision\_detalle da WHERE ad.id\_admision=da.id\_admision and da.num\_id\_per = '{0}' , da.estado ={1}

Se debe tener cuidado en las condiciones del where, si el dato es un varchar se coloca entre comillas simples ‘{0}’, de lo contrario se deja sin comillas.

**Parámetros:** Acá se debe poner el nombre de los campos consultados separados por guion(-), para el caso del ejemplo anterior se debe poner de la siguiente manera:

id\_admision-estado

**Condiciones:** Se deben poner los campos del archivo origen de datos que deben ir en el where, para el caso del ejemplo anterior, se realiza la consulta por el campo id de persona y estado. Entonces en el archivo de origen de datos deben los dos campos a sustituir en la consulta.

NoId-Estado -> Campos del origen de datos con valor

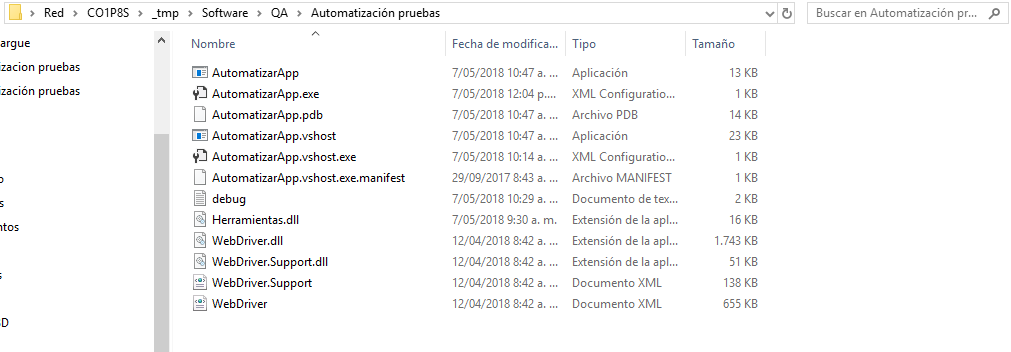
**RutaProyecto:** Ruta donde se encuentra el txt con toda la lógica de la aplicación.

**Navegador:** Se indica con cual navegador se debe abrir la aplicación (Chrome-C, InternetExplore –I, Mozilla-M).

**DatoEntrada:** Se especifica si esa página requiere datos de entrada (true, false)

Se debe tener el ejecutable del proyecto de automatización, el cual estará ubicado en la siguiente ruta: \\CO1P8S\\_tmp\Software\QA\Automatización pruebas.

Allí se encuentran los archivos que componen a todo el proyecto de automatización para una página web en general, ver figura 14.



**Figura 14.Archivos de proyecto de automatización**

Una vez ubicados en la carpeta, se debe tener en cuenta el archivo de configuración (ver figura 16), en este se guardan los siguientes parámetros:



**Figura 15.Archivo de configuración.**

**RutaLog:** Corresponde a la ruta donde se quiere dejar guardado el log y el archivo con el código programado de la aplicación web.

**RutaOrigen:** Corresponde a la ruta donde se debe dejar el archivo de origen de datos.

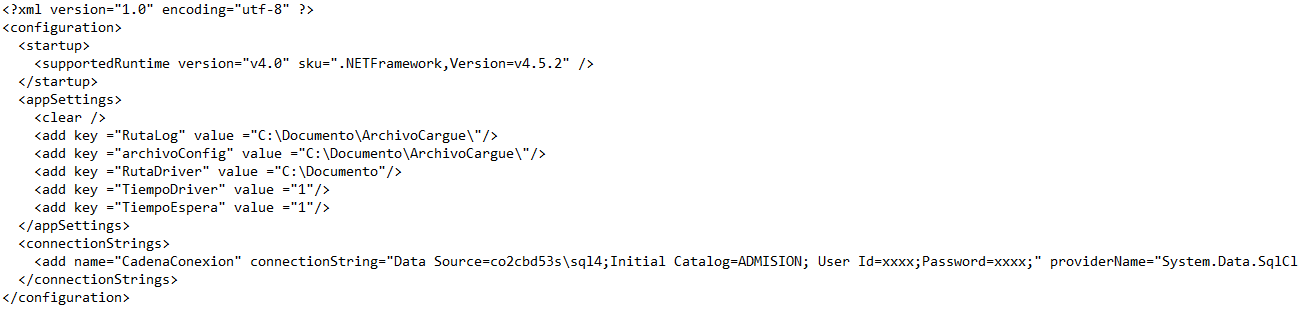
**RutaDriver**: Corresponde a la ruta donde se debe dejar el driver.

**Url:** Corresponde a la dirección url de la página que se está automatizando.

**TiempoDriver** (Minutos): Corresponde al tiempo de espera que tiene el driver para iniciar, para maquinas que presenten lentitud es conveniente manejar tiempos mayores a uno.

**TiempoEspera:** Es el multiplicador usado para los retardos que se configuren en la página.

El archivo de configuración tiene la estructura de la figura 16, donde Key representa el nombre del parámetro y value su valor, se recomienda dejar los valores como aparecen en la imagen y crear las rutas de las carpetas de las variables.



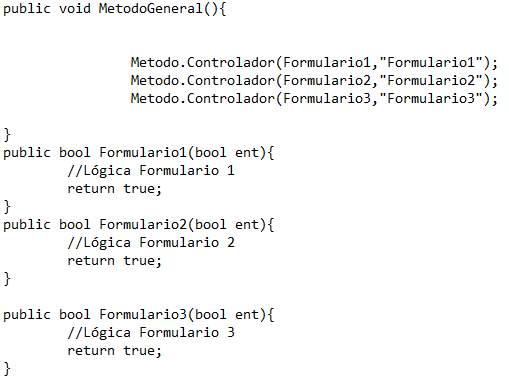
**Figura 16.Estructura archivo de configuración.**

**RutaLog:** Ruta general donde se van a dejar las evidencias de la aplicación.

**archivoConfig:** Ruta donde se debe dejar el archivo de configuración.

**RutaDriver:** Ruta donde estará el driver del navegador.

El archivo que contiene la lógica de la aplicación web debe tener la siguiente estructura:



**Figura 17.Estructura archivo lógica de aplicación web.**

Cada formulario de la figura 17 corresponde a una pantalla de la aplicación, la lógica que se implementa aquí corresponde a la explicada al inicio del documento, se debe tener en cuenta la siguiente sintaxis para el documento de la figura 17.

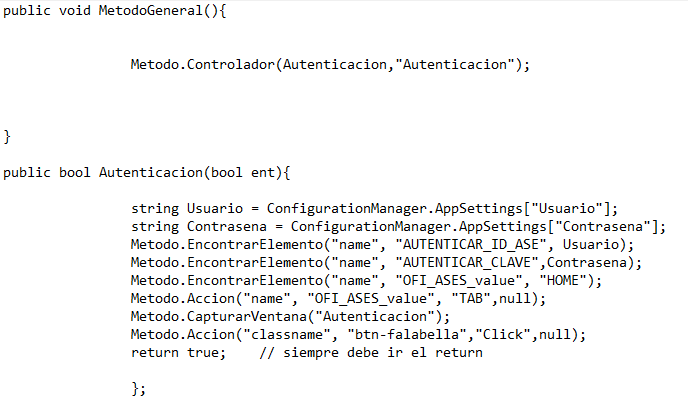
* Para el método encontrar elemento, se debe llamar así

Metodo.EncontrarElemento(“Atributo”,”ValorAtributo”,”Dato”);

* Para el método Accion, se debe llama así:

Metodo. Accion(“Atributo”,”NombreAtributo”,”Acción”,”Posicion ó null”);

La figura 18 muestra un ejemplo de cómo debería quedar el archivo de lógica.



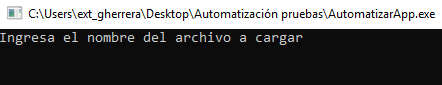
**Figura 18. Ejemplo de guía**

Una vez se tenga el archivo con la lógica implementada, se procede a ejecutar el programa dando click en el siguiente icono que se encuentra en la carpeta de la figura 14.



**Figura 19. Ejecutable de la aplicación**

Al abrir, aparece una consola como la de la figura 20, en donde se coloca el nombre del archivo con la lógica de la aplicación web y se da enter.



**Figura 20. Consola**