|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\jpataco\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C3718269.tmp  Resultado de imagen de mi casa ya | | *Proyecto:* ***Mi Casa Ya***  **Estrategia Certificación pruebas Automatizadas**  **Manual de Desarollador** | | | | Resultado de imagen de gml software | |
| Resumen | | | | | | | |
| El presente documento recopila información técnica del proyecto de Automatización proyecto MiCasaYa | | | | | | | |
| Registro de modificaciones | | | | | | | |
| Versión | Descripción [o descripción de cambios] | | Autor | Fecha creación | Aprobado por | | Fecha aprobación |
| 1.0 | Proyecto Automatización Mi Casa Ya | | Jairo Patacon | 17/02/2021 | Olga Martinez | | 25/02/2021 |
| 1.1 | Ajustes Revisión ScrumMaster | | Jairo Patacon | 26/02/2021 | Olga Martinez | | 27/20/2021 |
|  |  | |  |  |  | |  |

1. **Propósito**

El presente documento describe la Arquitectura y el funcionamiento de las pruebas automatizadas para el Proyecto MiCasaYa.

1. **Objetivos**

Determinar la arquitectura de Automatización, los parámetros a configurar, las entradas, el proceso automatizado y las evidencias del proceso automatizado (Capturas de pantalla y Logs).

1. **Alcance**

Identificar todos los puntos que se deben tener en cuenta para la ejecución y mantenimiento de los scripts para la automatización de pruebas en MiCasaYa.

1. **Modelo de implementación**
   1. **Versionamiento de los códigos de pruebas automatizadas**

La automatización de pruebas es como un proyecto Maven que debe conservar el código implementado en herramientas de versionamiento. Se utilizó GitLab, Se tienen dos ramas permanentes que son la master y la develop, la master se sincronizara cuando el release de la aplicación sea generado.

* 1. **Repositorio**

La URL del repositorio del proyecto de automatización es:

[**https://git.universo.corp:8082/MiCasaYa/MCY-Automation**](https://git.universo.corp:8082/MiCasaYa/MCY-Automation)

1. **Actividades de QA antes de inicio de sprint -n**

• Identificar los escenarios de ruta crítica

• Velar y asegurar el ambiente de pruebas

• Crear el DECK de pruebas base

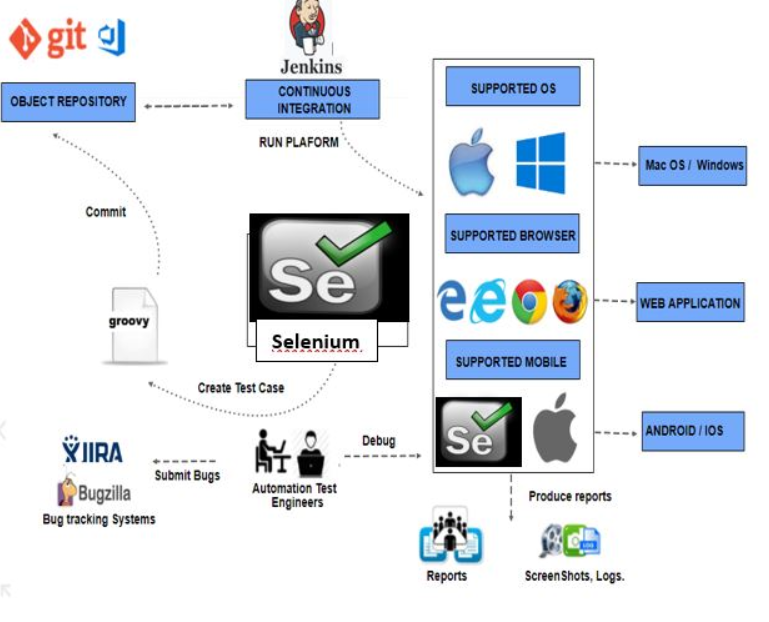
• Parametrizar, crear, gestionar los datos y condiciones del mismo para realizar las pruebas (usuario, contraseña, números de cedula…entre otros) para la ruta crítica y casos alternos, de tenerlos.

• Verificar Ambientes (Dev-QA-UAT), usuario de ingreso a ambiente y datos válidos para realizar la prueba.

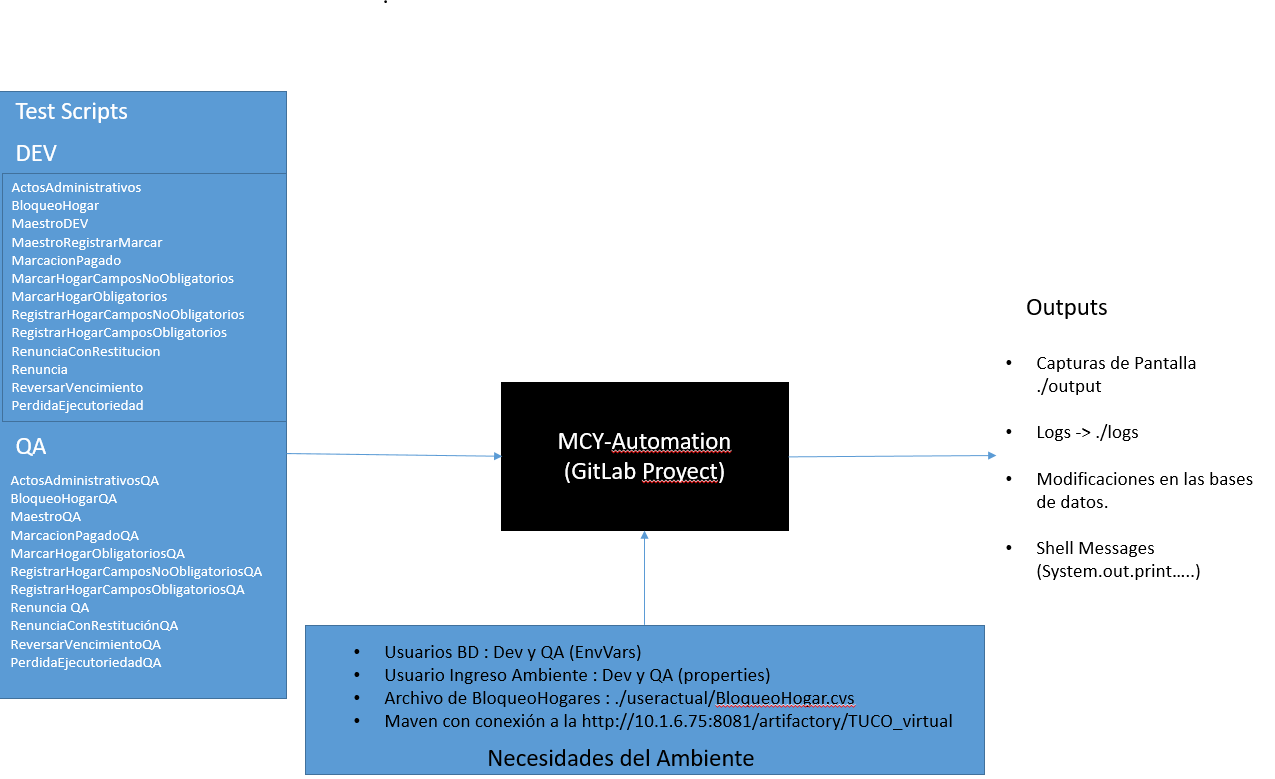
• Validar datos de entrada que no pueden ser obtenidos por base de datos para establecerlos como parámetros.

1. **Arquitectura:**

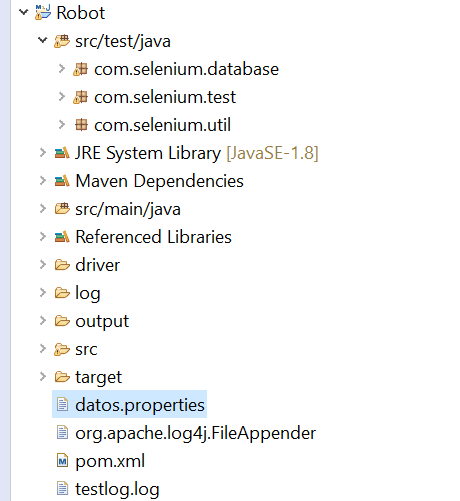
**6.1- Integración continua DEVOPS:**



**6.2- Caja Negra Proyecto Automatización:**



**6.3 – Estructura del Proyecto de automatización:**



El Proyecto de Automatización es un proyecto Maven que conecta con base de datos Oracle y tiene configurado un archivo parámetros donde se encuentran rutas y los datos de selección manual (Ejemplo datos de listas desplegables).

Como salida del proceso de automatización se tienen las imágenes de evidencia de las pruebas ejecutadas, las cuales se encuentran en la carpeta Output y el archivo testlog.log donde se encuentran los logs con información de las ejecuciones de las pruebas automatizadas.

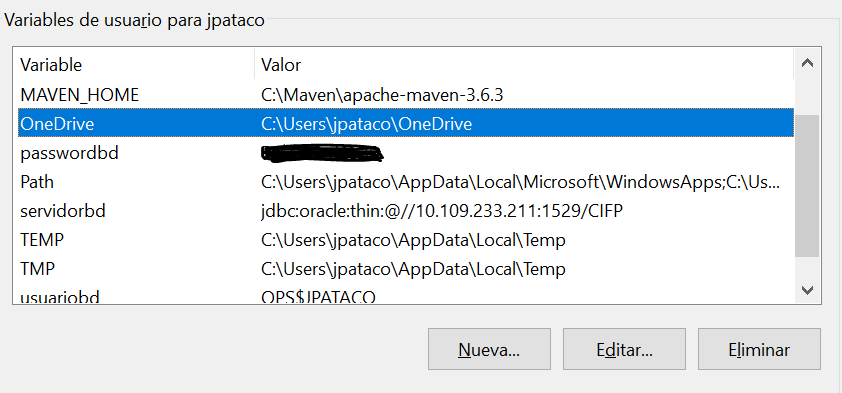
**6.3.1- Conexión con la Base de Datos:**

Dentro del paquete com.selenium.database se encuentra la Clase Conectar:



Por seguridad las credenciales de Conexión se encuentran en las variables de entorno, pero también para cambios de ambiente se pueden configurar en el archivo de parámetros, en el script de conexión se encuentran ambas opciones para habilitar la que se vaya a utilizar.

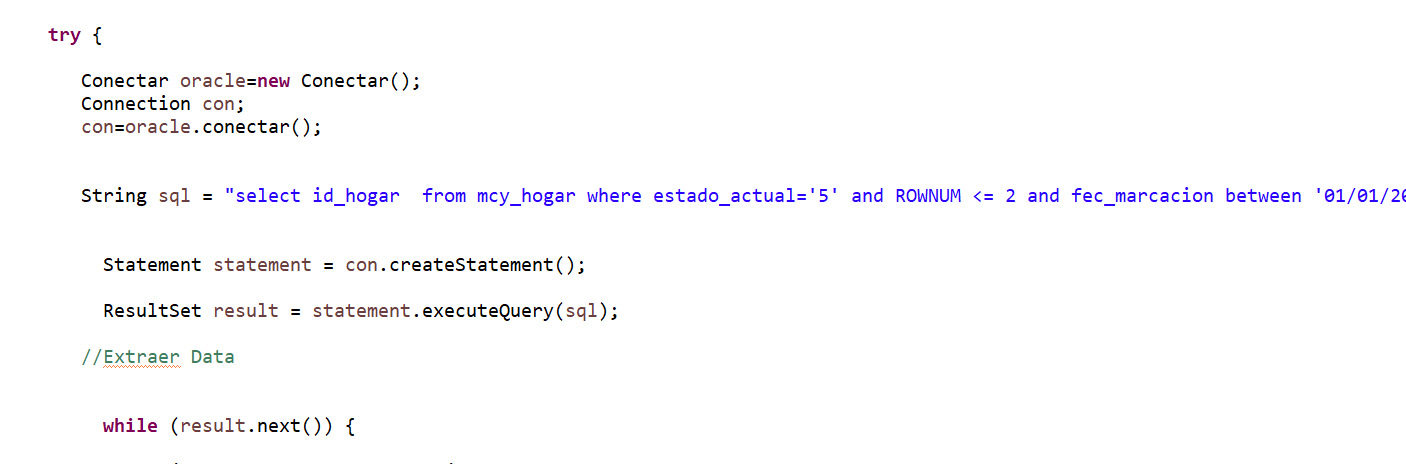
***Variables de entorno:***



Se recomienda configurar las credenciales de acceso a la bd como variables del entorno del servidor.

**6.3.2- Querys Base de Datos:**

Dentro de las clases se encuentra el Query para extraer los datos requeridos para la prueba:



**6.3.3- Parámetros:**

El archivo datos.properties tiene la configuración de ambientes, parámetros de entrada para las pruebas a ejecutar y ruta donde están las evidencias (Imágenes) y datos que serían ingresados por el usuario en la interacción con el modulo a Testear:

\*\*IMPORTANTE:

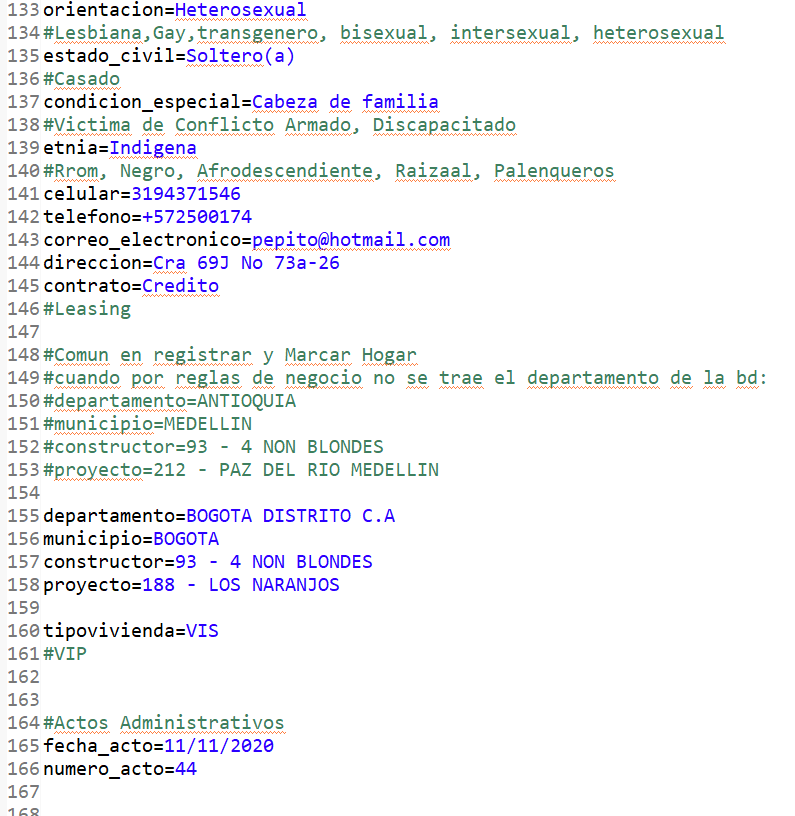
#Credenciales para configuración en ambiente de QA por variables de entorno.

#Es importante que se cambien los parámetros para cada ambiente

#comentando los parámetros que sean de otros Ambientes

#LA CONFIGURACION DE ESTOS PARAMETROS CRUZADA ENTRE AMBIENTES

#(DEV-QA-UAT) OCASIONA BLOQUEOS DE USUARIOS DE LA BASE DE DATOS



**7 – Necesidades del Ambiente:**

* Usuarios BD: Dev y QA (EnvVars)
* Usuario Ingreso Ambiente: Dev y QA (properties)
* Archivo de BloqueoHogares : ./useractual/BloqueoHogar.cvs
* Maven con conexión a la [**http://10.1.6.75:8081/artifactory/TUCO\_virtual**](http://10.1.6.75:8081/artifactory/TUCO_virtual)

**8- Test Script:**

**DEV:**

* ActosAdministrativos
* BloqueoHogar
* MaestroDEV
* MaestroRegistrarMarcar
* MarcacionPagado
* MarcarHogarCamposNoObligatorios
* MarcarHogarObligatorios
* Registrar
* RegistrarHogar\_MarcarHogar\_Obligatorios
* RegistrarHogarCamposNoObligatorios
* RegistrarHogarCamposObligatorios
* RenunciaConRestitucion
* Renuncia
* ReversarVencimiento
* PerdidaEjecutoriedad

**QA**

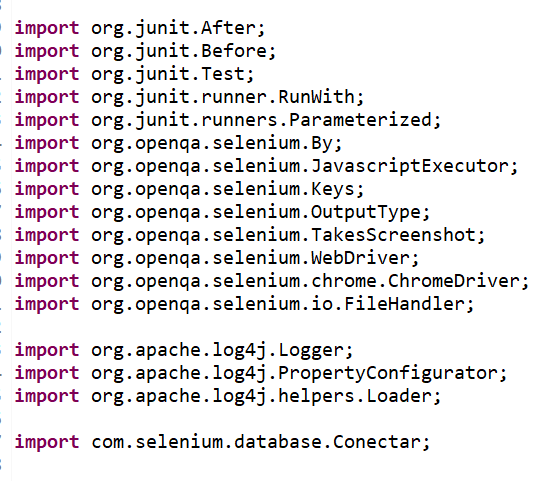
* ActosAdministrativosQA
* BloqueoHogarQA
* MaestroQA
* MarcacionPagadoQA
* MarcarHogarObligatoriosQA
* RegistrarHogarCamposNoObligatoriosQA
* RegistrarHogarCamposObligatoriosQA
* RenunciaConRestitucionQA
* RenunciaQA
* ReversarVencimientoQA
* PerdidaEjecutoriedadQA

**8.1. Estructura de Script de Automatización:**

El proyecto está desarrollado en JUnit y Selenium.

JUnit se trata de un Framework Open Source para la automatización de las pruebas (tanto unitarias, como de integración) en los proyectos Software. El framework provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en su sistema y así asegurar su consistencia y funcionalidad

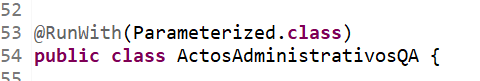
Selenium es un recurso que se utiliza para automatizar pruebas de sistemas. Esta herramienta permite al usuario reproducir el ambiente real de la aplicación. A  a través de una extensión se pueden crear los scripts de prueba.



Llamado al paquete y clase donde se encuentra la conexión con la base de datos

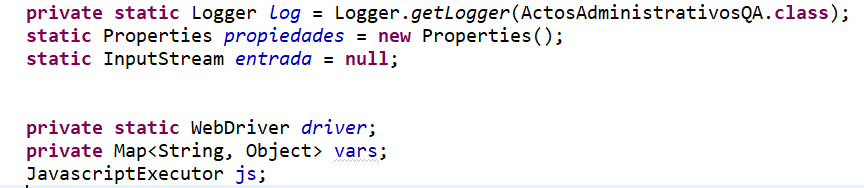


Se realizó con la anotación Junit:

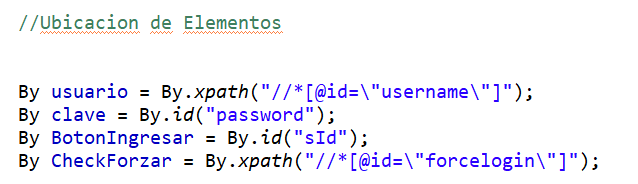


La funcionalidad de esta anotación es al conectarse con Bases de Datos y realizar una iteración de pruebas por cada uno de los datos que se obtienen de la Base de datos. Por ejemplo si se traen 3 números de Cedula, realizara 3 iteraciones, (ejecutara la anotación @ test )

Se crean variables donde se encuentran llamado a librería de logs, llamada a el archivo properties, librería de entrada de datos por teclado de Java y llamado al WebDriver de Selenium

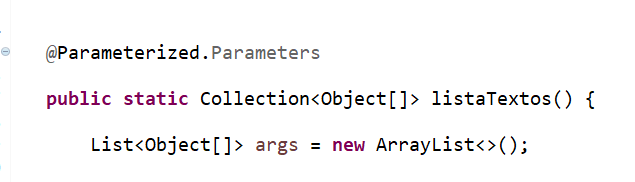


Luego se encuentra la ubicación de elementos de la Pagina Web a automatizar utilizando los identificadores de Selenium (Xpath, Ids…etc.) :

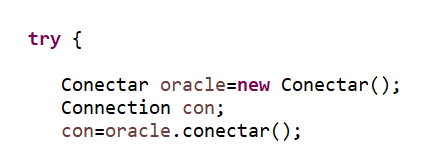


La anotación Junit @ Parameterized.parameters establece los parámetros del Test.

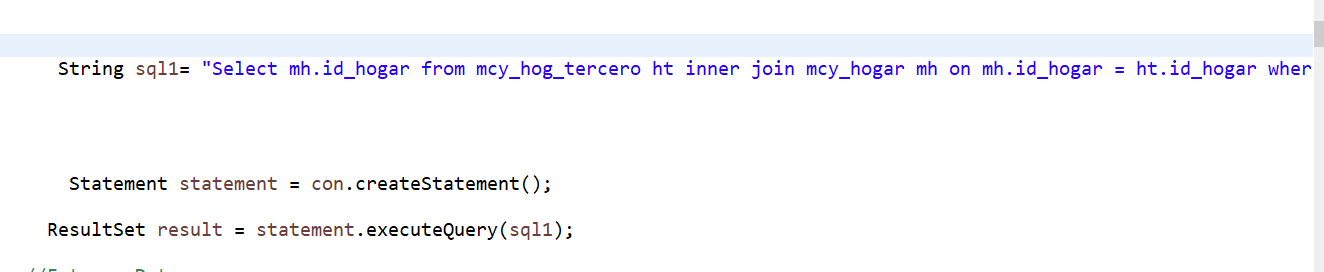
Se crea un arreglo donde almacenará los datos extraídos de la Base de Datos para la prueba que se realizará.



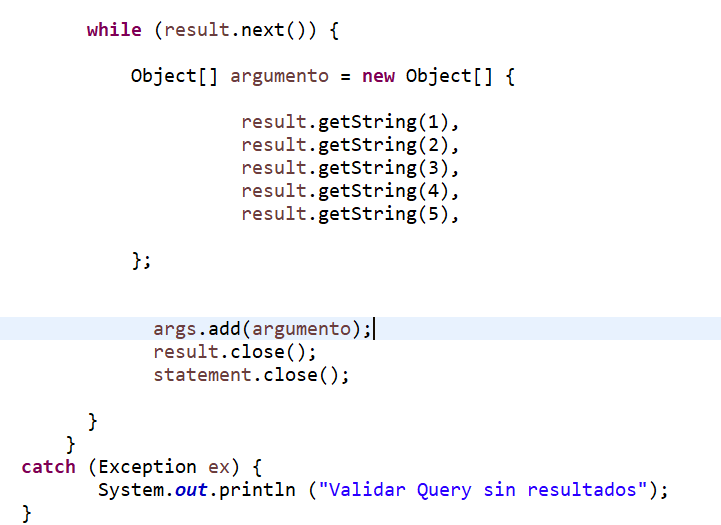
Se crea el constructor que llama la clase Conectar donde se encuentra la conexión con la base de datos.



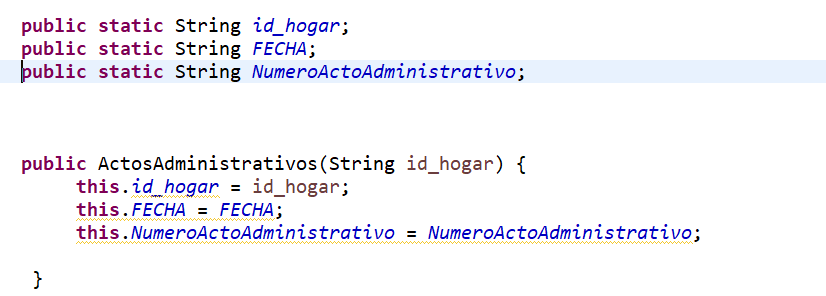
En la Variable String sql se ingresa la consulta que se realizara a la base de datos para obtener los datos requeridos para realizar el test y utilizando la función Statement realizara la ejecución del Query con la base de datos



Luego los datos extraídos de la BD se encuentran en la Variable result y se almacenan en el Objeto argumento cada result.getString(#) ,el # corresponde a una columna de la base de datos, si se extrae un dato por ejemplo cedula se guarda con result.getString (1), pero si adicional se guardara nombres(2), apellidos(3), dirección(4) y teléfono(5); se utilizara hasta result.getString(5)

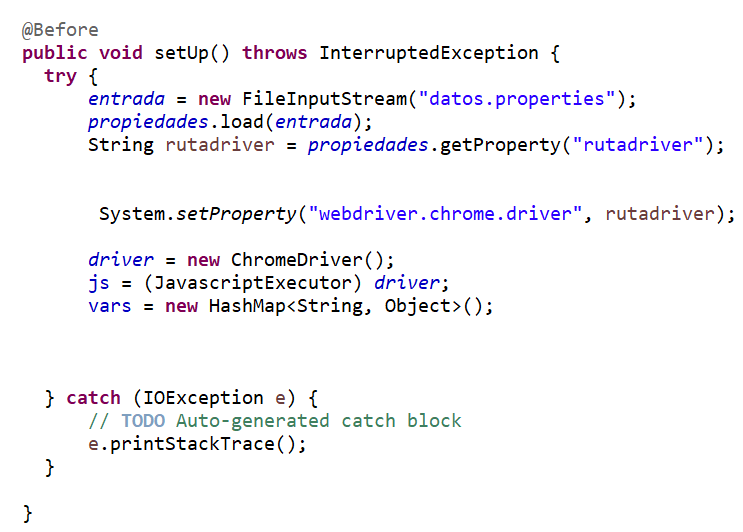


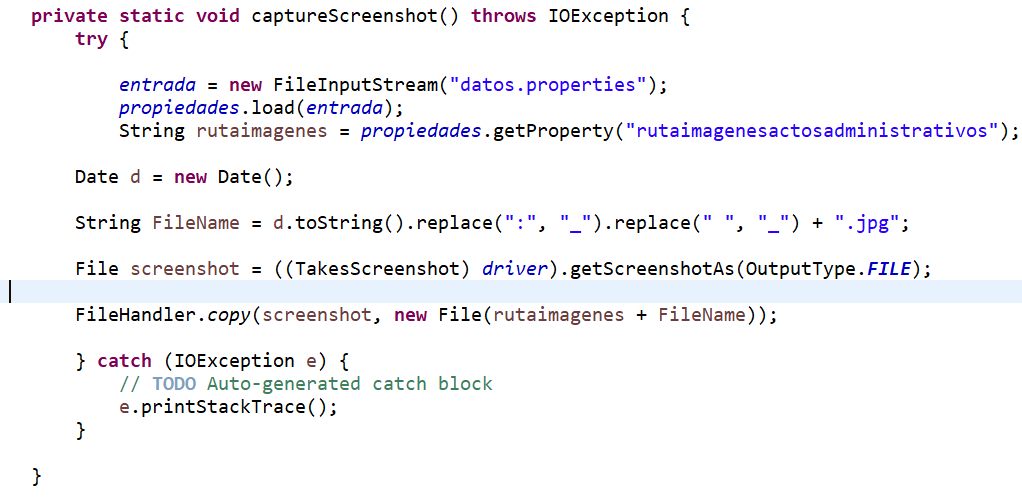
Luego crea las variables donde guardara los datos de cada result.getstring del arreglo y un método con los constructores para estas variables.



La anotación @Before están las funciones requeridas Antes de iniciar el Test, la función setUp carga el driver de Selenium para que pueda utilizar todos los métodos de Selenium durante la ejecución de test.

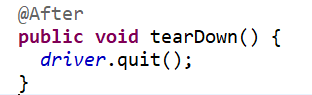
La función CaptureScreenshot se utiliza para realizar las capturas de pantalla y determinar la ruta que se encuentra en el archivo ***datos.properties*** apuntando a la carpeta output donde almacenara las capturas de pantalla.





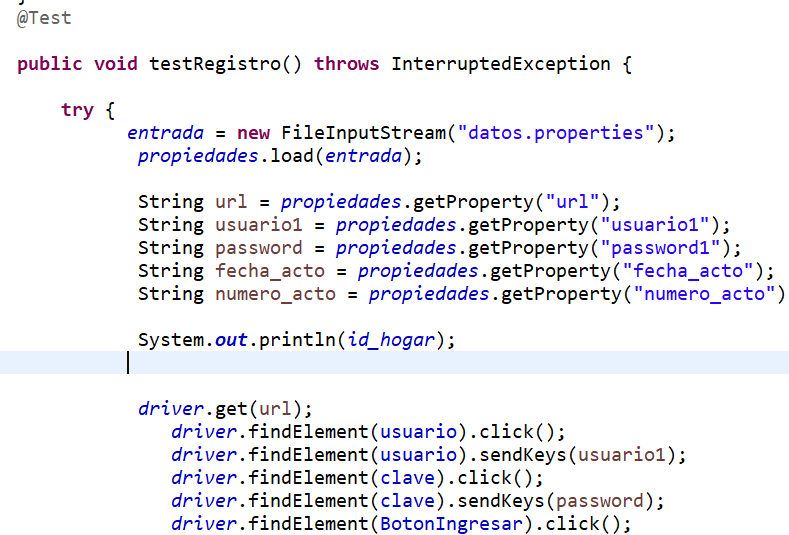
La anotación @ After es lo que realizara después de ejecutado el Test, driver.quit( ); cerrara la ventana

De Chrome al finalizar cada iteración



La anotación @Test contiene la ejecución de la prueba automatizada en primer lugar está el llamado al archivo properties donde se encuentran los parámetros requeridos para la realización del Test (ejemplo: credenciales de logueo) así como datos que no se encuentran en la base de datos que serán ingresados por el usuario.

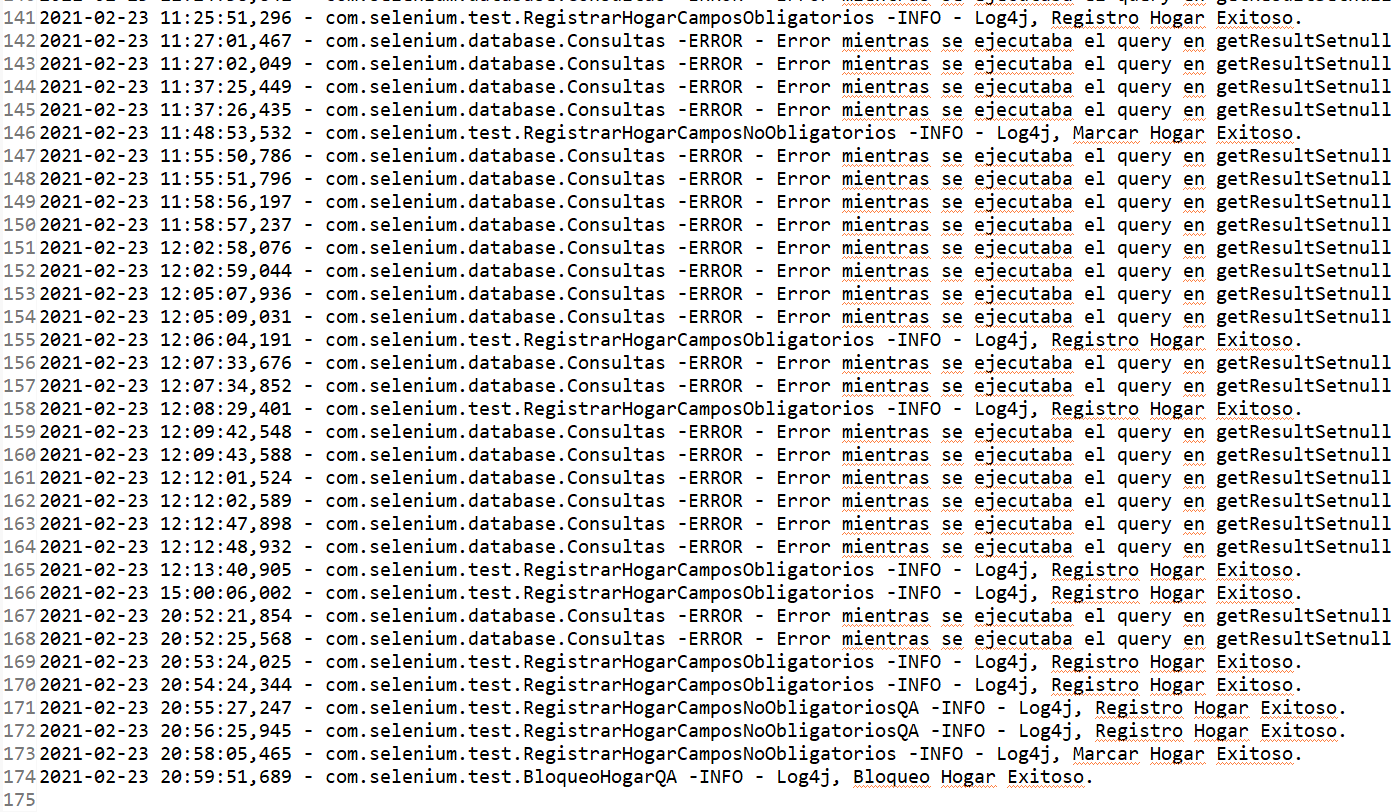
Luego llama uno a uno los elementos identificados en la página web y las acciones (métodos) que se ejecutaran en ellos (Click, escribir texto).



**9-OUTPUTS:**

**9.1-ARCHIVO TESTLOG.LOG:**

Esta la información acerca de cada una de las ejecuciones y los errores generados durante las ejecuciones.



**9.2-CAPTURAS DE PANTALLA:**

En la carpeta Output se encuentran las imágenes finales de cada uno de los Test, esta evidencia es importante para anexar a la documentación de Certificación de QA



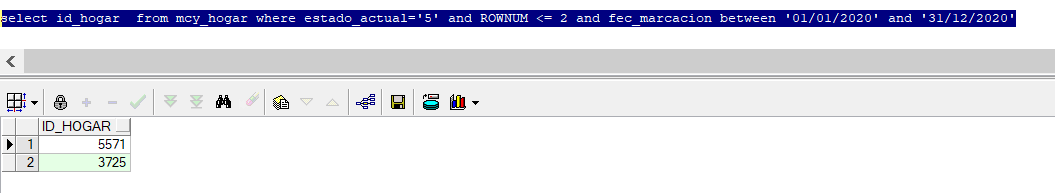
**9.3-MODIFICACION DE REGISTROS EN BASE DE DATOS:**

Como resultado de la ejecución automatizada se producen cambios de los registros de entrada en las bases de datos de acuerdo al test ejecutado.

Por ejemplo, luego de realizar el proceso de Registrar Hogares con un numero de cedula resultado de un Query ejecutado de manera automatizada la base de datos se genera un Id\_Hogar y el registro queda en estado por Marcar.

Cuando se ejecuta la automatización de marcación, el registro del test cambian a estado Marcado.

De acuerdo al Test ejecutado se puede reversar, anular o bloquear una solicitud.



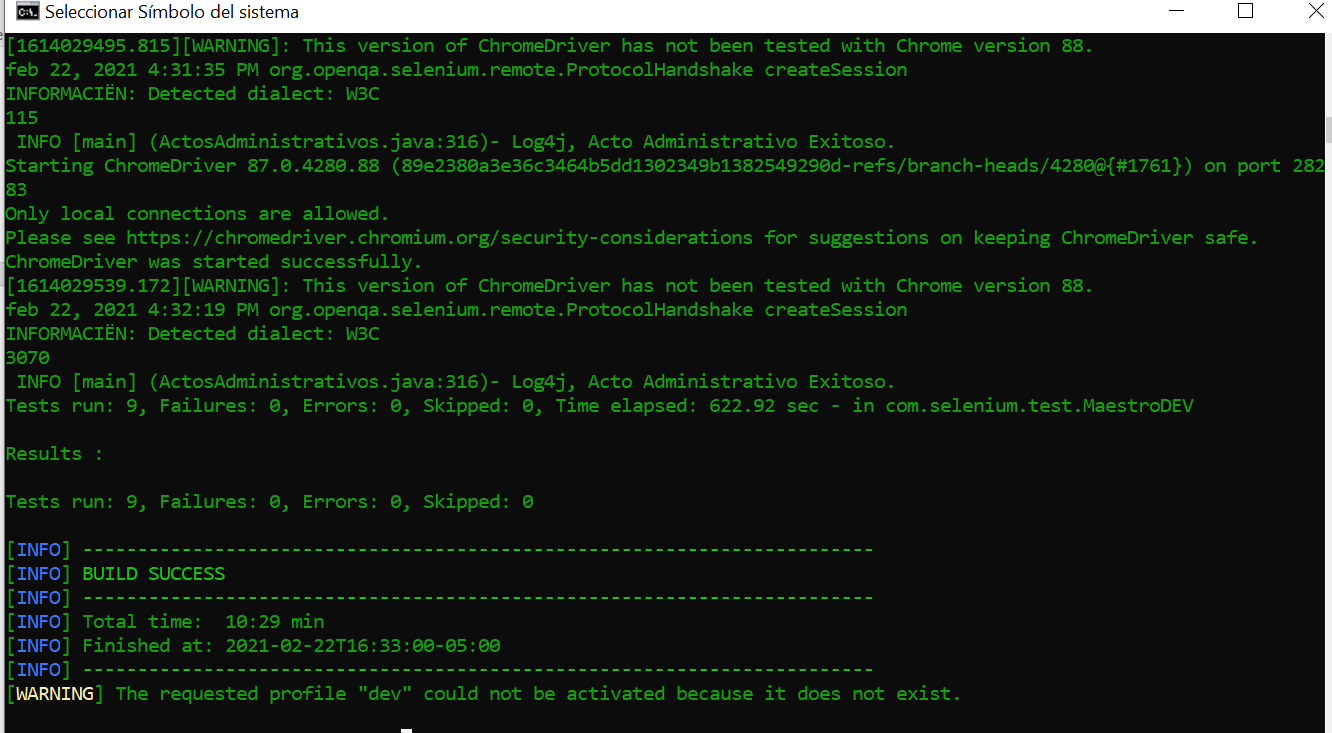
**9.4-** **SHELL- MENSAJES:**

La Automatización se ejecuta desde línea de comandos Maven:

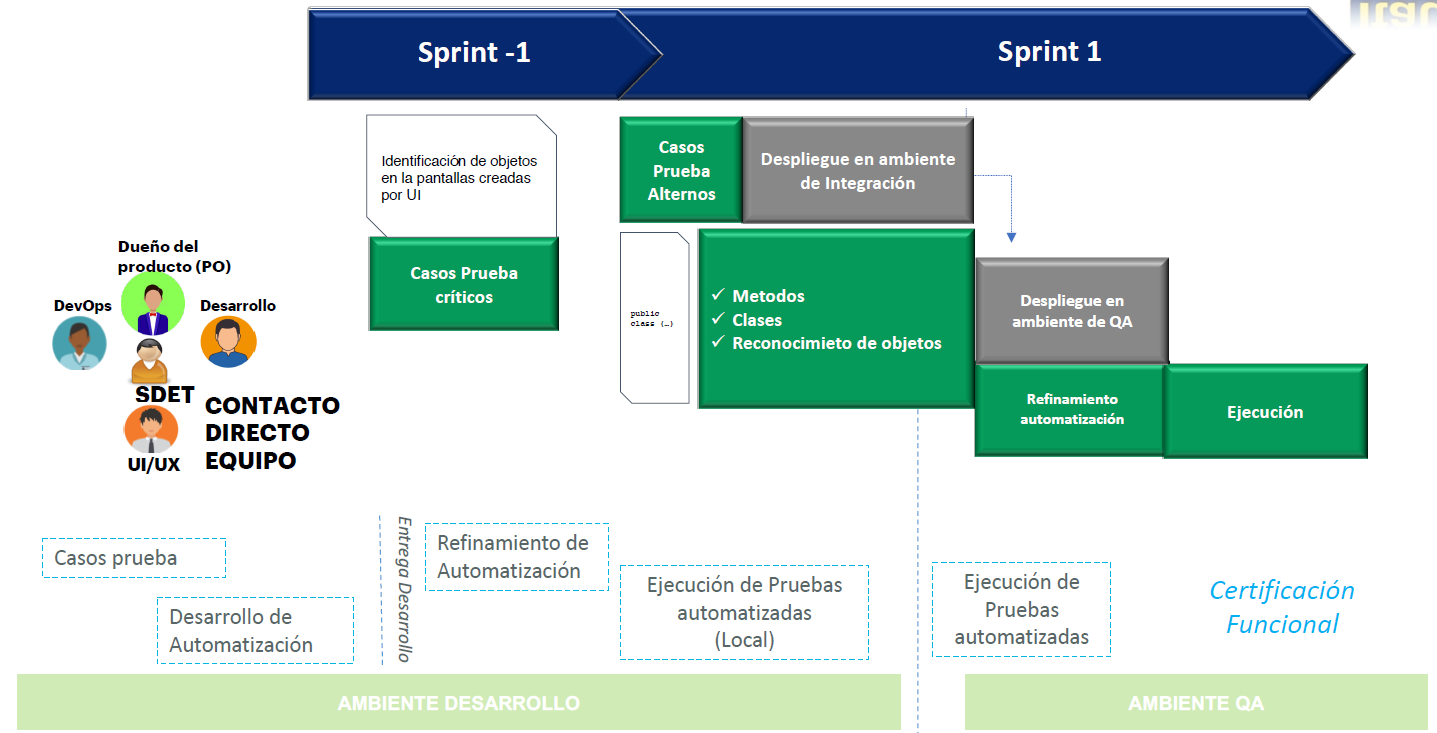
**Comando:**

**mvn test -Dtest=Nombre\_Test**(mvn test -Dtest=RegistrarHogarCamposObligatorios)

al ejecutarse se generan mensajes que informan el avance o los errores encontrados en el Test realizado.



**10: ESTRATEGIA DE TRABAJO Y DISTRIBUCION:**



En la figura Se presenta la estrategia para el desarrollo de los Scripts Automatizados.

**10.1- DOR -Definition of ready QA:**

Participación activa desde el refinamiento de las historias por parte del equipo de certificación.

En los inicios de cada sprint se levantan la casuística concerniente a las funciones que se implementan en el sprint.

**UI/UX**

Pantallas de visualización creadas y socializadas

**PO**

HU con los criterios de aceptación bien definidos y socializados con Dev y QA

**DEV**

Realización de pruebas unitarias, de servicios cumpliendo con un 70 % de set de pruebas entregado por QA y asignar los ID’s requeridos para la automatización.

**SM/TL**

Velar por ambientes de pruebas estables sin limitantes que puedan provocar stoppers.