

Proyecto Fin de Carrera
Grado en Ingeniería de las Tecnologías de
Telecomunicación

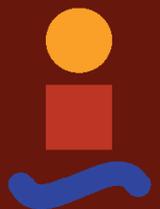
Automatización de pruebas de regresión

Autor: Eduardo Díaz Asencio

Tutor: Juan Manuel Vozmediano Torres

Dpto. de Ingeniería telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021



Proyecto Fin de Carrera
Grado en Ingeniería de las Tecnologías de Telecomunicación

Automatización de pruebas de regresión

Autor:

Eduardo Díaz Asencio

Tutor:

Juan Manuel Vozmediano Torres

Profesor titular

Dpto. de Ingeniería telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Proyecto Fin de Carrera: Automatización de pruebas de regresión

Autor: Eduardo Díaz Asencio

Tutor: Juan Manuel Vozmediano Torres

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis maestros

AGRADECIMIENTOS

A mis padres, mi hermana y toda mi familia por todo el calor recibido. A Marta, por su apoyo incondicional e infinita paciencia. A mis compañeros y profesores por brindarme más que una magnífica etapa universitaria. A todos ellos, gracias.

RESUMEN

Históricamente, la automatización surgió para reducir el esfuerzo humano requerido en actividades que podrían ser replicadas por un sistema o máquina programable. Al automatizar pruebas de software se persigue el objetivo de simplificar el trabajo dispendioso, repetitivo o complejo, haciéndolo efectivo y más productivo. De esta manera, es posible ahorrar energía, tiempo y costos, al tiempo que libera a las personas para que se concentren en otras tareas.

Este trabajo ofrece un entorno de desarrollo autocontenido para la realización de pruebas software de la aplicación web de Sigma4Lifts. Para la creación del entorno completo hemos utilizado la herramienta Vagrant, herramienta de creación y configuración de entornos de desarrollo virtualizados. Con ayuda de VirtualBox, creamos una máquina virtual corriendo en Debian 10 con todas las herramientas instaladas y listas para usar: IDEs, base de datos, aplicación web desplegada, códigos, etc.

Para la creación del proyecto de automatización se ha usado Java como lenguaje de programación, debido a su flexibilidad, robustez y soporte. Para el testeado de la aplicación web utilizaremos Selenium: conjunto de utilidades que facilita la labor de obtener juegos de pruebas para aplicaciones web. Nos permite grabar, editar y depurar casos de prueba que podrán ser ejecutados de forma automática e iterativa posteriormente. Además de ser una herramienta para registrar acciones, permite editarlas manualmente o crearlas desde cero.

Con este proyecto se ha conseguido obtener una batería de pruebas automatizadas que podrán ser utilizadas cuando sea necesario con un informe detallado de la ejecución. Todo ello en un entorno de desarrollo portable y compatible con los principales sistemas operativos del mercado.

ABSTRACT

Historically, automation arose to reduce the human effort required in activities that could be replicated by a programmable system or machine. By automating software testing, the objective is to simplify time-consuming, repetitive or complex work, making it effective and more productive. In this way, it is possible to save energy, time and costs, while freeing people to concentrate on other tasks.

This work provides a self-contained development environment for testing Sigma4Lifts web application software. For the creation of the complete environment we have used Vagrant, a tool for creating and configuring virtualized development environments. With VirtualBox's help, we created a virtual machine running con Debian 10 with all needed tools already installed and ready to use: IDEs, database, deployed web application, codes, etc.

For automation project's creation, Java has been used as the programming language, due to its flexibility, robustness and support. For the web application testing we used Selenium: a set of utilities that facilitates the effort of obtaining test kits for web applications. It allow us to record, edit and debug test cases that can be executed automatically and iteratively later. In addition to being a tool to record actions, it allows you to edit them manually or create them from scratch.

With this project, it has been possible to obtain a battery of automated tests that can be used whenever necessary with a detailed report of the execution. All this in a portable development environment compatible with the main operating systems on the market.

Agradecimientos	ix
Resumen	xi
Abstract	xiii
Índice	xv
Índice de figuras	xviii
Índice de códigos	xxii
Notación	xxiv
1. Motivación y objetivos	1
1.1. <i>Motivación</i>	1
1.2. <i>Objetivos</i>	1
2. Pruebas de software	3
2.1. <i>Concepto de calidad</i>	3
2.2. <i>Pruebas de software</i>	3
2.3. <i>Tipos de pruebas</i>	4
2.3.1. Tipos de pruebas por su ejecución	5
2.3.2. Tipos de prueba por su enfoque	5
2.3.3. Tipos de prueba según lo que verifican	5
3. Dependencias y librerías	7
3.1. <i>Herramientas de automatización de pruebas para interfaces de usuario web</i>	7
3.1.1. Selenium	7
3.2. <i>Herramientas para la gestión y construcción de proyectos software</i>	8
3.2.1. Maven	8
3.3. <i>Framework para pruebas</i>	9
3.1.1. TestNG	9
3.4. <i>Herramientas de reportes para Selenium</i>	9
3.4.1. Relevantcodes	11
3.5. <i>Herramientas para generación de trazas</i>	12
3.5.1. Log4j	12
4. Proyecto de automatización	11
4.1. <i>Patrones de diseño</i>	11
4.1.1. Patrón Page Object	11
4.1.2. Patrón Test-Actions-Page	13
4.1.3. Patrón Page Factory	17
4.2. <i>Organización del Proyecto</i>	18
4.2.1. Paquetes	18
4.2.2. Recursos	19
4.2.3. Configuración	20
4.3. <i>Ejecución de pruebas</i>	21
4.4. <i>Reporte</i>	22
5. Automatización de las pruebas	27
5.1. <i>Suite de búsqueda</i>	27
5.1.1. Flujo de navegación del usuario	27
5.1.2. Flujo de navegación de Selenium	31
5.2. <i>Suite de persistencia de datos</i>	33
5.2.1. Flujo de navegación del usuario	33

5.2.2.	Flujo de navegación de Selenium	44
5.3.	<i>Suite de casos de uso</i>	45
5.3.1.	Flujo de navegación del usuario	45
5.3.2.	Flujo de navegación de Selenium	56
6.	Conclusiones	58
6.1.	<i>Conclusiones del proyecto</i>	58
6.2.	<i>Líneas de trabajo futuras</i>	58
Anexo A:	Entorno de pruebas	59
1.	<i>Instalar VirtualBox</i>	59
2.	<i>Instalar Vagrant</i>	59
3.	<i>Desplegar el entorno de pruebas</i>	60
3.1.	<i>Instalación de herramientas básicas</i>	62
3.2.	<i>Instalación del IDE y códigos</i>	64
3.3.	<i>Despliegue de Sigma4Lifts</i>	65
3.3.1.	Creación de la base de datos	65
3.3.2.	Configuración de Sigma4Lifts	65
3.3.3.	Instalación de WildFly	66
3.3.4.	Despliegue del conector de la base de datos	66
3.3.5.	Despliegue de la Sigma4Lifts	67
3.4.	<i>Arrancar el entorno</i>	67
Referencias		75

ÍNDICE DE FIGURAS

Figura 2.1. Tipos de pruebas de software.	4
Figura 3.1. Arquitectura Selenium WebDriver.	7
Figura 3.2. Arquitectura Maven.	8
Figura 3.3. Ejemplo reporte TestNG.	10
Figura 3.4. Ejemplo reporte ReportNG.	10
Figura 3.5. Ejemplo reporte ExtentReport.	11
Figura 3.6. Ejemplo reporte ExtentReport personalizado con Relevantcodes.	11
Figura 4.1. Patrón Page Object.	11
Figura 4.2. Página cliente de Sigma4Lifts.	12
Figura 4.3. Representación patrón Test-Actions-Page.	13
Figura 4.4. Página búsqueda clientes Sigma4Lifts.	14
Figura 4.5. Diagrama de clases entre Actions y Pages.	15
Figura 4.6. Diagrama de clases completo de las tres capas del patrón Test-Actions-Page.	17
Figura 4.7. Diagrama de paquetes del proyecto.	18
Figura 4.8. Directorio de recursos de configuración y datos.	20
Figura 4.9. Visión general del reporte.	23
Figura 4.10. Reporte de test concreto sin filtros.	24
Figura 4.11. Reporte de test concreto con filtro de logs 'pass'.	25
Figura 5.1. Primer paso del flujo de navegación de búsqueda.	27
Figura 5.2. Segundo paso del flujo de navegación de búsqueda.	28
Figura 5.3. Tercer paso del flujo de navegación de búsqueda.	28
Figura 5.4. Cuarto paso del flujo de navegación de búsqueda.	29
Figura 5.5. Quinto paso del flujo de navegación de búsqueda.	29
Figura 5.6. Sexto paso del flujo de navegación de búsqueda.	30
Figura 5.7. Diagrama de secuencia del flujo de navegación de búsqueda.	30
Figura 5.8. Diagrama de secuencia para la automatización de búsqueda.	31
Figura 5.9. Primer paso del flujo de navegación de persistencia.	34
Figura 5.10. Segundo paso del flujo de navegación de persistencia.	34
Figura 5.11. Tercer paso del flujo de navegación de persistencia.	35
Figura 5.12. Cuarto paso del flujo de navegación de persistencia.	35
Figura 5.13. Quinto paso del flujo de navegación de persistencia.	36
Figura 5.14. Sexto paso del flujo de navegación de persistencia.	36
Figura 5.15. Séptimo paso del flujo de navegación de persistencia.	37
Figura 5.16. Octavo paso del flujo de navegación de persistencia.	37
Figura 5.17. Noveno paso del flujo de navegación de persistencia.	38

Figura 5.18. Décimo paso del flujo de navegación de persistencia.	38
Figura 5.19. Decimoprimer paso del flujo de navegación de persistencia.	39
Figura 5.20. Decimosegundo paso del flujo de navegación de persistencia.	39
Figura 5.21. Decimotercer paso del flujo de navegación de persistencia.	40
Figura 5.22. Decimocuarto paso del flujo de navegación de persistencia.	40
Figura 5.23. Decimoquinto paso del flujo de navegación de persistencia.	41
Figura 5.24. Decimosexto paso del flujo de navegación de persistencia.	41
Figura 5.25. Decimoséptimo paso del flujo de navegación de persistencia.	42
Figura 5.26. Diagrama de secuencia del flujo de navegación de persistencia.	43
Figura 5.27. Diagrama de secuencia para la automatización de búsqueda.	44
Figura 5.28. Primer paso del flujo de navegación de casos de uso.	46
Figura 5.29. Segundo paso del flujo de navegación de casos de uso.	46
Figura 5.30. Tercer paso del flujo de navegación de casos de uso.	47
Figura 5.31. Cuarto paso del flujo de navegación de casos de uso.	47
Figura 5.32. Quinto paso del flujo de navegación de casos de uso.	48
Figura 5.33. Sexto paso del flujo de navegación de casos de uso.	48
Figura 5.34. Séptimo paso del flujo de navegación de casos de uso.	49
Figura 5.35. Octavo paso del flujo de navegación de casos de uso.	49
Figura 5.36. Noveno paso del flujo de navegación de casos de uso.	50
Figura 5.37. Décimo paso del flujo de navegación de casos de uso.	50
Figura 5.38. Decimoprimer paso del flujo de navegación de casos de uso.	51
Figura 5.39. Decimosegundo paso del flujo de navegación de casos de uso.	51
Figura 5.40. Decimotercer paso del flujo de navegación de casos de uso.	52
Figura 5.41. Decimocuarto paso del flujo de navegación de casos de uso.	52
Figura 5.42. Decimoquinto paso del flujo de navegación de casos de uso.	53
Figura 5.43. Decimosexto paso del flujo de navegación de casos de uso.	53
Figura 5.44. Decimoséptimo paso del flujo de navegación de casos de uso.	54
Figura 5.45. Decimooctavo paso del flujo de navegación de casos de uso.	54
Figura 5.46. Diagrama de secuencia para la automatización de caso de uso.	55
Figura 5.47. Diagrama de secuencia para la automatización de caso de uso.	56
Figura A.1. Descarga VirtualBox.	59
Figura A.2. Descarga Vagrant.	60
Figura A.3. Máquina virtual sin GUI.	62
Figura A.4. Máquina virtual con GUI.	63
Figura A.5. Comando interactivo.	64
Figura A.6. Despliegue correcto de Sigma4Lifts en la máquina virtual.	71
Figura A.7. IntelliJ IDEA instalado correctamente.	71
Figura A.8. Abrir proyecto en IntelliJ IDEA.	72
Figura A.9. Abrir proyecto de automatización sigma-web-test.	72

ÍNDICE DE CÓDIGOS

Código 4.1. PageObject para la página de los clientes UserDataPage.java	12
Código 4.2. Método searchByLogin de ActionsClientes.java	15
Código 4.3. Caso de prueba para la búsqueda de un cliente en SearchClienteByLoginTest.java	15
Código 4.4. BaseTest.java	16
Código 4.5. Método de ActionsClientes.java	17
Código 4.6. Constructor ClientesPage.java	17
Código 4.7. Constructor BasePage.java	18
Código 4.10. Environment.properties	20
Código 4.8. Fichero pom.xml	21
Código 4.9. SearchSuite.xml	21
Código 4.10. DebugSuite.xml	22
Código 4.11. Trazas en método searchByLogin de ActionsClientes.java	26
Código 5.1. SearchClienteByLoginTest.java	31
Código 5.2. ActionsClientes.java	32
Código 5.3. ClientesPage.java	32
Código 5.4. EditClienteDataTest.java	45
Código 5.5. AddAlarmaToLiftTest.java	57
Código A.1. Vagrantfile	61
Código A.2. Instalación de las herramientas básicas.	63
Código A.3. Instalación del IDE y código del proyecto.	64
Código A.4. Creación de la base de datos.	65
Código A.5. Configuración de la aplicación	65
Código A.6. Instalación de WildFly con Galleon.	66
Código A.7. Despliegue del conector de la base de datos.	66
Código A.6. Despliegue de Sigma4Lifts.	67
Código A.7. setup.sh	67

NOTACIÓN

API	Application Programming Interface. Interfaz de programación de aplicaciones.
HTML	HyperText Markup Language. Lenguaje de marcado de hipertexto.
ISO	International Organization for Standardization. Organización Internacional de Normalización
JSON	JavaScript Object Notation. Notación de Objetos de JavaScript.
XML	eXtensible Markup Language. Language de Marcas Extensible.
UI	User Interface. Interfaz de usuario.
POM	Project Object Model. Modelo de objeto de proyecto.
MVC	Model-view-controller. Modelo-vista-controlador.
MVP	Model-view-presenter. Modelo-vista-presentador.
XPath	XML Path Language. Language de rutas XML.
CSS	Cascading Style Sheets. Hojas de estilo en cascada.
GUI	Graphical User Interface. Interfaz gráfica de usuario.
JDK	Java Development Kit. Kit de Desarrollo de de Java.
JRE	Java Runtime Environment. Entorno en tiempo de ejecución de Java.
WAR	Web Application Archive. Archivo de la aplicación web.
JAR	Java Archive. Archivo Java.
IDE	Integrated Development Environment. Entorno de Desarrollo integrado.

1. MOTIVACIÓN Y OBJETIVOS

*La calidad es ante todo prestar un servicio y satisfacer al cliente.
Ocupémonos más del cliente y la calidad se ocupará de sí misma.*

- Brian Codling -

1.1. Motivación

Los clientes cada vez se vuelven más selectivos y comienzan a rechazar productos poco fiables o que no dan respuesta a sus necesidades, por lo que la estabilidad de un sistema es un requerimiento mínimo e indispensable. A diferencia de un edificio o un dispositivo móvil, el software es un producto abstracto al igual que su calidad. Se desarrolla, no se fabrica, por lo que el costo está fundamentalmente en el diseño y no en la producción. Además no se deteriora con el tiempo: todos los problemas que surgen durante el mantenimiento estaban desde el principio y afecta a todas las copias del mismo. La aparición de un error en el producto tiene un efecto mucho más negativo en las fases de producción que en las etapas iniciales del proyecto, por lo que el control de calidad del mismo es esencial para evitar pérdidas y reducir costes.

En 2002 un estudio encargado por el National Institute of Standards and Technology del Department of Commerce en Estados Unidos concluyó que los errores informáticos costaban a la economía estadounidense 59.500 millones de USD al año, un 0.6% del PIB.

Dos accidentes de avión Boeing 737 MAX en 2018 y 2019 llevaron a que el 13 de marzo de 2019 la FAA (Administración Federal de Aviación) iniciara la suspensión de vuelo del Boeing 737 MAX durante meses por fallos en el software MCAS, falleciendo 346 personas en los accidentes. En marzo de 2019 quedaron en tierra 387 aviones MAX que realizaban 8600 vuelos semanales para 59 compañías aéreas, perdiendo 18.400 millones de dólares.

Aun así, un software con errores no se rechaza porque se asume que es inevitable los presente. Al fin y al cabo, estos sistemas están diseñados por el ser humano y este comete fallos, ya sea por presión de plazos de entrega, entorno de trabajo, complejidad del sistema o malinterpretaciones.

1.2. Objetivos

En este proyecto se pretende llevar a cabo pruebas software funcionales. Se caracterizan por ser de tipo caja negra basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Dicho de otro modo, son pruebas específicas, concretas y exhaustivas para probar y validar que el producto hace lo que debe y, sobre todo, lo que se ha especificado. Ya que las pruebas funcionales son de tipo caja negra, estas pueden ser ejecutadas sin tener conocimiento de los mecanismos internos del software bajo análisis. Uno de los beneficios que surgen a raíz de esto es la reducción del sesgo de confirmación en las pruebas ya que el tester, al contrario que el desarrollador, no ha estado involucrado en el desarrollo.

La solución propuesta en este trabajo se trata de un proyecto de automatización de pruebas de regresión para una aplicación web de gestión de una flota de ascensores. En él se cubren las funcionalidades principales de la misma para garantizar que en las actualizaciones del software no generen errores. Algunas de estas pruebas están enfocadas a casos de uso típicos de la aplicación como alta/baja de ascensores, gestión de técnicos, búsqueda de información, etc. Por otro lado, otro conjunto de pruebas están diseñadas para comprobar la persistencia de datos de la base de datos de la aplicación, editando información de elementos de la web para comprobar su permanencia.

El otro objetivo del trabajo es ofrecer al tester una forma de desplegar la aplicación autocontenida e inmediata en su propia máquina. Para ello nos ayudaremos de software de virtualización como VirtualBox y Vagrant, con lo que podremos desplegar el software, la base de datos y un IDE con el código del proyecto.

2. PRUEBAS DE SOFTWARE

2.1. Concepto de calidad

La calidad de software es una preocupación a la que se dedican muchos esfuerzos. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios.

En el desarrollo, la calidad de diseño acompaña a la calidad de los requisitos, especificaciones y diseño del sistema. La calidad de concordancia es un aspecto centrado principalmente en la implementación; Si la implementación sigue al diseño y el sistema resultante cumple con los objetivos de requisitos y de rendimiento, la calidad de concordancia es alta. A continuación vamos a ver los tipos de pruebas de software para mejorar la calidad del mismo.

2.2. Pruebas de software

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Es una actividad más del control de calidad.

Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno le corresponde un nivel distinto de involucramiento en las actividades de desarrollo.

Definimos caso de pruebas como el mecanismo, manual o automático, de verificar si el comportamiento del sistema es el deseado o no [1].

Teniendo esta afirmación en mente, la información que puede ser requerida es de lo más variada. Esto hace que el proceso de testing sea completamente dependiente del contexto en el que se desarrolla. El ambiente ideal de las pruebas es aquel que es independiente del desarrollo del software, de esta manera se logra objetividad en las pruebas.

A pesar de lo que muchos promueven, no existen las mejores prácticas como tales. Toda práctica puede ser ideal para una situación, pero completamente inútil o incluso perjudicial en otra. Por esto, las actividades técnicas, documentación, enfoques y demás elementos que condicionarán las pruebas a realizar deben ser seleccionadas y utilizadas de la manera más eficiente según el contexto del proyecto.

2.3. Tipos de pruebas

En este apartado vamos a repasar los tipos de prueba de software más relevantes según distintos aspectos como el tipo de ejecución, su enfoque o la verificación que realizan.

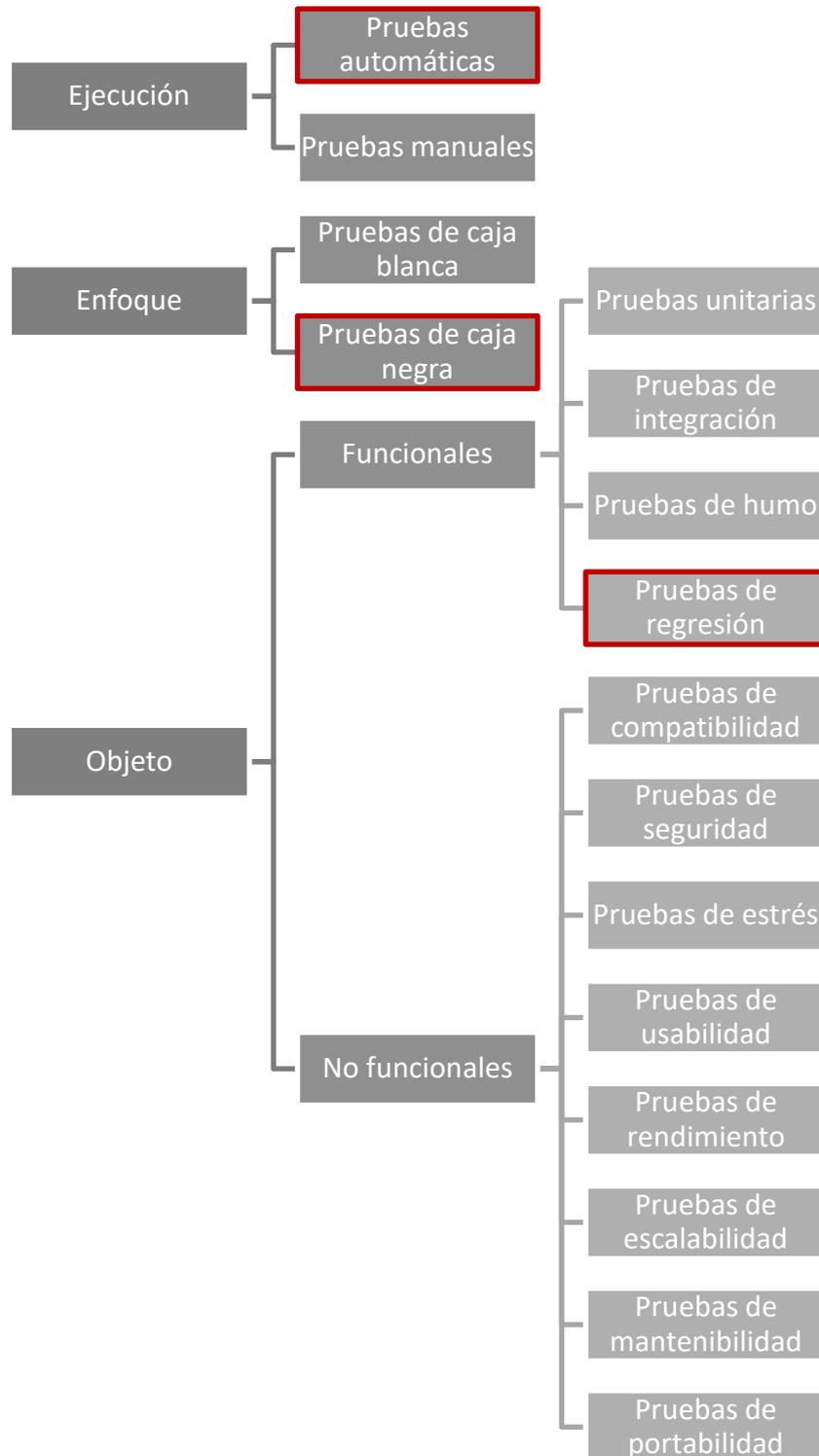


Figura 2.1. Tipos de pruebas de software.

2.3.1. Tipos de pruebas por su ejecución

2.3.1.1. Pruebas manuales

Las pruebas manuales son ejecutadas directamente por uno o más testers, simulando las acciones del usuario final. La repetición de pruebas manuales podría dejar pasar defectos, ya que una acción repetitiva para una persona es cansada y tediosa. Además, pueden llegar a consumir una gran cantidad de tiempo. Sin embargo, permiten un análisis profundo por parte del tester, lo cual es un mayor beneficio cuando se quiere mejorar la experiencia de usuario.

2.3.1.2. Pruebas automáticas

La automatización de pruebas consiste en el uso de software especial (casi siempre separado del software que se prueba) para controlar la ejecución de pruebas y la comparación entre los resultados obtenidos y los resultados esperados. La automatización de pruebas permite incluir pruebas repetitivas y necesarias dentro de un proceso formal de pruebas ya existente o bien adicionar pruebas cuya ejecución manual resultaría difícil [2].

2.3.2. Tipos de prueba por su enfoque

2.3.2.1. Caja blanca

En programación, se denomina cajas blancas a un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo [3]. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las de caja blanca están dirigidas a las funciones internas. Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas (integración).

2.3.2.2. Caja Negra

En teoría de sistemas y física, una caja negra es un elemento que se estudia desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno [4]. En otras palabras, de una caja negra nos interesará su forma de interactuar con el medio que le rodea (en ocasiones, otros elementos que también podrían ser cajas negras) entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace. Por tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento.

2.3.3. Tipos de prueba según lo que verifican

2.3.3.1. Pruebas funcionales

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Hay de distintos tipos, como por ejemplo:

2.3.3.1.1. Pruebas unitarias

En programación, una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código [5]. Por ejemplo en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, verificamos que sea correcto los nombres y tipos de los parámetros, el tipo de lo que se devuelve. Si el estado inicial es válido, entonces el estado final es válido también.

2.3.3.1.2. Pruebas de integración

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias y lo que prueban es que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo [6]. Se centra principalmente en probar la comunicación entre los componentes y sus comunicaciones ya sea hardware o software.

2.3.3.1.3. Pruebas de humo

En ingeniería de software y pruebas de software, las pruebas de humo son una revisión rápida de un producto de software para comprobar que funciona y no tiene defectos evidentes que interrumpen la operación básica del mismo [7]. Son pruebas que pretenden hacer una evaluación inicial de la calidad de un producto de software previo a una recepción formal, ya sea al equipo de pruebas o al usuario final.

2.3.3.1.4. Pruebas de regresión

Las pruebas de regresión son cualquier tipo de pruebas de software con el objeto de descubrir errores (bugs), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, causados por la realización de un cambio en el programa [8]. Se evalúa el correcto funcionamiento del software desarrollado frente a evoluciones o cambios funcionales. El propósito de éstas es asegurar que los casos de prueba que ya habían sido probados y fueron exitosos permanezcan así. Se recomienda que este tipo de pruebas sean automatizadas para reducir el tiempo y esfuerzo en su ejecución.

Las pruebas de regresión se pueden considerar como el subconjunto de pruebas planificadas que se seleccionan para ser ejecutadas, generalmente de forma automática y periódicamente en cada nueva liberación del producto, teniendo como objetivo la verificación de que el producto no haya sufrido regresiones.

2.3.3.2. Pruebas no funcionales

Una prueba no funcional es una prueba cuyo objetivo es la verificación de un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema como por ejemplo la disponibilidad, accesibilidad, usabilidad, mantenibilidad, seguridad o rendimiento. Podemos clasificar las pruebas no funcionales según el tipo de requisito no funcional que abarcan: pruebas de compatibilidad, seguridad, estrés, usabilidad, rendimiento, escalabilidad, mantenibilidad, portabilidad, etc.

Como se muestra en la Figura 2.1, en este proyecto vamos a realizar pruebas automáticas de caja negra de regresión. Es por ello que no vamos a desarrollar todos los tipos de prueba no funcionales. No obstante, se deja como propuesta en las líneas de trabajo futuras la implementación de pruebas no funcionales para la aplicación.

3. DEPENDENCIAS Y LIBRERÍAS

En este capítulo vamos a explicar las principales herramientas, dependencias, frameworks y librerías utilizadas para desarrollar el framework de automatización. Todas ellas son software libre o de código abierto y tienen un gran soporte por parte de la comunidad de desarrolladores.

3.1. Herramientas de automatización de pruebas para interfaces de usuario web

Para realizar las pruebas de automatizadas es necesaria una herramienta que sea capaz de interactuar con la interfaz de usuario de la aplicación web. Tenemos Protactor, capaz de identificar elementos web de JavaScript y AngularJS, aunque es un proyecto comunitario en evolución con la mayoría de documentación obsoleta. También tenemos otras alternativas como Katalon Studio o Micro Focus UFT que, aunque ofrecen un muy buen servicio, no son de código abierto y tienen un precio elevado. Es por esto que la mejor alternativa es Selenium, no solo por ser gratuito, sino por la inmensa documentación actualizada y soporte de la comunidad.

3.1.1. Selenium

Selenium es un entorno de pruebas de software libre para aplicaciones basadas en la web [9]. Dichas pruebas pueden ejecutarse usando la mayoría de navegadores web modernos en diferentes sistemas operativos como Windows, Linux y OSX. Esto es perfecto para realizar la automatización de una prueba de funcionalidad de una página web ahorrando tiempo en el desarrollo. Para ello haremos uso de Selenium WebDriver.

Selenium WebDriver acepta comandos y los envía a un navegador. Esto se implementa a través de un controlador específico para cada navegador que trae los resultados de regreso. En Selenium WebDriver no se requiere de un servidor especial para ejecutar las pruebas, en vez de ello inicia una instancia del navegador y lo controla.

Además, una ventaja del WebDriver es que actuaría como una persona real. Si aparece un cuadro de texto que está deshabilitado en la página web, una persona “real” no podría escribir en dicho cuadro al igual que el WebDriver. Sin embargo, otras herramientas si podrían, lo que le restaría realismo a la prueba.

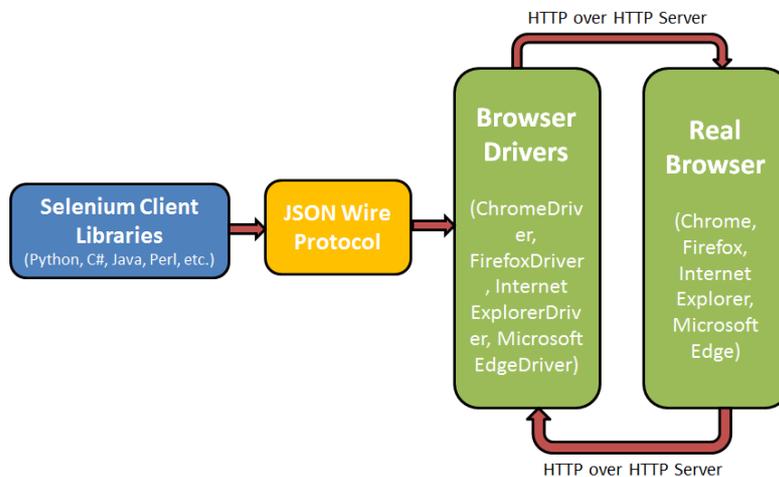


Figura 3.1. Arquitectura Selenium WebDriver.

3.2. Herramientas para la gestión y construcción de proyectos software

Durante el desarrollo de software es necesario construir y reconstruir el mismo código una y otra vez. Normalmente se utilizan scripts o ejecutables para automatizar estas acciones. Sin embargo, existen herramientas de construcción disponibles que son más apropiadas para esta tarea. Las principales alternativas son Ant, Maven y Gradle.

Ant tiene como principal ventaja el control sobre el proceso de construcción. Al tratarse de casi un lenguaje de programación, te permite hacer cualquier proceso, aunque requiere de mucho más conocimiento. Sin embargo, ha dejado de ser el estándar por la llegada de nuevas herramientas que facilitan mucho más esta labor.

Por otro lado, Maven fue creado para resolver los problemas de Ant. Mantuvo los ficheros de configuración en XML pero cambió el rumbo de la solución. En Ant los desarrolladores tienen que crear todas las tareas. En Maven disminuye la creación de estas tareas por la implementación de estándares robustos para organizar el código. Como resultado: tenemos una herramienta más sencilla de utilizar. Además, introdujo la descarga automática de dependencias, lo que agilizó la parte de desarrollo.

Por último, Gradle combina el poder de Ant y Maven. En lugar de usar ficheros XML, utiliza el lenguaje Groovy. Como resultado, los scripts de construcción en Gradle son mucho más fáciles de entender y escribir. No obstante, su integración con los IDEs no es tan buena como la de Maven.

Por la naturaleza del proyecto la mejor opción es Maven, ya que tenemos que importar varias librerías y tiene una muy buena integración con el entorno de desarrollo.

3.2.1. Maven

Es una herramienta de software libre para la gestión y construcción de proyectos Java. Es similar en funcionalidad a Apache Ant, pero tiene un modelo de configuración más simple basado en un formato XML [10]. Maven utiliza un Project Object Model (POM) para describir el proyecto software a construir, sus dependencias de otros módulos y componentes externos y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores.

Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios.

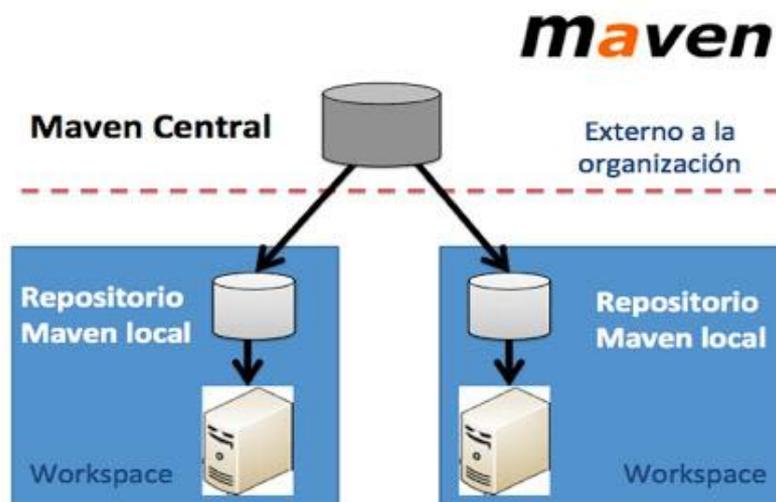


Figura 3.2. Arquitectura Maven.

3.3. Framework para pruebas

Para la automatización de las pruebas es esencial el uso de un framework. Ayudará a que el código de automatización sea reutilizable, mantenible y estable. A no ser que tengamos una ingente cantidad de tiempo y recursos, crear nuestro propio framework de pruebas no es una opción. Por ello lo mejor es revisar algunas de las alternativas disponibles basadas en código abierto y evaluar sus capacidades.

JUnit es un framework de pruebas de código abierto usado para pruebas unitarias en Java. El propósito clave de esta herramienta es permitir a los desarrolladores diseñar e implementar casos de prueba. Sin embargo, no es el más apropiado para hacer pruebas de alto nivel o ejecutar suites largas, característica principal de las suites de regresión.

Serenity es un framework basado en Java que se integra a la perfección con herramientas de desarrollo como Cucumber y JBehave. Está diseñado para facilitar la escritura de pruebas automáticas de aceptación y regresión pues también le permite abstraer gran parte del código repetitivo que a veces necesita escribir para comenzar a usar herramientas como Selenium WebDriver. Sin embargo, crear y mantener los casos requiere de mucho más tiempo que el resto de alternativas.

Finalmente nos decidimos por TestNG, principal competidor de JUnit. Los dos están basados en Java y te permiten automatizar tests y escrutar sus resultados. Sus principales ventajas son que te permite ejecutar tests en paralelo, crear un reporte HTML, organizar mejor los casos de prueba, añadir prioridades, etc. Por último, tiene una muy buena integración con la herramienta para generación de reportes que explicaremos en el siguiente apartado.

3.1.1. TestNG

Se trata de un framework para pruebas y testing que trabaja con Java y está basado en JUnit y NUnit, pero introduciendo nuevas funcionalidades que los hacen más poderosos y fáciles de usar [11]. Será necesario en el proyecto para la creación del reporte final de la ejecución. Entre otras cosas permite relanzar tests fallidos el número deseado de veces antes de declararlo como fallido.

3.4. Herramientas de reportes para Selenium

Cuando hablamos de testing con Selenium, generar un informe detallado usando la herramienta adecuada puede mejorar drásticamente las pruebas. De nada sirve tener unas pruebas de software muy buenas si luego el informe que generan es pobre o difícil de leer. Elementos como gráficos, colores y estadísticas permiten, con un simple vistazo, entender la situación general de la ejecución. Además es importante que permita, en caso de que sea necesario, acceder a la información más detallada de cada test para discernir el motivo de su fallo.

La primera de ellas es TestNG. Como hemos mencionado anteriormente, este framework genera un reporte en formato HTML. Como podemos ver en la Figura 3.3 es bastante anticuado y poco estético, así que no lo utilizaremos como herramienta de reportes.

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Suite						
Test	2	0	0	27,961		

Class	Method	Start	Time (ms)
Suite			
Test — passed			
com.test.DemoA	run	1471859670194	13509
com.test.DemoB	run	1471859655754	14438

Test

com.test.DemoA#run

[back to summary](#)

com.test.DemoB#run

[back to summary](#)

Figura 3.3. Ejemplo reporte TestNG.

Para resolver el problema de la herramienta anterior existe ReportNG, un plugin de TestNG que mejora notablemente la presentación del informe. Como vemos en la Figura 3.4, utiliza más colores para distinguir la información y facilita el entendimiento de la ejecución.

ReportNG Sample	Duration	Passed	Skipped	Failed	Pass Rate
Only Successful Tests	0.000s	4	0	0	100%
All Sample Tests	0.009s	10	3	4	58%
Total		14	3	4	66%

Another Suite	Duration	Passed	Skipped	Failed	Pass Rate
Only Skipped Tests	0.000s	0	2	0	0%
Total		0	2	0	0%

Figura 3.4. Ejemplo reporte ReportNG.

Sin embargo, existe una mejor alternativa en cuanto a la visualización del reporte: ExtentReport. Se trata de una librería que también genera un fichero HTML como sus contrapartes, aunque con una mayor legibilidad y claridad. Dichos informes son mas limpios, detallados e interactivos. Además cuenta con una documentación muy exhaustiva que facilita su implementación. Otra ventaja que tiene es que puede personalizar la plantilla del reporte utilizando CSS y XML. En la Figura 3.5 podemos observar un ejemplo de un ExtentReport.

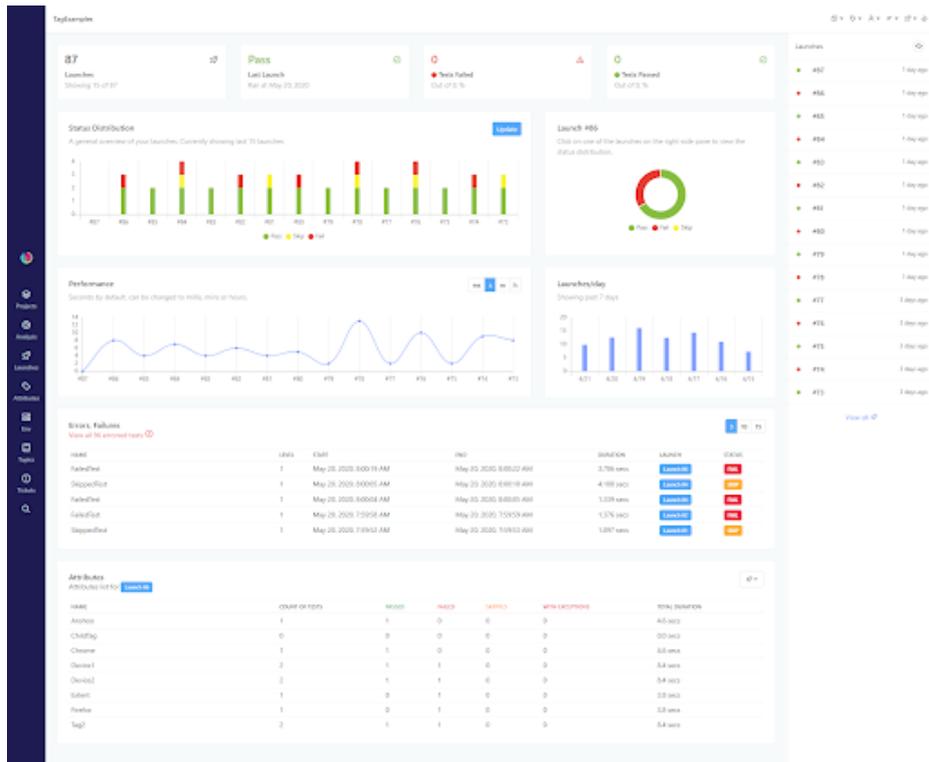


Figura 3.5. Ejemplo reporte ExtentReport.

3.4.1. Relevantcodes

Como hemos mencionado anteriormente, los ExtentReport se pueden personalizar utilizando CSS y ficheros XML. Relevantcodes es una librería de código abierto que adapta los Extent Reports para Selenium y TestNG ya personalizados. Permite añadir eventos, capturas de pantalla, etiquetas o cualquier información relevante para las pruebas. Podemos ver un ejemplo en la Figura 3.6.

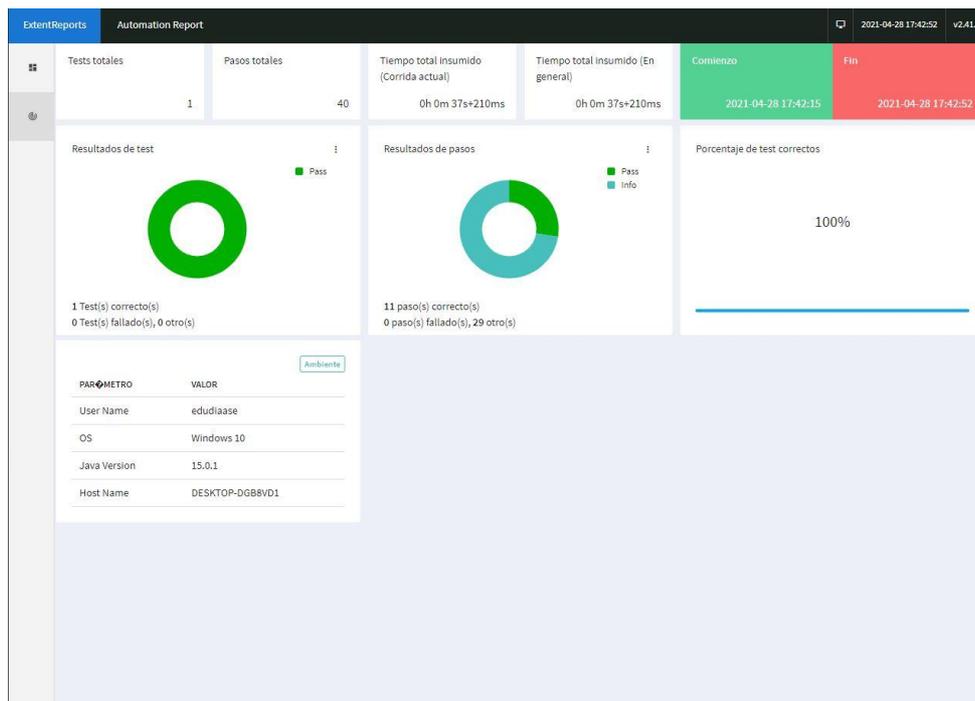


Figura 3.6. Ejemplo reporte ExtentReport personalizado con Relevantcodes.

3.5. Herramientas para generación de trazas

Manejo de errores, mensajes de depuración, auditoría y archivos de log son diferentes aspectos del mismo tópico: como realizar un seguimiento de eventos dentro de una aplicación. Una de las principales necesidades cuando estamos revisando las trazas de una aplicación es que aporten suficiente información y sean fáciles de consultar. Para esto es vital que definamos unas buenas trazas a lo largo del código y que sus niveles estén bien definidos.

Por otro lado, el hecho de utilizar trazas estructuradas nos facilita enormemente el poder ver que datos se estaban utilizando en el momento concreto y el poder analizar los datos de una manera más eficiente. Gracias a su inmediata implementación con Maven, Log4j es la herramienta de monitorización de trazas utilizada en este proyecto.

3.5.1. Log4j

Es una biblioteca open source desarrollada en Java por la Apache Software Foundation que permite a los desarrolladores de software escribir mensajes de registro, cuyo propósito es dejar constancia de una determinada transacción en tiempo de ejecución [12]. Log4j permite filtrar los mensajes en función de su importancia: OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL ordenados decrecientemente. Ha sido de gran ayuda para dejar un rastro de la ejecución y añadir trazas de debug en caso de errores.

4. PROYECTO DE AUTOMATIZACIÓN

Una vez mencionadas todas las librerías y dependencias del mismo, vamos a explicar la arquitectura del proyecto con ayuda de diagramas y patrones de diseño. Además haremos un repaso de los recursos del proyecto y la configuración del mismo.

4.1. Patrones de diseño

Si durante la automatización de pruebas nos encontramos muchas veces con los mismos problemas, ¿por qué no aplicamos un patrón de diseño? Si en el desarrollo de software los patrones de arquitectura son de lo más habituales (MVC, MVP, Presentation Model, etc), ¿no ocurre lo mismo cuando automatizamos? Si el coste de mantener nuestras pruebas automatizadas es demasiado alto por código duplicado o complejidad, ¿por qué no usamos una estructura que ayude a simplificarlo? Con la idea de minimizar en gran parte estos problemas, nos ayudaremos de algunos patrones de diseño.

4.1.1. Patrón Page Object

El concepto básico en el que se basa este patrón es el de representar cada una de las pantallas que componen el sitio web o la aplicación que nos interesa probar, como una serie de objetos que encapsulan las características y funcionalidades representadas en la página. De esta manera nos permite consolidar el código para interactuar con los elementos de una pagina en cada uno de los PageObjects.

Al crear un PageObject, lo que estamos consiguiendo es crear una capa de abstracción entre el “¿Qué podemos hacer/ver en esta página?” y el “¿Cómo se realiza esta acción?”. Esto simplifica enormemente la creación de nuevas pruebas y reutilizando el código con el que interactuamos con la página en concreto. Además, con esto conseguimos que cualquier cambio que se produzca en la UI únicamente afectará al PageObject en cuestión y no a los tests ya implementados.

Esto se debe a que un test nunca debe manipular directamente elementos de la página (UI), sino que este manejo debe realizarse a través del PageObject correspondiente que representa la página. Para entendernos, el PageObject se convierte en una “API” con la que fácilmente podemos encontrar y manipular los datos de la página.

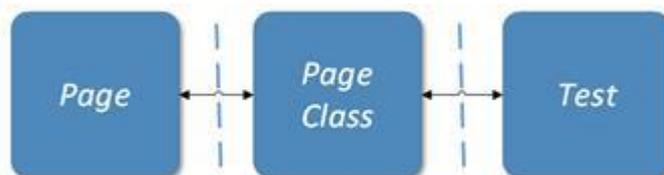


Figura 4.1. Patrón Page Object.

Además, disponemos de una clase abstracta base de la que heredarán todos los PageObject. En ella se implementarán métodos comunes a todas las páginas, como las acciones básicas que nos ofrece Selenium: clicks, scrolls, doble click, inputs o cualquier lectura del documento html. A continuación vamos a ver un ejemplo de una de las páginas de la web.

Esta es la página de un cliente. En ella podemos editar su email, nombre, idioma o cualquiera de los campos presentes.

Figura 4.2. Página cliente de Sigma4Lifts.

Primero debemos crear un PageObject para dicha página. En dicho objeto almacenaremos todos los elementos de utilidad de la misma, como los recuadros de input de cada uno de los campos de información o los botones de “Editar” y “Cancelar”.

Código 4.1. PageObject para la página de los clientes UserDataPage.java

```
public class UserDataPage extends BasePage {

    @FindBy(id = "j_id64:btnModificar")
    private WebElement editButton;
    @FindBy(id = "j_id64:btnGuardar")
    private WebElement saveButton;
    @FindBy(id = "j_id64:inSegundoApellido")
    private WebElement apellido2Input;

    public UserDataPage(Browser browser) {
        super(browser);
    }
}
```

```

    public void checkUserIsDisplayed(String login) {
        Assert.assertTrue(isDisplayed(getElement(By.xpath("//td[.=' " + login
+ "']"))), "User with login " + login + " is displayed");
    }

    public void clickEdit() {
        click(editButton);
    }

    public void clickSave() {
        click(saveButton);
    }

    public void editApellido2(String value) {
        sendKeys(apellido2Input, value);
    }

    public void checkApellido2(String value) {
        Assert.assertTrue(getValue(apellido2Input).equals(value), "Value " +
value + " is correct.");
    }
}

```

Lo primero que podemos observar es que la clase hereda de `BasePage`, que es la base de todos los `PageObjects`. Gracias a ello tenemos implementados métodos como `click` o `sendKeys` presentes en el código. Cada elemento del HTML con el que queremos interactuar se guardará como atributo de la clase. Para hacer referencia a ellos utilizamos los `id` o `XPath`. Es buena práctica siempre que sea posible recurrir a los `id` antes que a los `XPath` porque así aseguramos que dicho elemento sea único en toda la página. Por ejemplo, si queremos hacer click en “Editar”, creamos el método `clickEdit`, o si queremos comprobar que el segundo apellido tiene un cierto valor, hacemos un `assertion` como podemos ver en el método `checkApellido2`. Es importante que los métodos sean lo más atómicos posible para facilitar la reutilización del código al máximo.

4.1.2. Patrón Test-Actions-Page

El objetivo de este patrón es la división en tres capas de abstracción las interacciones con Selenium para ayudar a la comprensión y estructuración de los tests. Con esto conseguimos que una persona poco familiarizada con el framework sea capaz de entender a simple vista qué está haciendo una prueba automatizada. En el anterior punto hemos explicado la primera capa del modelo con el `Page Object`. Pasemos ahora a explicar la siguiente capa: `Actions`.

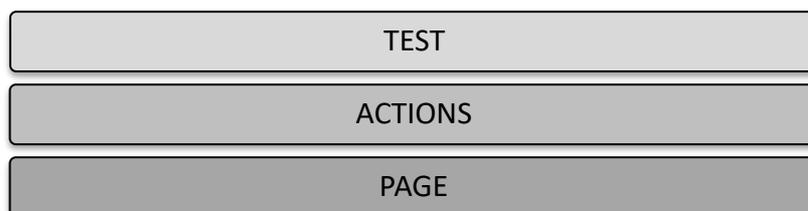


Figura 4.3. Representación patrón Test-Actions-Page.

4.1.2.1. Actions

Supongamos que estamos en la página de búsqueda de los clientes. En ella podemos realizar acciones como buscar clientes por nombre, apellido, email o incluso login.

Sem.22 28/05/2021 13:28 Te encuentras en: INICIO >> ADMINISTRACIÓN >> CLIENTES CERRAR SESIÓN

Empresa: SIGMA4LIFTS
Usuario: EDUARDO DIAZ ASENCIO
Perfil: ADMINISTRADOR MP

interlift 2015

Criterios de Búsqueda

Nombre: Primer apellido:
Segundo apellido: Login:
Email: Estado: Activo Inactivo Todos

BUSCAR LIMPIAR

ACTIVO	LOGIN	NOMBRE	EMAIL
<input checked="" type="checkbox"/>	S4L005	Garcia Nuñez, Antonio	ejm@gmail.com
<input checked="" type="checkbox"/>	S4L025	apellido_1_cliente 2021.04.19.18.20.08, nombre_cliente	cliente@gmail.com

2 elementos encontrados

ALTA DE CLIENTE

mp
mp4lifts.com

Figura 4.4. Página búsqueda clientes Sigma4Lifts.

Sea cual sea el método de búsqueda, todos tienen en común que una vez rellenen los campos pertinentes, harán click en el botón de “Buscar” y comprobarán que arroja un resultado válido. El objetivo de esta nueva capa es doble: reutilizar el código de los PageObjects y agruparlos en métodos de más alto nivel. Esto significa que para buscar un cliente nombre, existe un método específico para ello. Para buscar un cliente por apellido, existe otro método específico también. Para comprenderlo vamos a ayudarnos del siguiente diagrama:

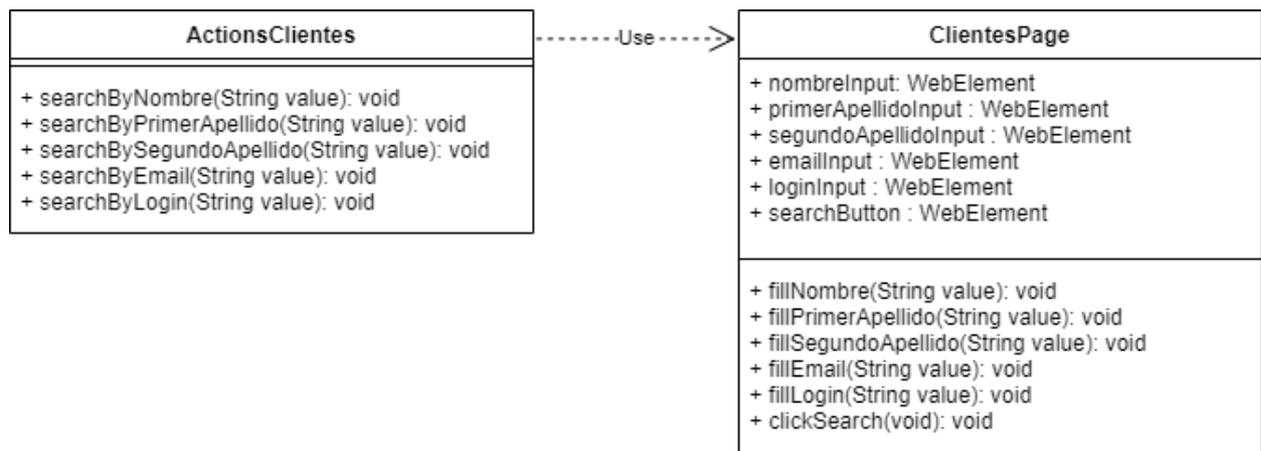


Figura 4.5. Diagrama de clases entre Actions y Pages.

Como podemos ver conseguimos simplificar todo en acciones concretas como buscar un elemento a partir de un cierto valor como puede ser el nombre o el login del usuario, quedando el código parecido a esto:

Código 4.2. Método searchByLogin de ActionsClientes.java

```

public void searchByLogin(String login) {
    ClientesPage clientesPage = new ClientesPage(browser);
    clientesPage.openCriteriosDeBusqueda();
    clientesPage.fillLogin(login);
    clientesPage.clickSearch();
    clientesPage.checkLiftWithLoginIsDisplayed(login);
    clientesPage.seeDetails(login);
}
  
```

4.1.2.2. Test

Por último tenemos la capa de test. Esta se encarga de manejar las acciones necesarias para completar un caso de prueba concreto de tal forma que sea lo más cercana al lenguaje de negocio. Con esto logramos una fácil comprensión del código para aquel que esté poco familiarizado con el mismo. Vamos a mostrar el código de un test que busca la información de un cliente mediante su identificador de login:

Código 4.3. Caso de prueba para la búsqueda de un cliente en SearchClienteByLoginTest.java

```

public class SearchClienteByLoginTest extends BaseTest {

    @Test
    public void searchClienteByLoginTest() {
        setUpExtentTest(testDescription);
        acciones.loginPage().login(data.getUser());
        acciones.clientesPage().navigate();
        acciones.clientesPage().searchByLogin(
            data.getClientUser().getLogin());
        acciones.userDataPage().checkUserIsDisplayed(
            data.getClientUser().getLogin());
        acciones.logout();
    }
}
  
```

Como podemos ver en el código, lo que hace Selenium es bastante claro:

1. Inicia sesión en la página web.
2. Navega a la página de búsqueda de clientes.
3. Busca al cliente mediante el identificador de inicio de sesión.
4. Comprueba que la página web muestra el resultado esperado.
5. Cierra la sesión.

Si nos fijamos, todas las clases de test heredan de una clase padre, `BaseTest`. En ella se realizan una serie de acciones antes y después de la ejecución de la prueba. Primero inicializan la capa inferior, `Actions`. Además, crea una variable llamada “data” donde se almacena toda la información necesaria para el test: usuarios, ascensores, clientes, empresas, etc. Cuando acaba la prueba, se cierra el navegador y el driver asociado a él.

Código 4.4. `BaseTest.java`

```
public class BaseTest {
    final static Logger logger = Logger.getLogger(BaseTest.class);
    protected Actions acciones;
    protected ExtentTest test;
    protected SigmaData data = SigmaData.getDefaultData();

    @BeforeClass
    public void setUp() {
        acciones = new Actions();
    }

    @AfterClass
    public void tearDown() {
        acciones.closeBrowser();
    }

    public WebDriver getDriver() {
        WebDriver result = null;
        try {
            result = this.acciones.getBrowser().getDriver();
        } catch (Exception var1) {
            logger.error(var1.getMessage());
        }
        return result;
    }

    public void setUpExtentTest(String testDescription) {
        this.test = ExtentTestManager.startTest(getClass().getSimpleName(),
        testDescription);
        this.acciones.setExtentTest(test);
    }
}
```

Terminada de explicar la tercera capa del patrón, vamos a actualizar el diagrama de clases mostrado anteriormente para tener una visión más general.

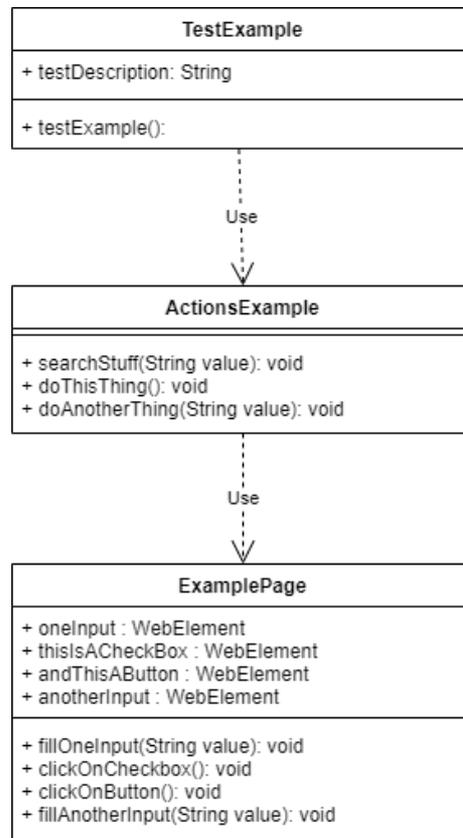


Figura 4.6. Diagrama de clases completo de las tres capas del patrón Test-Actions-Page.

4.1.3. Patrón Page Factory

Para soportar el modelo Page Object, usamos el Page Factory. Es una extensión y se puede usar de varias maneras. En este caso lo utilizaremos para inicializar los elementos web que se definen en las clases de página. Por cada método de la capa Actions, llamaremos al constructor del Page Object. Dicho constructor es heredado de la página base (recordemos que era el PageObject del que heredaban todas las demás).

Código 4.5. Método de ActionsClientes.java

```

public void navigate() {
    ClientesPage clientesPage = new ClientesPage(browser);
    clientesPage.navigate();
}
    
```

Primero el método navigate invoca al constructor del PageObject. Dicho método es llamado desde la capa de test.

Código 4.6. Constructor ClientesPage.java

```

public ClientesPage(Browser browser) {
    super(browser);
}
    
```

El constructor del PageObject es heredado del BasePage. Ese constructor inicializa los elementos de la página web en la que se encuentra actualmente el driver de Selenium.

Código 4.7. Constructor BasePage.java

```
public BasePage(Browser browser) {
    this.browser = browser;
    PageFactory.initElements(browser.getDriver(), this);
}
```

4.2. Organización del Proyecto

En este apartado vamos a desarrollar la organización seguida en el proyecto. Primero mostraremos los paquetes y dependencia entre paquetes de clases del framework. Después explicaremos dónde se encuentran los recursos del mismo y para qué se utilizan. Para finalizar haremos un repaso de la parte de configuración. En ella veremos como modificar variables parametrizadas como el lugar de la aplicación, la localización del WebDriver en el equipo o el timeout entre sentencias de Selenium.

4.2.1. Paquetes

Vamos a ayudarnos de un diagrama de paquetes general para tener una visión general. Después iremos paquete por paquete explicando lo que contiene, para lo que sirve y qué dependencia tiene con el resto.

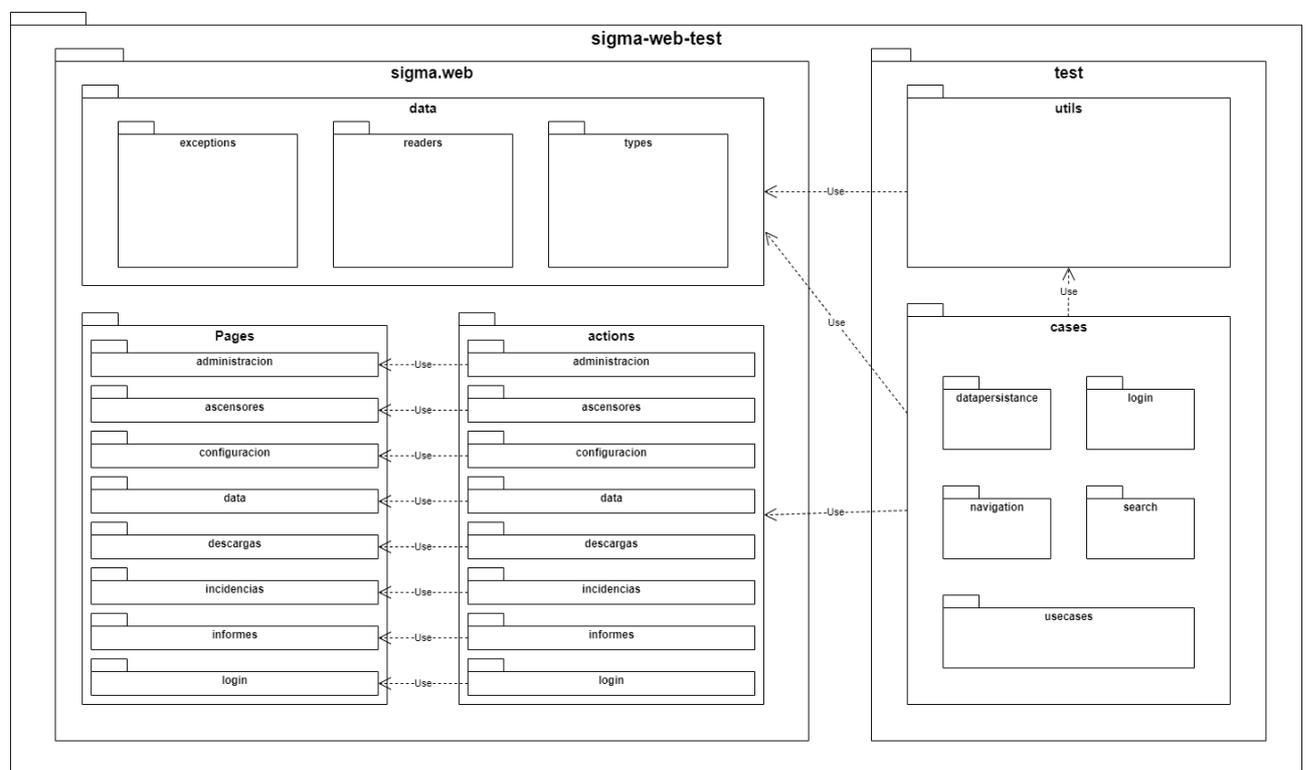


Figura 4.7. Diagrama de paquetes del proyecto.

- El paquete *cases* contiene todos los casos de prueba del proyecto. Éste a su vez se subdivide en cinco paquetes. Cada uno se encarga de probar un aspecto de la funcionalidad de la aplicación web. El paquete *datapersistence* hace cambios en la información de objetos persistentes en la web para comprobar la correcta comunicación con la base de datos. El paquete *login* hace pruebas para comprobar que el inicio de sesión (funcionalidad indispensable para el resto) no falle. El paquete *navigation* sirve para corroborar que la navegación entre las distintas páginas de la web funciona como debe. El paquete *usecases* contiene pruebas de los casos de uso más importantes de la aplicación: suscripciones de usuarios a ascensores, generar alarmas manuales, averías, etc.
- Los paquetes *actions* y *pages* contienen los Actions y PageObjects de cada una de las páginas de la web. Están subdivididos en grupos de páginas. Como podemos ver, el paquete de *cases* utiliza al de *actions*, que a su vez usa al paquete de *pages*. Esto es por el patrón Test-Actions-Pages que hemos definido previamente.
- El paquete *utils* contiene una serie de clases necesarias para generar el reporte de la ejecución final. Es utilizado por los casos de prueba para ir añadiendo logs y trazas en la ejecución. Si por ejemplo queremos que un caso de prueba se relance un número de veces hasta que sea exitoso, podemos configurarlo en la clase `Retry.java` dentro del paquete.
- El paquete *exceptions* contiene una serie de excepciones que heredan de `RuntimeException` para informar al desarrollador en qué fragmento del código está ocurriendo un error en tiempo de ejecución.
- El paquete *readers* contiene una serie de clases para leer ficheros de tipo JSON o properties. Los primeros los utilizamos principalmente para almacenar datos de JavaBeans necesarias en el testing. Los properties se usan para la configuración del framework de automatización.
- El paquete *types* contiene las JavaBeans mencionadas anteriormente. Tenemos varios tipos y todos alimentan su información a partir de la lectura de ficheros JSON. Esos tipos son empresas, ascensores, grupos de ascensores y usuarios.

4.2.2. Recursos

Es importante mencionar que tenemos dos directorios de recursos en el proyecto. El primero de ellos se encuentra bajo el directorio `main`. En él se encuentran ficheros de configuración y datos. El segundo está bajo el directorio `test`, donde se definen las suites de los casos de prueba. Aquí nos enfocaremos en el primero de ellos.

En el directorio `json` guardamos diversos archivos para almacenar información de las JavaBeans. Se dividen en empresas, ascensores, grupos de ascensores y usuarios. Los ficheros `properties` son ficheros de configuración, por lo que lo explicaremos en el siguiente punto.

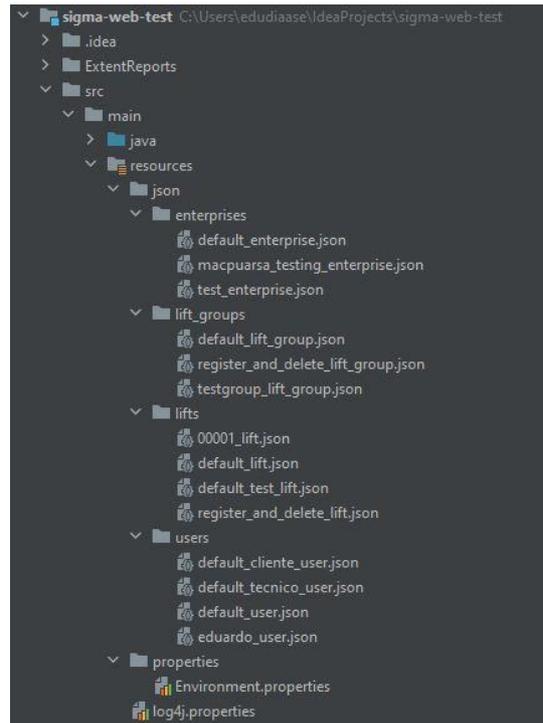


Figura 4.8. Directorio de recursos de configuración y datos.

4.2.3. Configuración

Hay ciertas variables que es importante que estén parametrizadas en algún lugar del proyecto porque aportan una gran adaptabilidad al código. En concreto hay tres variables que son muy importantes y le dedicamos esta sección en la memoria.

- URL de la aplicación: este valor nos indica en qué punto iniciará el WebDriver de Selenium la ejecución. Si por ejemplo tenemos varios entornos donde está desplegada, cada uno en versiones del código distintas, sería de gran utilidad poder apuntar a cualquiera de ellos de forma inmediata.
- Timeout del navegador: este valor indica el número de segundos máximo que puede ocurrir entre sentencias del WebDriver de Selenium. Si no lo hiciésemos así, la ejecución podría quedarse estancada en cualquier parte del código. Supongamos que queremos leer un campo de una etiqueta HTML pero la página se ha quedado colgada. El driver se quedaría parado intentando leer algo que no existe, deteniendo la ejecución en ese punto sin saber por qué ocurre.
- Ubicación del Geckodriver: aquí se especifica la ruta relativa desde el directorio raíz del proyecto hasta la ubicación del WebDriver. No es necesario modificarlo aunque si se quiere utilizar otro driver, simplemente se añade en la carpeta deseada y se actualiza el gecko.path.

Código 4.10. Environment.properties

```
environment.url=http://127.0.0.1:49001/sigma/pages/core/login.jsf
browser.execution.wait.timeout=5
gecko.path=/src/main/resources/Geckodriver/geckodriver
```

4.3. Ejecución de pruebas

Para la ejecución de pruebas tenemos un directorio de recursos debajo del directorio de test donde tenemos las diferentes suites que podemos ejecutar. Dichas suites están en formato XML y dentro de ellas se define qué test cases lanzar. Además, una suite puede lanzar otras suites, como el caso de la de regresión. En ella se apunta a todas las suites existentes, por lo que se lanzan todos los casos automatizados del proyecto. Todo esto es gracias a un plugin de Maven llamado SureFire y TestNG.

Dentro del fichero pom.xml tenemos un apartado donde configuramos la ubicación de las suites y pasamos mediante parámetros a Maven la suite que deseamos ejecutar.

Código 4.8. Fichero pom.xml

```
<properties>
  <maven.compiler.source>8</maven.compiler.source>
  <maven.compiler.target>8</maven.compiler.target>
  <suiteXmlPath>src/test/resources/suites/</suiteXmlPath>
  <suiteName></suiteName>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
...
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>3.0.0-M5</version>
  <configuration>
    <suiteXmlFiles>
      <suite>${suiteXmlPath}${suiteName}.xml</suite>
    </suiteXmlFiles>
  </configuration>
</plugin>
```

Vemos que dentro del plugin apuntamos a suite con un path y un nombre. El path está definido dentro de las etiquetas properties, mientras que el nombre de la suite lo pasamos por parámetro. Si por ejemplo queremos lanzar la suite *SearchSuite.xml*, tendríamos que ejecutar el siguiente comando:

```
$ mvn clean install -DsuiteName=SearchSuite
```

Si observamos el fichero de esa suite, vemos que a su vez llama a todos los test cases del paquete de *search*. De esta manera es como lograríamos de una manera muy sencilla lanzar los casos de prueba.

Código 4.9. SearchSuite.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="SearchSuite" >
  <listeners>
    <listener class-name="test.utils.TestListener"/>
    <listener class-name="test.utils.AnnotationTransformer"/>
  </listeners>
  <test name="Search tests">
    <classes>
      <class name="test.cases.search.SearchClienteByLoginTest"/>
      <class name="test.cases.search.SearchEmpresaByAbreviaturaTest"/>
    </classes>
  </test>
</suite>
```

```

        <class name="test.cases.search.SearchLiftGroupNameTest"/>
        <class name=
            "test.cases.search.SearchLiftInMapasByReferenciaTest"/>
        <class name=
            "test.cases.search.SearchLiftRelacionByReferenciaTest"/>
        <class name="test.cases.search.SearchTecnicoByLoginTest"/>
    </classes>
</test>
</suite>

```

En el caso de que queramos lanzar el test de SearchClienteByLoginTest.java aislado en lugar de una suite completa tenemos una suite específica para lanzar casos concretos, DebugSuite. Simplemente tendríamos que editar dentro de ella la etiqueta class apuntando al test case de interés:

Código 4.10. DebugSuite.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="DebugSuite" >
    <listeners>
        <listener class-name="test.utils.TestListener"/>
        <listener class-name="test.utils.AnnotationTransformer"/>
    </listeners>
    <test name="DebugSuite">
        <classes>
            <class name="test.cases.search.SearchClienteByLoginTest"/>
        </classes>
    </test>
</suite>

```

Después lanzamos el commando de Maven pertinente:

```
$ mvn clean install -DsuiteName=DebugSuite
```

4.4. Reporte

Una vez lanzados los casos de prueba de nada sirven si no se genera un reporte de ejecución. Para ello hemos mencionado anteriormente que utilizamos una dependencia de Maven llamada Relevantcodes, con la que generamos un ExtentReport. Dicho fichero se genera bajo el directorio ExtentReports y se abre con cualquier navegador. Dicho informe es interactivo y se pueden hacer cosas como filtrar los logs, buscar un test en específico o tener una visión general de todos. A continuación dejamos la visión general del informe, el informe concreto de un test y el informe concreto de un test con filtros, respectivamente.

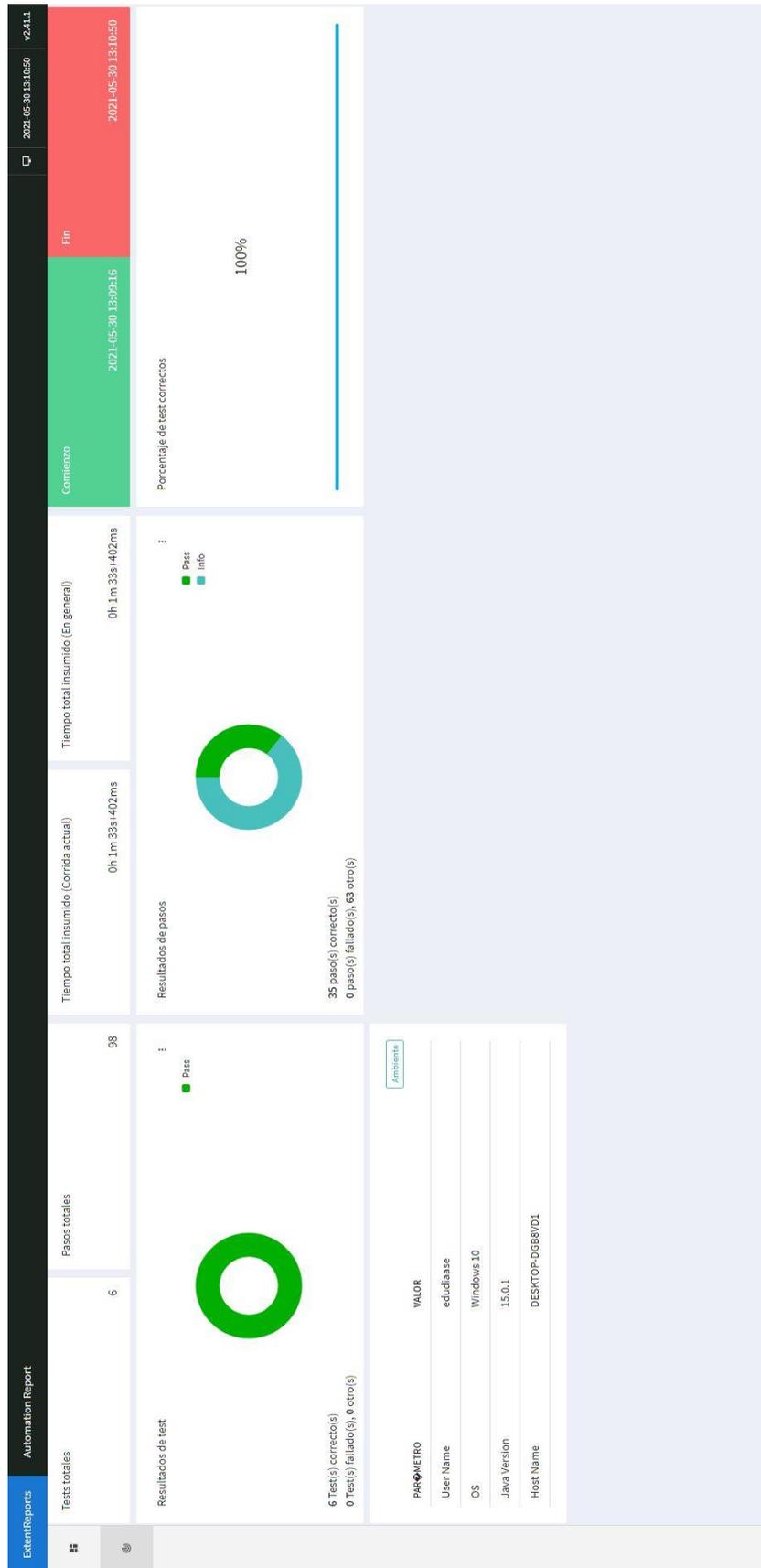


Figura 4.9. Visión general del reporte.

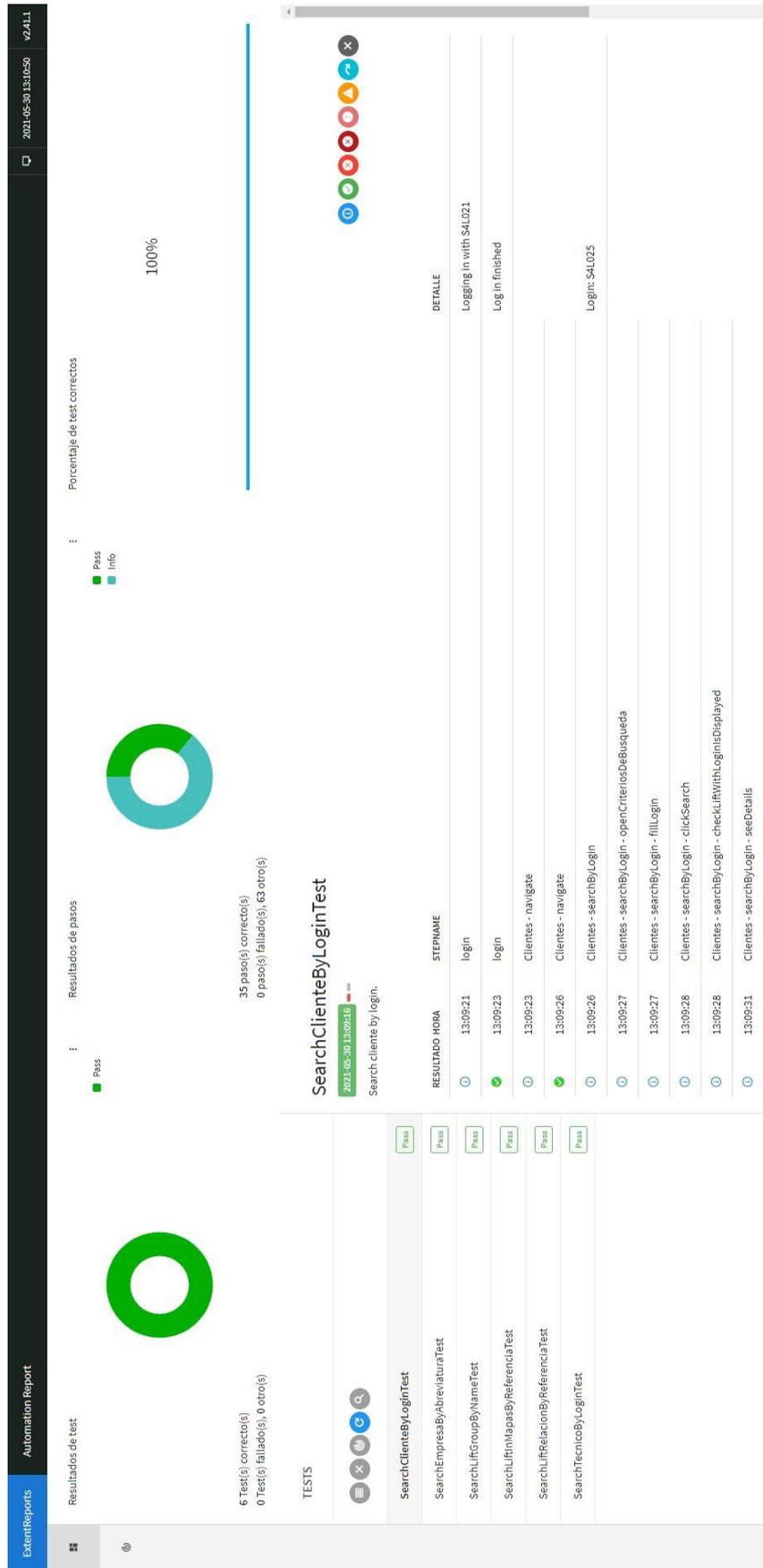


Figura 4.10. Reporte de test concreto sin filtros.

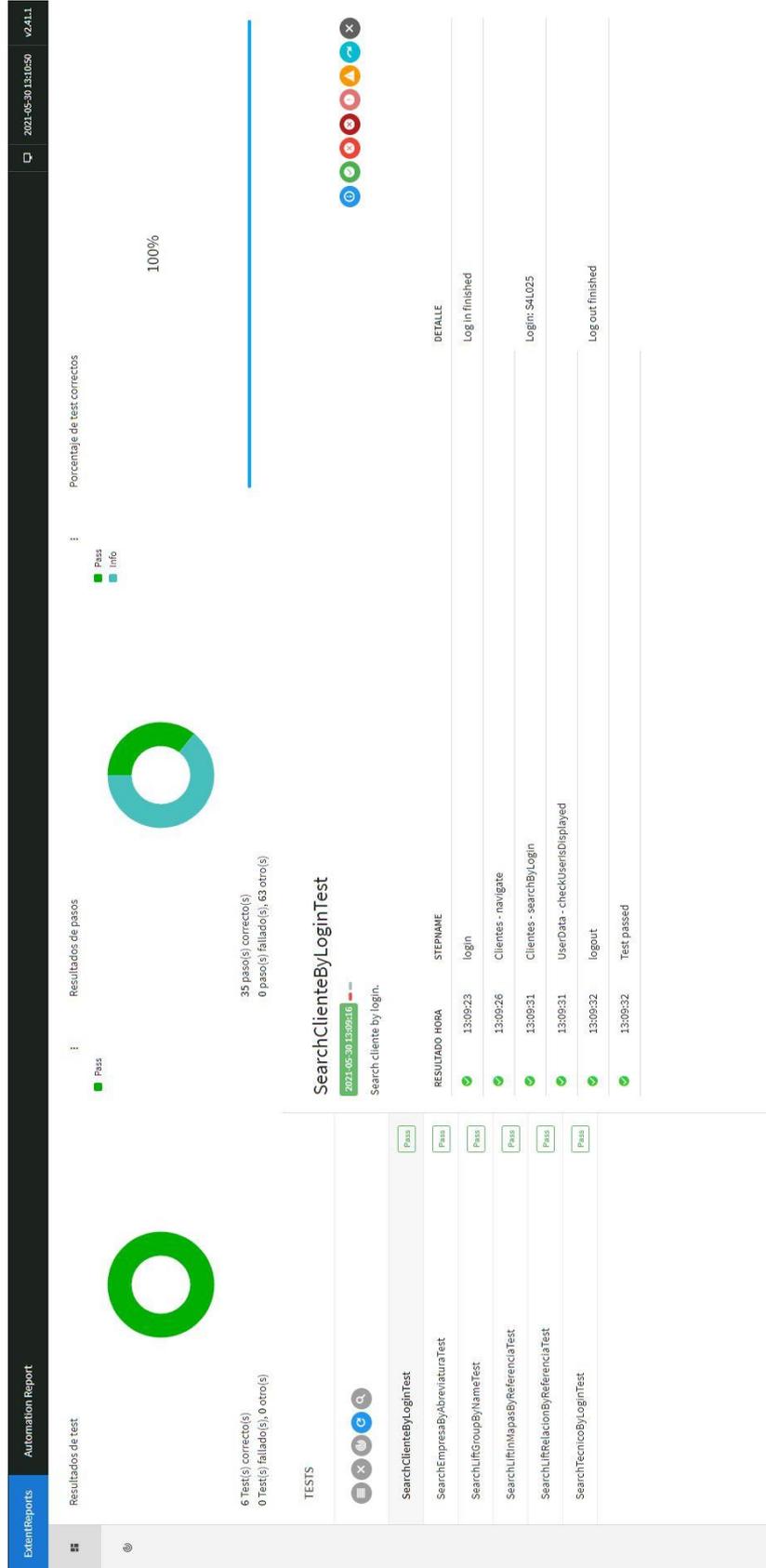


Figura 4.11. Reporte de test concreto con filtro de logs 'pass'.

Todos estos logs se alimentan desde el código en las clases de Action. Para ello mostramos el siguiente extracto de código donde se muestran las trazas y sus diferentes tipos.

Código 4.11. Trazas en método searchByLogin de ActionsClientes.java

```
public void searchByLogin(String login) {
    test.log(LogStatus.INFO, "Clientes - searchByLogin", "Login: " + login);
    ClientesPage clientesPage = new ClientesPage(browser);
    clientesPage.openCriteriosDeBusqueda();
    test.log(LogStatus.INFO, "Clientes - searchByLogin -
openCriteriosDeBusqueda");
    clientesPage.fillLogin(login);
    test.log(LogStatus.INFO, "Clientes - searchByLogin - fillLogin");
    clientesPage.clickSearch();
    test.log(LogStatus.INFO, "Clientes - searchByLogin - clickSearch");
    clientesPage.checkLiftWithLoginIsDisplayed(login);
    test.log(LogStatus.INFO, "Clientes - searchByLogin -
checkLiftWithLoginIsDisplayed");
    clientesPage.seeDetails(login);
    test.log(LogStatus.INFO, "Clientes - searchByLogin - seeDetails");
    test.log(LogStatus.PASS, "Clientes - searchByLogin", "Login: " + login);
}
```

Tenemos marcado en rojo todos los logs para trazar con el máximo detalle la ejecución del test. Si nos fijamos, los logs intermedios son de clase INFO ya que nos informan de cómo está yendo la ejecución. Una vez que el método se ha completado con éxito, se deja una última traza de tipo PASS.

Todo eso se ve finalmente reflejado en el informe y podemos filtrar las trazas para eliminar información que no nos interese. Las figuras 4.10 y 4.11 son del mismo test con la diferencia que en la segunda solamente mostramos las trazas de tipo PASS.

5. AUTOMATIZACIÓN DE LAS PRUEBAS

En este capítulo vamos a ver el proceso de automatización de los casos de pruebas ya creados. Debido a su similitud, nos limitaremos a explicar un caso de prueba por cada suite. Nos ayudaremos de imágenes, trozos de código y diagramas de secuencia. Para ayudar al entendimiento, es posible que la explicación de algunos de estos casos se simplifique y no coincida totalmente con el código del proyecto.

5.1. Suite de búsqueda

Esta suite se encarga de comprobar que la búsqueda de información dentro de la aplicación web es correcta. Para verificarlo, usaremos diversas herramientas que la propia página nos ofrece para buscar clientes, técnicos o ascensores. El caso concreto será el de buscar los datos de un cliente.

5.1.1. Flujo de navegación del usuario

En este apartado vamos a describir el flujo de navegación del usuario para buscar un cliente en la aplicación. Esto es necesario para poder automatizar el caso de prueba.

1. El usuario ingresará sus datos en la página de inicio de sesión.

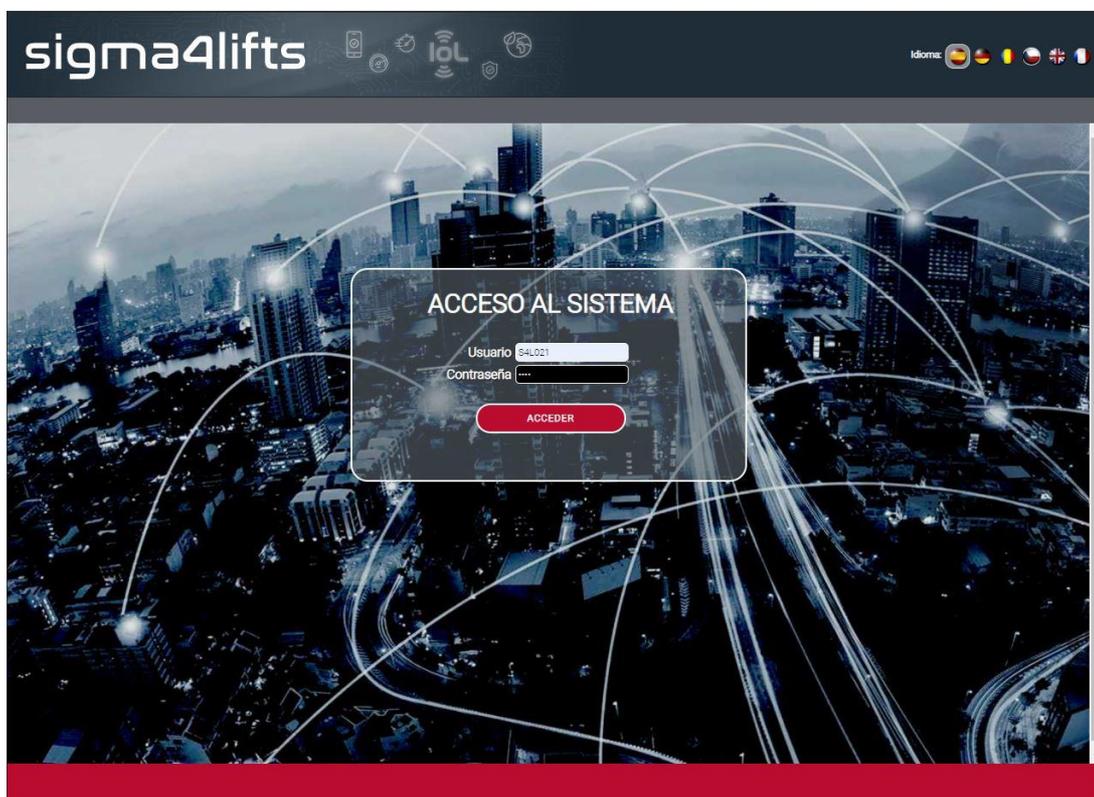


Figura 5.1. Primer paso del flujo de navegación de búsqueda.

2. El usuario hará clic en “Clientes” en el menú lateral izquierdo.

The screenshot shows the Sigma4Lifts Interlift 2015 interface. The left sidebar contains a menu with 'Clientes' highlighted. The main content area displays a table of search results for elevators. The table has columns for 'Ref. Ascensor', 'Descripción', 'Localidad', 'Código Postal', 'Notary', 'N. Teléfono', 'TARIFA SIGMA4LIFTS CONTRATADA', 'ENB1-28', 'Estado', and 'Acceso Remoto'. The table contains 32 rows of data, with the first few rows showing details for elevators in Dos Hermanas and Sevilla.

Ref. Ascensor	Descripción	Localidad	Código Postal	Notary	N. Teléfono	TARIFA SIGMA4LIFTS CONTRATADA	ENB1-28	Estado	Acceso Remoto
00001	Via Serie Torre Navisa - Placa 2	Dos Hermanas	41006		34590 1000226976	MP CONNECT plus	!	✓	✓
00002	Torre Navisa SIM Externa	Sevilla	41006		638606789	MP CONNECT plus	!	✓	✓
00005	Via Serie Torre Navisa - ASC2	Dos Hermanas	41006		34590 1000226976	MP CONNECT plus	!	✓	✓
002015	Iestul-Interlift 2015 - ecoGO	Augsburg	86100		34590 100022533	MP CONNECT ecoGO plus	!	✓	
Ruud_Australia	Australia SIM externa				0061498801140	MP CONNECT plus	!	✓	
Prov - 00001					34590 1000645943	Desactivada	!	✓	
Prov - 00002					24324	Desactivada	!	✓	
Prov-0003 MDC	Provisional:					Desactivada	!	✓	✓
EJMG	Link_ecoGO Placa desarrollo EJMG				34590 1000018424	MP CONNECT ecoGO plus	!	✓	✓
Test_MAY	Sim Manuel-8934076100144523577				34590 1000918031	MP CONNECT ecoGO plus	!	✓	✓
00099	Duplex ecoGO - Montaplatos ASC1				34590 1000908214	MP CONNECT ecoGO plus	!	✓	✓
00070	Duplex ecoGO - Montaplatos ASC2				34590 1000908214	MP CONNECT ecoGO plus	!	✓	
MAY_simplex	provisional- Simplex				34590 1000914361	MP CONNECT ecoGO plus	!	✓	✓
MAY_duplex	provisional- Duplex				34590 1000914361	Desactivada	!	✓	✓
MAY_Triplex	provisional- Triplex				34590 1000914361	Desactivada	!	✓	

Figura 5.2. Segundo paso del flujo de navegación de búsqueda.

3. El usuario hará clic en “Criterios de búsqueda”.

The screenshot shows the Sigma4Lifts Interlift 2015 interface. The left sidebar contains a menu with 'Criterios de búsqueda' highlighted. The main content area displays a table of search criteria for clients. The table has columns for 'Activo', 'LOGIN', 'Nombre', and 'EMAIL'. The table contains 2 rows of data, with the first row showing details for a client named Garcia Nuñez, Antonio.

Activo	LOGIN	Nombre	EMAIL
✓	S4L005	Garcia Nuñez, Antonio	ejmg@mpscensores.com
✓	S4L025	apellido_1_cliente 2021 04 19 18 20 08. nombre_cliente	cliente@gmail.com

Figura 5.3. Tercer paso del flujo de navegación de búsqueda.

4. El usuario insertará el login del cliente en el campo de "Login".

The screenshot shows the Sigma4lifts Interlift 2015 web application interface. The top navigation bar includes the company logo, the date 'Sem23 01/06/2021 14:42', the user's location 'Te encuentras en INICIO >> ADMINISTRACIÓN >> CLIENTES', and a 'CERRAR SESIÓN' button. The left sidebar contains a menu with categories like 'ASCENSORES', 'DASHBOARDING', 'ADMINISTRACIÓN', 'INFORMES', 'DESCARGAS', and 'MI CONFIGURACIÓN'. The main content area features a search form titled 'Criterios de Búsqueda' with fields for 'Nombre', 'Segundo apellido', 'Email', 'Primer apellido', 'Login', and 'Estado'. The 'Login' field is highlighted with a red box and contains the text 'S4L025'. Below the form is a table with columns 'ACTIVO', 'LOGIN', 'NOMBRE', and 'EMAIL'. The table contains two rows of data. At the bottom of the search area, there is a 'BUSCAR' button and a 'LIMPIAR' button. The footer of the page features the 'mp' logo and the website 'mplifts.com'.

ACTIVO	LOGIN	NOMBRE	EMAIL
<input checked="" type="checkbox"/>	S4L005	García Nuñez, Antonio	ejmg@mpascensores.com
<input checked="" type="checkbox"/>	S4L025	apellido_1_cliente 2021.04.19.18.20.08, nombre_cliente	cliente@gmail.com

Figura 5.4. Cuarto paso del flujo de navegación de búsqueda.

5. El usuario hará clic en "Buscar".

This screenshot is identical to the previous one, showing the same search form and results table. The 'BUSCAR' button is now highlighted with a red box, indicating the user's next action. The rest of the interface, including the sidebar, navigation bar, and footer, remains the same.

Figura 5.5. Quinto paso del flujo de navegación de búsqueda.

6. El usuario comprobará que la búsqueda arroja el resultado esperado en la tabla.

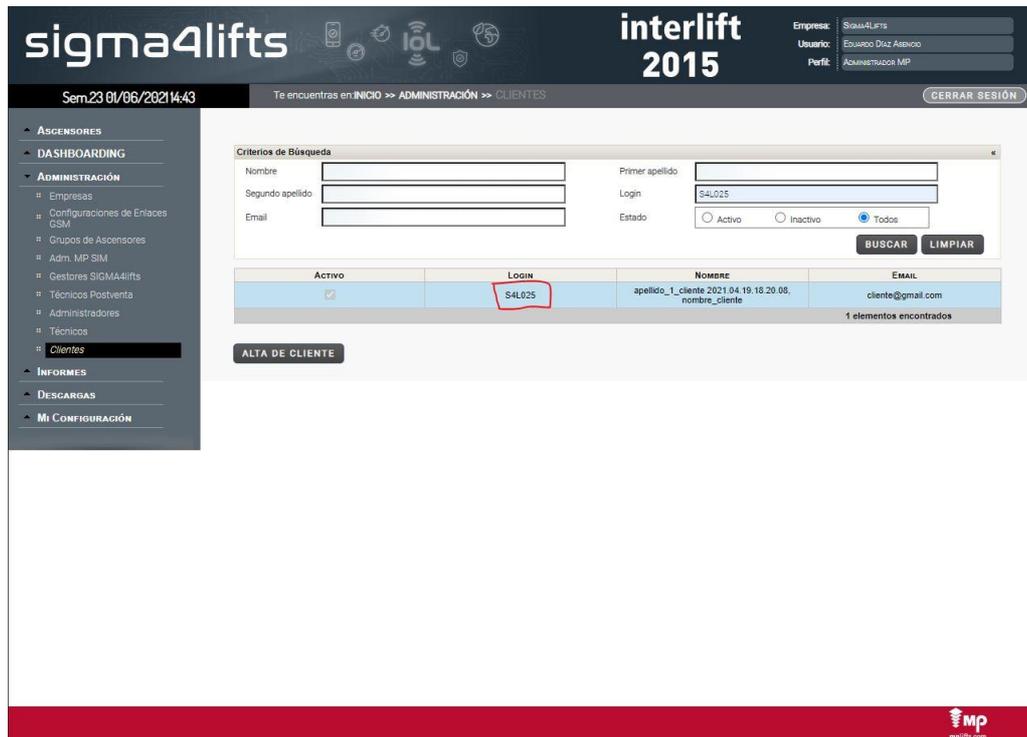


Figura 5.6. Sexto paso del flujo de navegación de búsqueda.

Todo esto podría resumirse en el siguiente diagrama de secuencia:

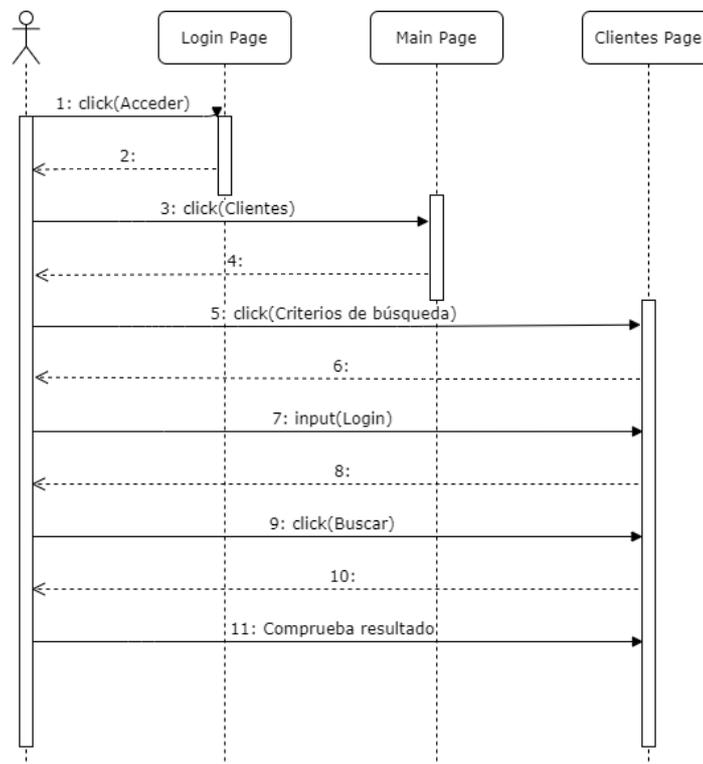


Figura 5.7. Diagrama de secuencia del flujo de navegación de búsqueda.

5.1.2. Flujo de navegación de Selenium

Una vez visto el flujo de navegación manual de un usuario, vamos a hacer la traducción a Selenium. Para ello vamos a crear otro diagrama de secuencia en el que ahora las páginas serán PageObjects y el actor usuario será reemplazado por la capa de Test.

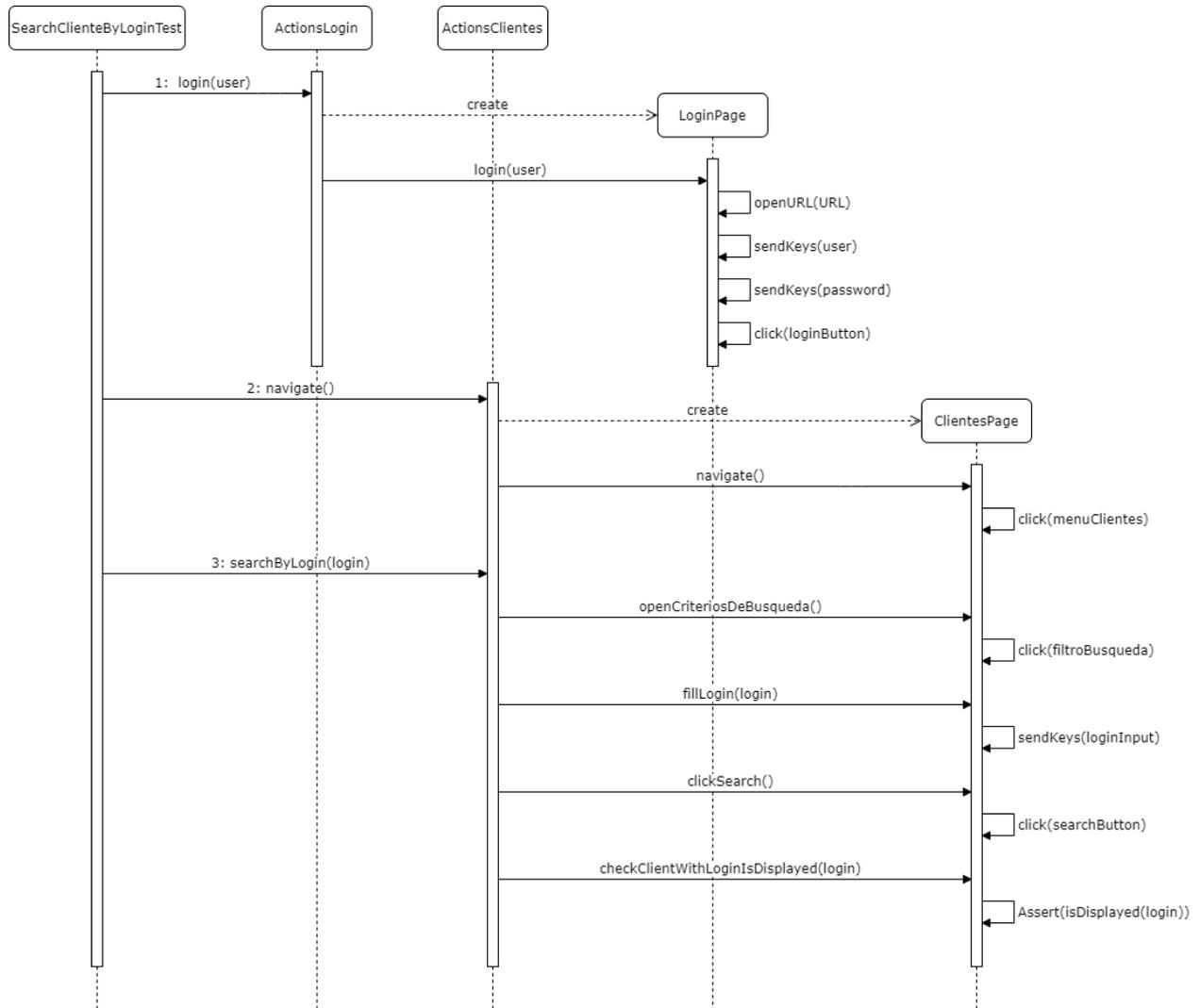


Figura 5.8. Diagrama de secuencia para la automatización de búsqueda.

Si observamos detenidamente la primera columna se trata del fichero java donde crearemos nuestra prueba. Las dos columnas intermedias son la capa Actions, que si recordamos el patrón Test-Actions-Page, hay un objeto de acción por cada PageObject. Con esto el código de la prueba nos quedaría así:

Código 5.1. SearchClienteByLoginTest.java

```

public class SearchClienteByLoginTest extends BaseTest {

    String testDescription = "Search cliente by login.";

    @Test
    
```

```

public void searchClienteByLoginTest() {
    setUpExtentTest(testDescription);
    acciones.loginPage().login(data.getUser());
    acciones.clientesPage().navigate();
    acciones.clientesPage().searchByLogin(data.getCliente().getLogin());
}
}

```

Siguiendo el diagrama de secuencia, la capa Actions debería quedar de la siguiente manera:

Código 5.2. ActionsClientes.java

```

public class ActionsClientes {

    private Browser browser;
    private ExtentTest test;

    public ActionsClientes(Browser browser, ExtentTest test) {
        this.browser = browser;
        this.test = test;
    }

    public void navigate() {
        ClientesPage clientesPage = new ClientesPage(browser);
        clientesPage.navigate();
    }

    public void searchByLogin(String login) {
        ClientesPage clientesPage = new ClientesPage(browser);
        clientesPage.openCriteriosDeBusqueda();
        clientesPage.fillLogin(login);
        clientesPage.clickSearch();
        clientesPage.checkClienteWithLoginIsDisplayed(login);
    }
}

```

Para simplificar el código hemos quitado las trazas de info y pass, pero en un caso real es una buena práctica ponerlos. Por último, veremos cómo tendríamos que crear el PageObject de la página de los clientes:

Código 5.3. ClientesPage.java

```

public class ClientesPage extends BasePage {

    @FindBy(xpath = "//img[@id='leftIconform:MENU MANTENIMIENTO']")
    private WebElement menuAdministracionArrow;
    @FindBy(id = "iconform:MENU MANTENIMIENTO")
    private WebElement menuAdministracion;
    @FindBy(id = "iconform:MENU PROPIETARIOS")
    private WebElement menuClientes;
    @FindBy(xpath = "//a[.='Clientes']")
    private WebElement clientesHeader;
    @FindBy(id = "frmFiltroBusqueda :stpFiltroBusqueda switch on")
    private WebElement filtroBusquedaMarker;
    @FindBy(id = "frmFiltroBusqueda :stpFiltroBusqueda header")
    private WebElement filtroBusquedaHeader;
}

```

```

@FindBy(id = "frmFiltroBusqueda :inLogin")
private WebElement loginInput;
@FindBy(id = "frmFiltroBusqueda :j id74")
private WebElement searchButton;

public ClientesPage(Browser browser) {
    super(browser);
}

public void navigate() {
    if (getAttribute(menuAdministracionArrow,
"src").contains("TriangleUp")) {
        click(menuAdministracion);
    }
    click(menuClientes);
    Assert.assertEquals(isDisplayed(clientesHeader), true);
}

public void openCriteriosDeBusqueda() {
    if (getAttribute(filtroBusquedaMarker, "style").equals("display:
none;")) {
        click(filtroBusquedaHeader);
    }
}

public void fillLogin(String login) {
    sendKeys(loginInput, login);
}

public void clickSearch() {
    click(searchButton);
}

public void checkClienteWithLoginIsDisplayed(String login) {
    Assert.assertTrue(isDisplayed(getElement(By.xpath("//td[.=' " + login
+ "']"))), "Cliente with login " + login + "is displayed");
}
}

```

5.2. Suite de persistencia de datos

Esta suite se encarga de comprobar que la comunicación con la base de datos es correcta y no se pierde la información. Para comprobar esto desde la interfaz web, haremos cambios en campos de ciertos objetos y, tras reiniciar la sesión, comprobamos que siguen ahí. El caso concreto será el de editar los datos de un cliente. Verificaremos que los cambios en su información se mantienen tras reiniciar la sesión.

5.2.1. Flujo de navegación del usuario

En este apartado vamos a describir el flujo de navegación del usuario para editar los datos de un cliente en la aplicación. Esto es necesario para poder automatizar el caso de prueba.

1. El usuario ingresará sus datos en la página de inicio de sesión.

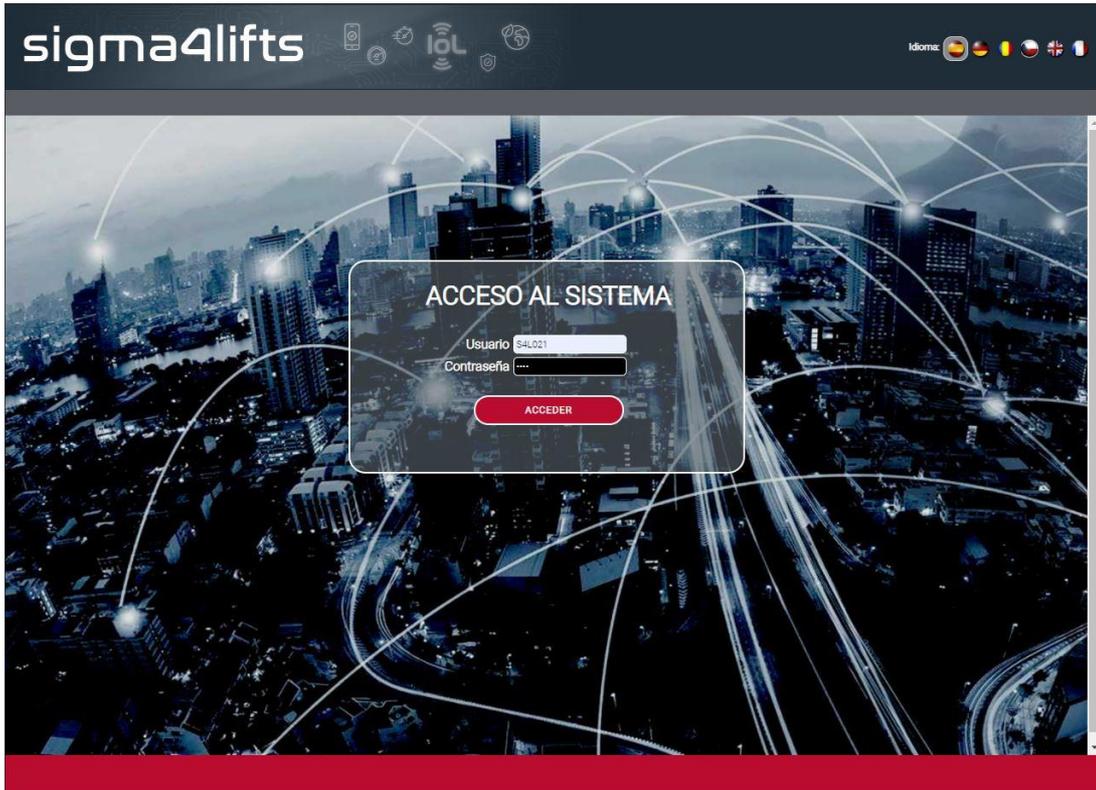


Figura 5.9. Primer paso del flujo de navegación de persistencia.

2. El usuario hará clic en “Clientes” en el menú lateral izquierdo.

Ref. Ascensor	Descripción	Localidad	Código Postal	Notif	N. Teléfono	Tabla SIGMA4LIFTS CONTRATADA	ENB1-2B	Estado	Acceso Remoto
00001	Via Serie Torre Navisa - Placa 2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
00002	Torre Navisa SIM Externa	Sevilla	41006		638606789	MP CONNECT plus	!	✓	🔒
00005	Via Serie Torre Navisa - ASC2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
002015	testu-interlift 2015 - ecoGO	Augsburg	86100		345901000225233	MP CONNECT ecoGO plus	!	✓	🔒
Ruud_Australia	Australia SIM externa				0061498801140	MP CONNECT plus	!	✓	
Prov - 00001					345901000645943	Desactivada	!	✓	
Prov - 00002					24324	Desactivada	!	✓	
Prov-0003 MDC	Provisionalx					Desactivada	!	✓	🔒
EJMG	Link_ecoGO Placa desarrollo EJMG				345901000018424	MP CONNECT ecoGO plus	!	✓	🔒
Test_MAY	Sim Manuel:8934076100144523577				345901000918031	MP CONNECT ecoGO plus	!	✓	🔒
00069	Duplex ecoGO - Montaplatos ASC1				345901000908214	MP CONNECT ecoGO plus	!	✓	🔒
00070	Duplex ecoGO - Montaplatos ASC2				345901000908214	MP CONNECT ecoGO plus	!	✓	🔒
MAY_simplex	provisional- Simplex				345901000914361	MP CONNECT ecoGO plus	!	✓	🔒
MAY_duplex	provisional- Duplex				345901000914361	Desactivada	!	✓	🔒
MAY_Triplex	provisional- Triplex				345901000914361	Desactivada	!	✓	🔒

Figura 5.10. Segundo paso del flujo de navegación de persistencia.

3. El usuario hará clic en “Criterios de búsqueda”.

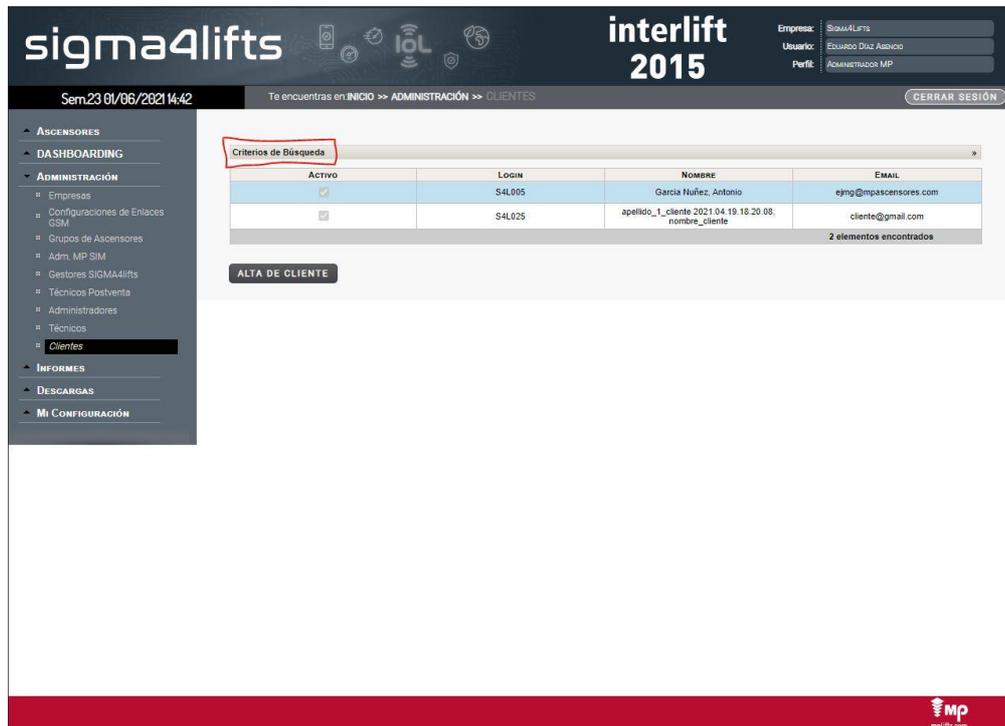


Figura 5.11. Tercer paso del flujo de navegación de persistencia.

4. El usuario insertará el login del cliente en el campo de “Login”.

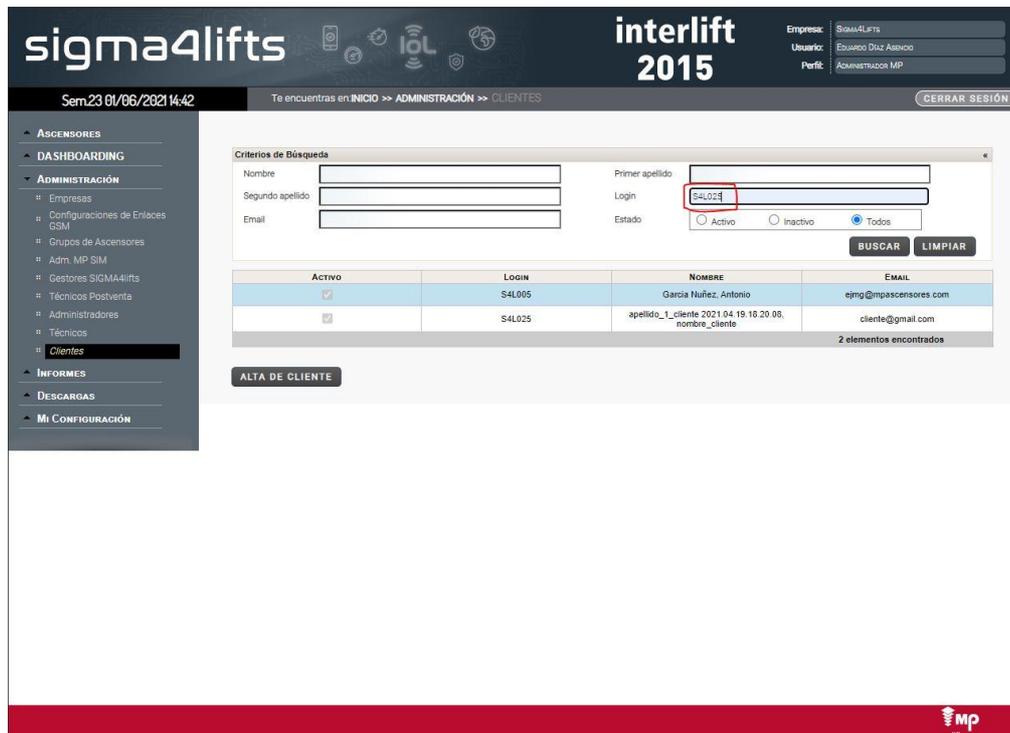


Figura 5.12. Cuarto paso del flujo de navegación de persistencia.

5. El usuario hará clic en “Buscar”.

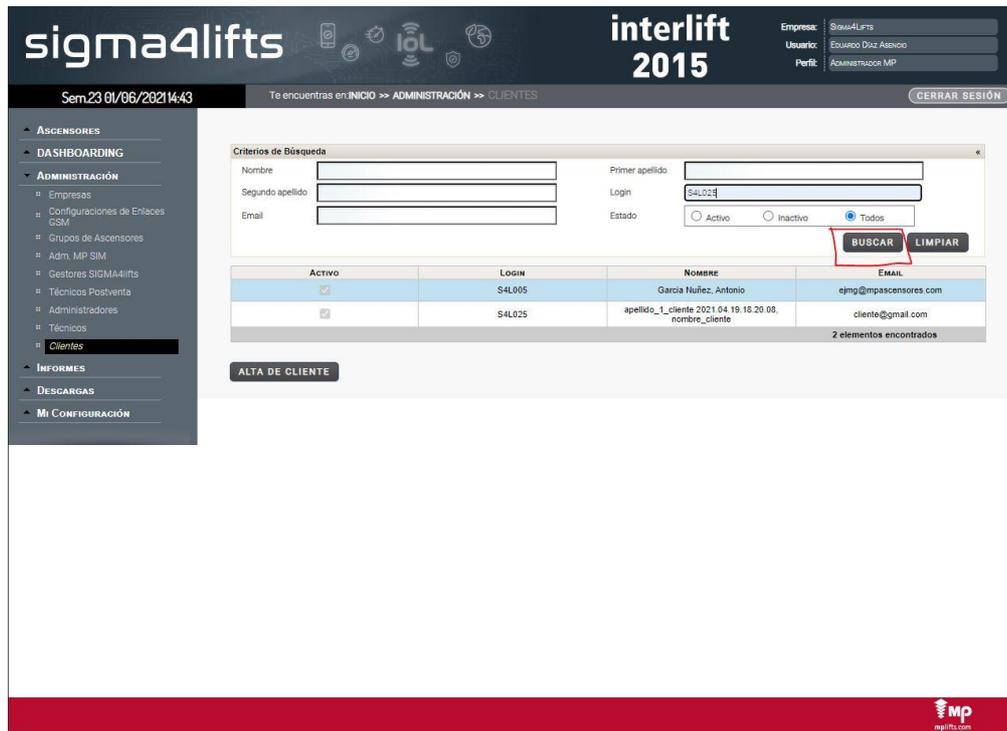


Figura 5.13. Quinto paso del flujo de navegación de persistencia.

6. El usuario comprobará que la búsqueda arroja el resultado esperado en la tabla y hará clic en “Detalles”.

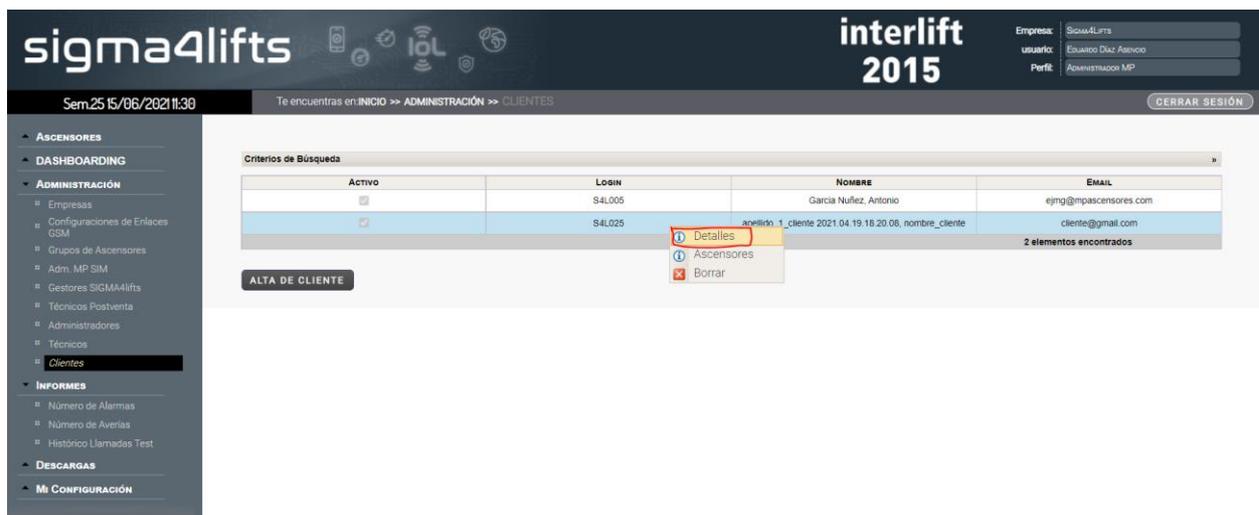


Figura 5.14. Sexto paso del flujo de navegación de persistencia.

7. El usuario hará clic en “Editar”.

The screenshot shows the Sigma4lifts administration interface. The header includes the logo 'sigma4lifts', 'interlift 2015', and user information: Empresa: SIGMA4LIFTS, usuario: EDUARDO DIAZ ASENCIO, Perfil: ADMINISTRADOR MP. The breadcrumb trail is 'Te encuentras en INICIO >> ADMINISTRACIÓN >> CLIENTES'. The left sidebar contains a menu with 'CLIENTES' selected. The main content area is titled 'DATOS DEL USUARIO' and contains a form with the following fields: Empresa (dropdown), Activo (checkbox), Login (text), Email (text), Nombre (text), Zona horaria (dropdown), Idioma de notificación (dropdown), Perfil (dropdown), Cliente (checkbox), Primer apellido (text), and Segundo apellido (text). The 'EDITAR' button is highlighted with a red box.

Figura 5.15. Séptimo paso del flujo de navegación de persistencia.

8. El usuario cambiará el valor del segundo apellido.

The screenshot shows the Sigma4lifts administration interface. The header includes the logo 'sigma4lifts', 'interlift 2015', and user information: Empresa: SIGMA4LIFTS, usuario: EDUARDO DIAZ ASENCIO, Perfil: ADMINISTRADOR MP. The breadcrumb trail is 'Te encuentras en INICIO >> ADMINISTRACIÓN >> CLIENTES'. The left sidebar contains a menu with 'CLIENTES' selected. The main content area is titled 'DATOS DEL USUARIO' and contains a form with the following fields: Empresa (dropdown), Activo (checkbox), Login (text), Email (text), Nombre (text), Zona horaria (dropdown), Idioma de notificación (dropdown), Perfil (dropdown), Cliente (checkbox), Clave (text), Confirmar clave (text), Primer apellido (text), and Segundo apellido (text). The 'SEGUNDO APELLIDO' field is highlighted with a red box.

Figura 5.16. Octavo paso del flujo de navegación de persistencia.

9. El usuario hará clic en “Guardar”.

The screenshot shows the 'DATOS DEL USUARIO' form in the Sigma4lifts administration system. The form includes fields for 'Empresa' (Sigma4lifts), 'Activo' (checked), 'Login' (S4L025), 'Email*' (cliente@gmail.com), 'Nombre*' (nombre_cliente), 'Zona horaria*' (Europe/Madrid), and 'Idioma de notificación*' (Español). There are also fields for 'Perfil' (Cliente), 'Clave', 'Primer apellido*' (apellido_1_cliente), 'Segundo apellido' (2021.04.19.18.20.08), and 'Confirmar clave'. At the bottom, the 'GUARDAR' button is highlighted with a red box, and the 'CANCELAR' button is also visible.

Figura 5.17. Noveno paso del flujo de navegación de persistencia.

10. El usuario cerrará la sesión.

The screenshot shows the search results for 'Criterios de Búsqueda' in the Sigma4lifts administration system. The results table has columns for 'Activo', 'LOGIN', 'NOMBRE', and 'EMAIL'. Two results are shown, with the second one highlighted in blue. The 'CERRAR SESIÓN' button is highlighted with a red box.

Activo	LOGIN	NOMBRE	EMAIL
<input type="checkbox"/>	S4L005	García Nuñez, Antonio	ejmg@mpascensores.com
<input checked="" type="checkbox"/>	S4L025	apellido_1_cliente 2021.04.19.18.20.08, nombre_cliente	cliente@gmail.com

2 elementos encontrados

Figura 5.18. Décimo paso del flujo de navegación de persistencia.

11. El usuario ingresará sus datos en la página de inicio de sesión.

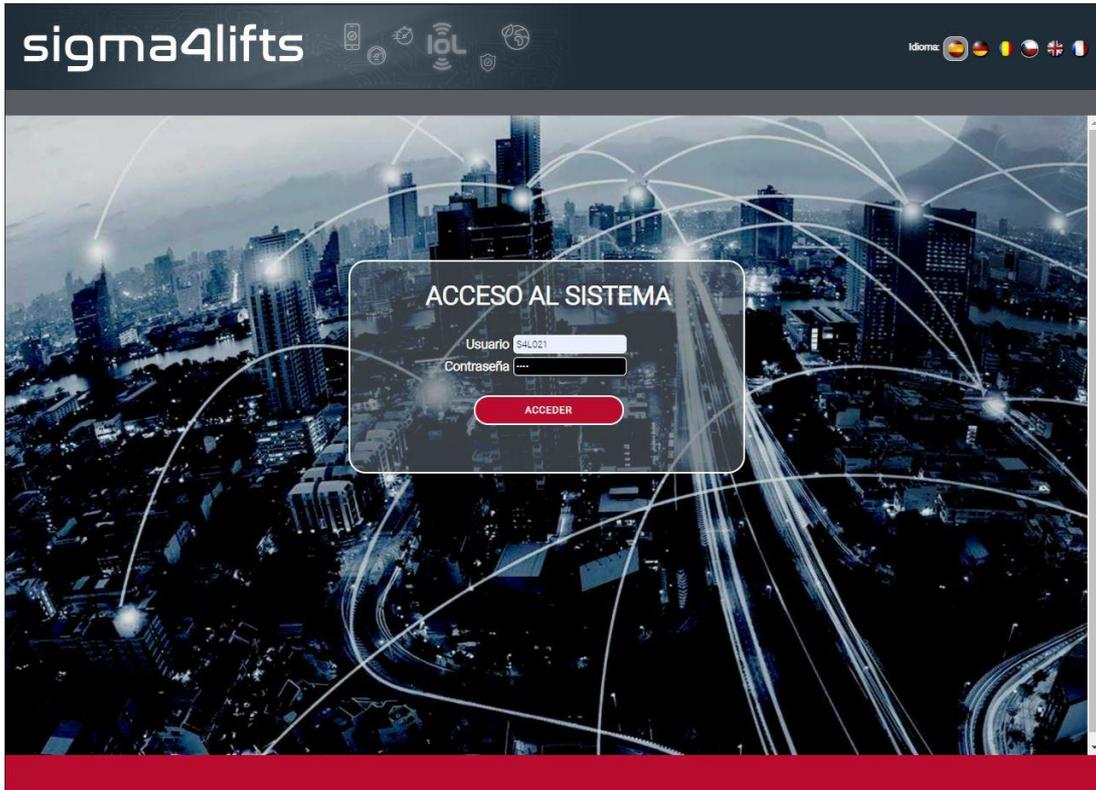


Figura 5.19. Decimoprimer paso del flujo de navegación de persistencia.

12. El usuario hará clic en “Clientes”.

Ref. Ascensor	Descripción	Localidad	Código Postal	Notif	N. Teléfono	Tabla SIGMA4LIFTS CONTRATADA	EN81-28	Estado	Acceso Remoto
00001	Via Serie Torre Navisa - Placa 2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
00002	Torre Navisa SIM Externa	Sevilla	41006		638606789	MP CONNECT plus	!	✓	🔒
00005	Via Serie Torre Navisa - ASC2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
002015	testu-interlift 2015 - ecoGO	Augsburg	86100		345901000225233	MP CONNECT ecoGO plus	!	✓	🔒
Ruud_Australia	Australia SIM externa				0061498801140	MP CONNECT plus	!	✓	
Prov - 00001					345901000645943	Desactivada	!	✓	
Prov - 00002					24324	Desactivada	!	✓	
Prov-0003 MDC	Provisionalx					Desactivada	!	✓	🔒
EJMG	Link_ecoGO Placa desarrollo EJMG				345901000018424	MP CONNECT ecoGO plus	!	✓	🔒
Test_MAY	Sim Manuel-8934076100144523577				345901000918031	MP CONNECT ecoGO plus	!	✓	🔒
00069	Duplex ecoGO - Montaplatos ASC1				345901000908214	MP CONNECT ecoGO plus	!	✓	🔒
00070	Duplex ecoGO - Montaplatos ASC2				345901000908214	MP CONNECT ecoGO plus	!	✓	🔒
MAY_simplex	provisional- Simplex				345901000914361	MP CONNECT ecoGO plus	!	✓	🔒
MAY_duplex	provisional- Duplex				345901000914361	Desactivada	!	✓	🔒
MAY_Triplex	provisional- Triplex				345901000914361	Desactivada	!	✓	🔒

Figura 5.20. Decimosegundo paso del flujo de navegación de persistencia.

13. El usuario hará clic en “Criterios de búsqueda”.

The screenshot shows the user interface of the sigma4lifts interlift 2015 application. The top navigation bar includes the logo, the text "interlift 2015", and user information: Empresa: SIGMA4LIFTS, Usuario: Eduardo Diaz Asencio, Perfil: ADMINISTRADOR MP. The breadcrumb trail reads "Te encuentras en INICIO >> ADMINISTRACIÓN >> CLIENTES". A "CERRAR SESIÓN" button is visible in the top right. On the left, a sidebar menu lists various sections: ASCENSORES, DASHBOARDING, ADMINISTRACIÓN (with sub-items like Empresas, Configuraciones de Enlaces GSM, Grupos de Ascensores, Adm. MP SIM, Gestores SIGMA4lifts, Técnicos Postventa, Administradores, Técnicos, and Clientes), INFORMES, DESCARGAS, and MI CONFIGURACIÓN. The main content area features a "Criterios de Búsqueda" dropdown menu, which is currently open, showing a table of search results. The table has columns for "Activo", "LOGIN", "NOMBRE", and "EMAIL". Two rows are visible, with the first row highlighted in blue. Below the table is a button labeled "ALTA DE CLIENTE".

Activo	LOGIN	NOMBRE	EMAIL
<input checked="" type="checkbox"/>	S4L005	García Nuñez, Antonio	ejmg@mpascensores.com
<input checked="" type="checkbox"/>	S4L025	apellido_1_cliente 2021 04 19 16 20 08, nombre_cliente	cliente@gmail.com

Figura 5.21. Decimotercer paso del flujo de navegación de persistencia.

14. El usuario insertará el login del cliente en el campo de “Login”.

This screenshot shows the same application interface as Figure 5.21, but with the search criteria form expanded. The form includes input fields for "Nombre", "Primer apellido", "Segundo apellido", "Login", and "Email". The "Login" field is filled with the value "S4L025" and is highlighted with a red box. Below the input fields are radio buttons for "Estado": "Activo", "Inactivo", and "Todos" (selected). "BUSCAR" and "LIMPIAR" buttons are located to the right of the form. The table below the form shows the same two search results as in Figure 5.21. The "ALTA DE CLIENTE" button is also present.

Figura 5.22. Decimocuarto paso del flujo de navegación de persistencia.

15. El usuario hará clic en “Buscar”.

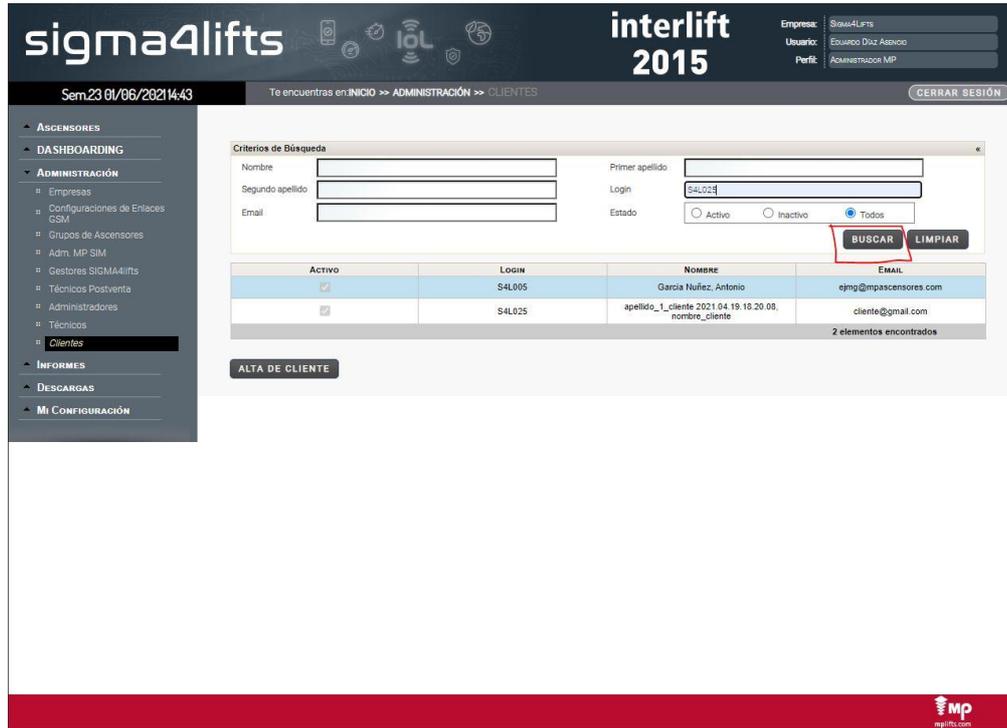


Figura 5.23. Decimoquinto paso del flujo de navegación de persistencia.

16. El usuario comprobará que la búsqueda arroja el resultado esperado en la tabla y hará clic en “Detalles”.

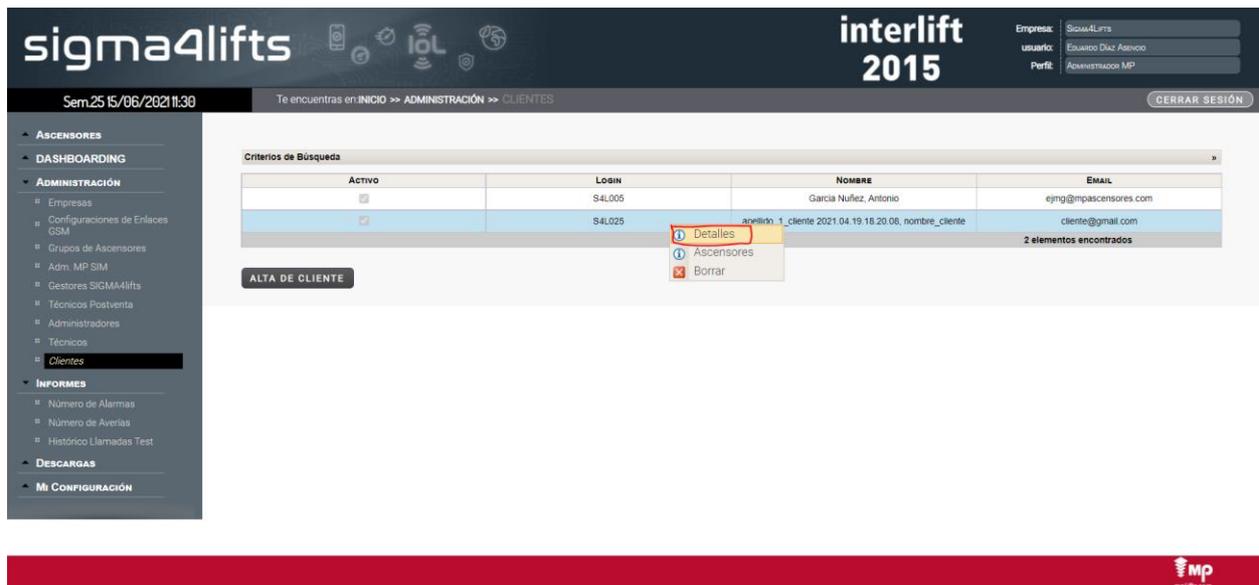


Figura 5.24. Decimosexto paso del flujo de navegación de persistencia.

17. El usuario comprobará que el valor del segundo apellido es el introducido en el paso 8.

The screenshot shows the 'interlift 2015' web application interface. At the top, there is a navigation bar with the company logo 'sigma4lifts' and 'interlift 2015'. The user is logged in as 'Eduardo Diaz Asuncio' with the profile 'Administrador MP'. The page title is 'DATOS DEL USUARIO'. The form contains the following fields:

Field	Value
Empresa*	Sigma4lifts
Activo	<input checked="" type="checkbox"/>
Login	S4L025
Email*	cliente@gmail.com
Nombre*	nombre_cliente
Zona horaria*	Europe/Madrid
Idioma de notificación*	Español
Perfil	Cliente
Primer apellido*	apellido_1_cliente
Segundo apellido	2021.04.19.18.20.08

Buttons: EDITAR, CANCELAR

Figura 5.25. Decimoséptimo paso del flujo de navegación de persistencia.

Como hemos hecho en el caso anterior, vamos a condensar todos los pasos en un diagrama de secuencia más simplificado para ayudarnos a diseñar el código del proyecto:

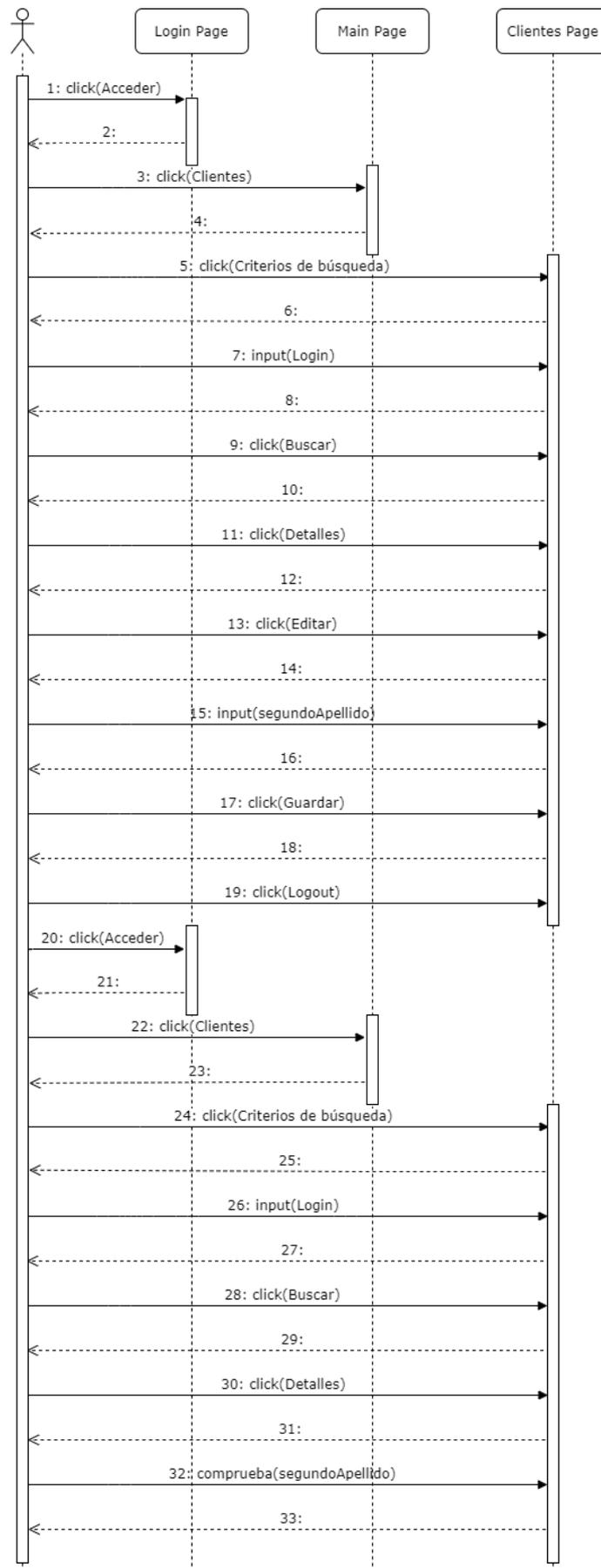


Figura 5.26. Diagrama de secuencia del flujo de navegación de persistencia.

5.2.2. Flujo de navegación de Selenium

Una vez visto el flujo de navegación manual de un usuario, vamos a pasarlo a software. Para ello vamos a crear otro diagrama de secuencia en el que ahora las páginas serán PageObjects y el actor usuario será reemplazado por la capa de Test.

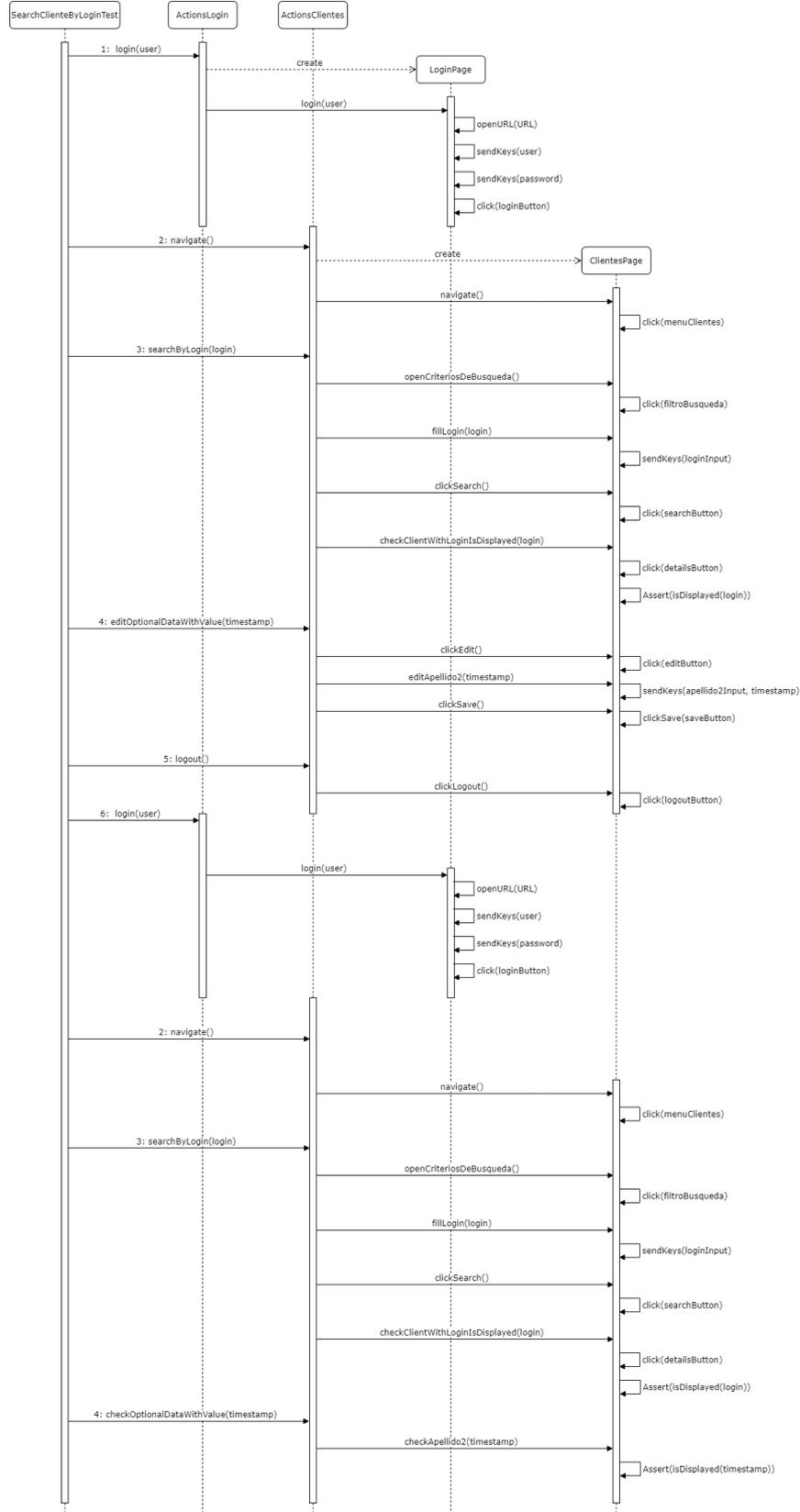


Figura 5.27. Diagrama de secuencia para la automatización de búsqueda.

Como resultado, el código del test sería el siguiente:

Código 5.4. EditClienteDataTest.java

```
public class EditClienteDataTest extends BaseTest {

    String testDescription = "Edit cliente data test.";

    @Test
    public void editClienteDataTest() {
        setUpExtentTest(testDescription);
        acciones.loginPage().login(data.getUser());
        acciones.clientesPage().navigate();
        acciones.clientesPage().searchByLogin(cliente.getLogin());
        String timestamp = new SimpleDateFormat("yyy.MM.dd.HH.mm.ss");
        acciones.clientesPage().editOptionalDataWithValue(timestamp);
        acciones.logout();
        acciones.loginPage().login(data.getUser());
        acciones.clientesPage().navigate();
        acciones.clientesPage().searchByLogin(cliente.getLogin());
        acciones.clientesPage().checkOptionalDataWithValue(timestamp);
        acciones.logout();
    }
}
```

En las capas Actions-Pages tendríamos los métodos descritos en el diagrama de secuencia de la figura 5.27 siguiendo el mismo patrón que lo visto en el punto 5.1.2.

5.3. Suite de casos de uso

En esta suite cubriremos los casos de uso más importantes de la aplicación web, tales como añadir averías a un ascensor, añadir alarmas, suscripciones entre usuarios y ascensores, etc. En concreto analizaremos el caso de prueba de añadir una alarma a un ascensor.

5.3.1. Flujo de navegación del usuario

En este apartado vamos a describir el flujo de navegación del usuario para añadir una alarma a un ascensor de forma manual. Esto es necesario para poder automatizar el caso de prueba.

1. El usuario ingresará sus datos en la página de inicio de sesión.



Figura 5.28. Primer paso del flujo de navegación de casos de uso.

2. El usuario navegará a la página de “Relación de ascensores” en el menú lateral izquierdo.

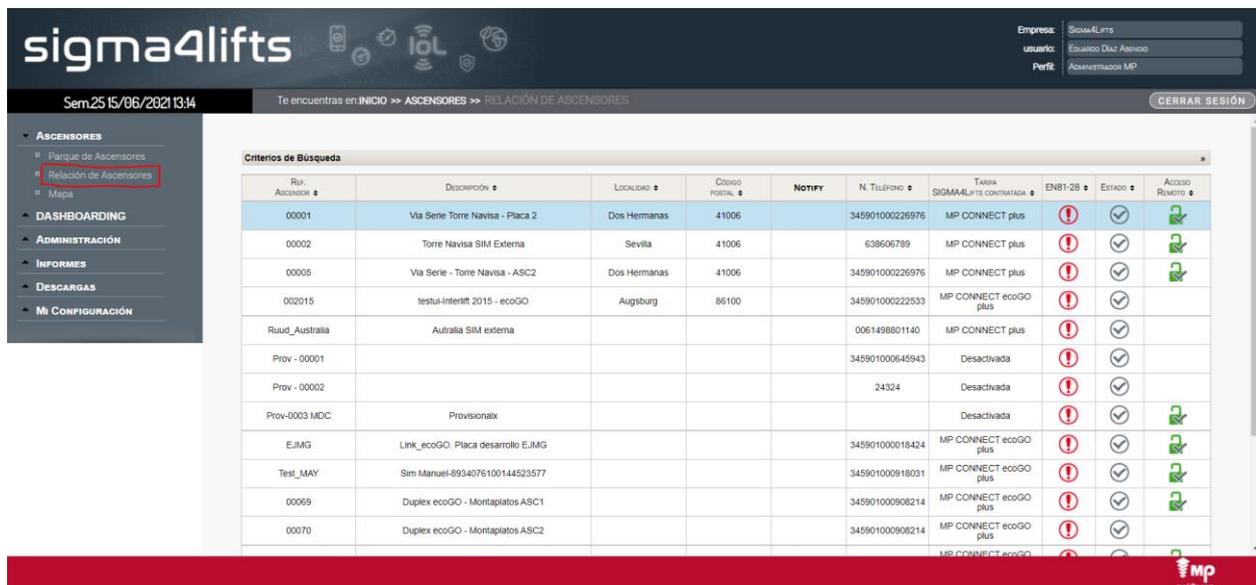


Figura 5.29. Segundo paso del flujo de navegación de casos de uso.

3. El usuario abrirá los criterios de búsqueda.

The screenshot shows the 'sigma4lifts interlift 2015' interface. The user is logged in as 'Eduardo Díaz Astengo' (Administrador MP). The breadcrumb trail is 'INICIO >> ASCENSORES >> RELACION DE ASCENSORES'. The left sidebar contains navigation options like 'ASCENSORES', 'DASHBOARDING', 'ADMINISTRACIÓN', etc. The main content area displays a table with search criteria and a list of elevators. The 'Criterios de Búsqueda' section is highlighted with a red box.

REF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	NOTIFY	N. TELÉFONO	TARIFA SIGMA4LIFTS CONTRATADA	EN81-28	ESTADO	ACCESO REMOTO
00001	Via Serie Torre Navisa - Placa 2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
00002	Torre Navisa SIM Externa	Sevilla	41006		638606789	MP CONNECT plus	!	✓	🔒
00005	Via Serie - Torre Navisa - ASC2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
002015	testui-interlift 2015 - ecoGO	Augsburg	86100		345901000222533	MP CONNECT ecoGO plus	!	✓	🔒
Ruud_Australia	Australia SIM externa				0061498801140	MP CONNECT plus	!	✓	🔒
Prov - 00001					345901000645943	Desactivada	!	✓	
Prov - 00002					24324	Desactivada	!	✓	
Prov-0003 MDC	Provisionalx					Desactivada	!	✓	🔒
EJMG	Link ecoGO. Placa desarrollo EJMG				345901000018424	MP CONNECT ecoGO plus	!	✓	🔒
Test_MAY	Sim Manuel-6934076100144523577				345901000918031	MP CONNECT ecoGO plus	!	✓	🔒
00069	Duplex ecoGO - Montapiatos ASC1				345901000908214	MP CONNECT ecoGO plus	!	✓	🔒
00070	Duplex ecoGO - Montapiatos ASC2				345901000908214	MP CONNECT ecoGO plus	!	✓	🔒

Figura 5.30. Tercer paso del flujo de navegación de casos de uso.

4. El usuario introducirá la referencia del ascensor.

The screenshot shows the 'sigma4lifts interlift 2015' interface. The user is logged in as 'Eduardo Díaz Astengo' (Administrador MP). The breadcrumb trail is 'INICIO >> ASCENSORES >> RELACION DE ASCENSORES'. The left sidebar contains navigation options like 'ASCENSORES', 'DASHBOARDING', 'ADMINISTRACIÓN', etc. The main content area displays a search form and a table of elevators. The 'Criterios de Búsqueda' section is highlighted with a red box.

The search form includes fields for: Grupo, Ref. Ascensor (highlighted with a red box), Descripción, Localidad, Código postal, N. Teléfono, Tarifa SIGMA4LifTS contratada, Estado, Acceso Remoto, ICIDID, Modelo Maniobra, and Región SIM. There are also date range fields (Fecha Alta: Desde/Hasta) and a 'Baja' checkbox. 'BUSCAR' and 'LIMPIAR' buttons are present.

REF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	NOTIFY	N. TELÉFONO	TARIFA SIGMA4LIFTS CONTRATADA	EN81-28	ESTADO	ACCESO REMOTO
00001	Via Serie Torre Navisa - Placa 2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
00002	Torre Navisa SIM Externa	Sevilla	41006		638606789	MP CONNECT plus	!	✓	🔒
00005	Via Serie - Torre Navisa - ASC2	Dos Hermanas	41006		345901000226976	MP CONNECT plus	!	✓	🔒
002015	testui-interlift 2015 - ecoGO	Augsburg	86100		345901000222533	MP CONNECT ecoGO plus	!	✓	🔒
Ruud_Australia	Australia SIM externa				0061498801140	MP CONNECT plus	!	✓	🔒
Prov - 00001					345901000645943	Desactivada	!	✓	
Prov - 00002					24324	Desactivada	!	✓	
Prov-0003 MDC	Provisionalx					Desactivada	!	✓	🔒

Figura 5.31. Cuarto paso del flujo de navegación de casos de uso.

5. El usuario hará clic en “Buscar”.

The screenshot shows the Sigma4lifts web application interface. At the top, there is a navigation bar with the logo 'sigma4lifts' and 'interlift 2015'. The user is logged in as 'Eduardo Diaz Asencio' from 'Administrador MP'. The main content area displays a search criteria form with fields for 'Grupo', 'Ref. Ascensor', 'N. Teléfono', 'ICCID', 'Fecha Alta: Desde', 'Fecha Alta: Hasta', 'Descripción', 'Localidad', 'Código postal', 'Tarifa SIGMA4lifts contratada', 'Estado', 'Acceso Remoto', 'Modelo Maniobra', 'Región SIM', and 'SIM'. A red box highlights the 'BUSCAR' button. Below the form is a table with columns: 'Rif. Ascensor', 'Descripción', 'LOCALIDAD', 'Código POSTAL', 'Notify', 'N. Teléfono', 'TARIFA SIGMA4LIFTS CONTRATADA', 'ENB1-28', 'Estado', and 'Acceso Remoto'. The table contains several rows of data, including 'Via Serie Torre Navisa - Placa 2', 'Torre Navisa SIM Externa', and 'Via Serie - Torre Navisa - ASC2'.

Figura 5.32. Quinto paso del flujo de navegación de casos de uso.

6. El usuario hará clic en “Añadir evento manual” en el resultado ofrecido.

The screenshot shows the Sigma4lifts web application interface. The search criteria form is the same as in Figure 5.32. The search results table now shows a single result: 'Torre Navisa SIM Externa' with 'Sevilla' as the location and '41006' as the postal code. A context menu is open over this result, listing options: 'Datos Identificativos', 'Preventive', 'Status - Remote Control', 'Traffic Control', and 'Añadir evento manual'. The 'Añadir evento manual' option is highlighted with a red box. The table also shows a 'Prov - 0002' row with 'Desactivada' status. The bottom of the page features a red bar with the 'MP' logo.

Figura 5.33. Sexto paso del flujo de navegación de casos de uso.

7. El usuario seleccionará “Alarma” como tipo de mensaje.

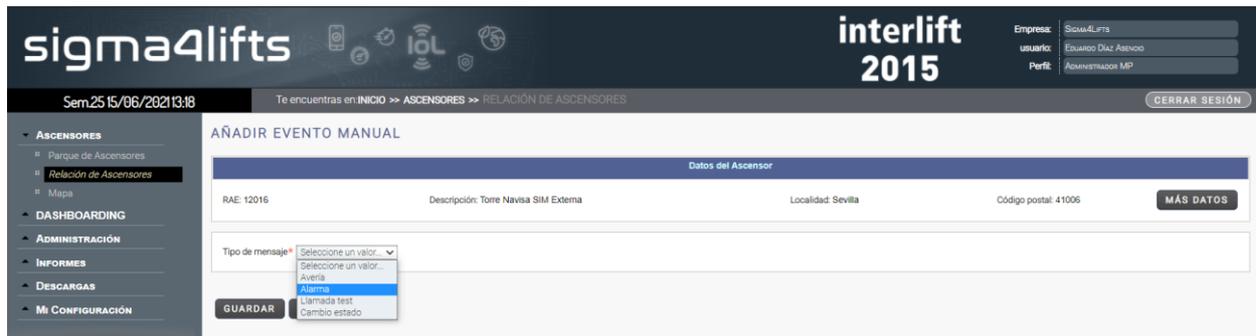


Figura 5.34. Séptimo paso del flujo de navegación de casos de uso.

8. El usuario pondrá la fecha de “Hoy” en la fecha del evento.

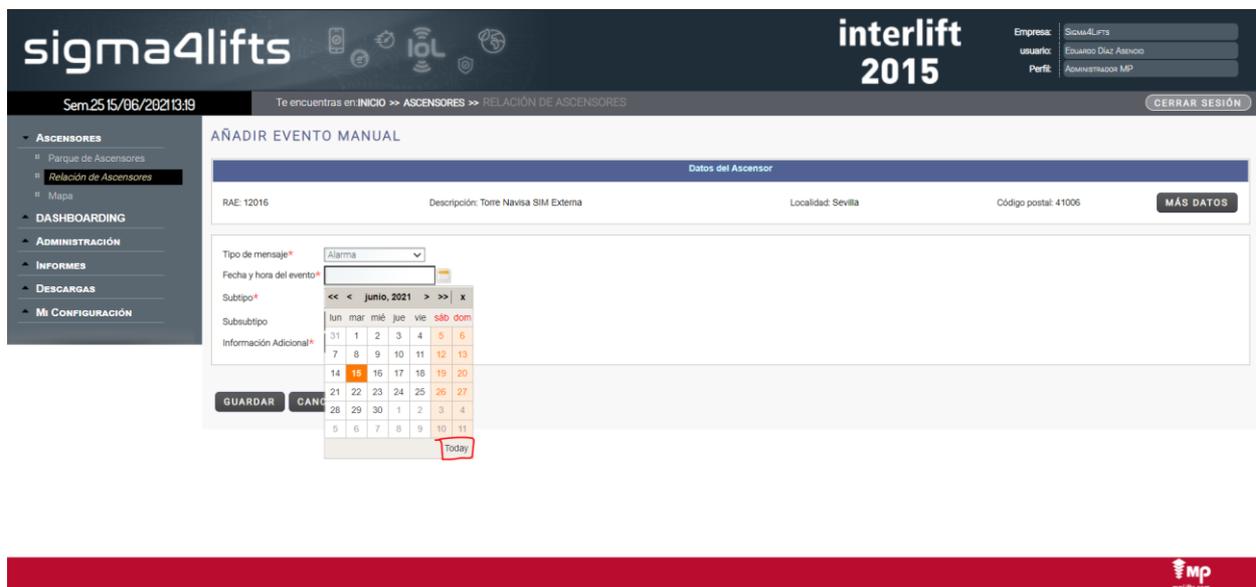


Figura 5.35. Octavo paso del flujo de navegación de casos de uso.

9. El usuario seleccionará “Alarma” como subtipo de alarma.

The screenshot shows the 'Añadir Evento Manual' form in the Sigma4lifts web application. The form is titled 'Añadir Evento Manual' and is located under the 'RELACION DE ASCENSORES' section. The form contains the following fields:

- RAE: 12016
- Descripción: Torre Navisa SIM Externa
- Localidad: Sevilla
- Código postal: 41006
- MÁS DATOS (button)
- Tipo de mensaje*: Alarma (dropdown)
- Fecha y hora del evento*: 15/06/2021 12:00
- Subtipo*: Alarma Alarma de técnico
- Subsubtipo: 0
- Información Adicional*: (empty text field)

Buttons: GUARDAR, CANCELAR

Figura 5.36. Noveno paso del flujo de navegación de casos de uso.

10. El usuario introducirá la información adicional.

The screenshot shows the 'Añadir Evento Manual' form in the Sigma4lifts web application. The form is titled 'Añadir Evento Manual' and is located under the 'RELACION DE ASCENSORES' section. The form contains the following fields:

- RAE: 12016
- Descripción: Torre Navisa SIM Externa
- Localidad: Sevilla
- Código postal: 41006
- MÁS DATOS (button)
- Tipo de mensaje*: Alarma (dropdown)
- Fecha y hora del evento*: 15/06/2021 12:00
- Subtipo*: Alarma Alarma de técnico
- Subsubtipo: 0
- Información Adicional*: No

Buttons: GUARDAR, CANCELAR

Figura 5.37. Décimo paso del flujo de navegación de casos de uso.

11. El usuario hará clic en “Guardar”.

The screenshot shows the 'Añadir Evento Manual' form in the Sigma4lifts application. The form includes fields for 'RAE: 12016', 'Descripción: Torre Navisa SIM Externa', 'Localidad: Sevilla', and 'Código postal: 41006'. Below these are fields for 'Tipo de mensaje*' (Alarma), 'Fecha y hora del evento*' (15/06/2021 12:00), 'Subtipo*' (Alarma selected), 'Subsubtipo' (0), and 'Información Adicional*' (No). At the bottom, the 'GUARDAR' button is highlighted with a red box, and a 'CANCELAR' button is also visible.

Figura 5.38. Decimoprimer paso del flujo de navegación de casos de uso.

12. El usuario navegará a la página de “En curso” en el menú lateral izquierdo.

The screenshot shows the 'En curso' page in the Sigma4lifts application. The 'En Curso' menu item in the left sidebar is highlighted with a red box. The main content area displays a table with search criteria and a list of items.

Criterios de Búsqueda									
RUF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	NOTIFY	N. TELÉFONO	TARIFA SIGMA4LIFTS CONTRATADA	ENB1-28	ENTRADO	ACCESO REMOTO
00002	Torre Navisa SIM Externa	Sevilla	41006		638606789	MP CONNECT plus	!	✓	✓
Prov - 00002					24324	Desactivada	!	✓	

2 elementos encontrados

ALTA DE ASCENSOR

Figura 5.39. Decimosegundo paso del flujo de navegación de casos de uso.

13. El usuario seleccionará el grupo de ascensores.

The screenshot shows the Sigma4lifts Interlift 2015 dashboard. The left sidebar contains navigation options like 'ASCENSORES', 'DASHBOARDING', 'ESTADÍSTICAS', 'ADMINISTRACIÓN', 'INFORMES', 'DESCARGAS', and 'MI CONFIGURACIÓN'. The main area displays a table titled 'AVERÍAS EN CURSO' with columns: REF ASCENSOR, DESCRIPCIÓN, LOCALIDAD, CÓDIGO POSTAL, FECHA Y HORA DE ALTA, FECHA Y HORA DE CIERRE, and CÓDIGO ERROR. A dropdown menu is open over the 'Grupo' column, showing options like 'showroom_SSP', 'MAY_simplex', 'Pinc DT_MAY3', 'SF_Calonge_01', '00001', '00069', '00070', 'MAY_duplex', 'EJMG', and 'Test_MAY'. The table contains 11 rows of data. At the bottom, there is a section for 'ALARMAS EN CURSO'.

Figura 5.40. Decimotercer paso del flujo de navegación de casos de uso.

14. El usuario introducirá fecha de cierre a la alarma creada previamente.

The screenshot shows the Sigma4lifts Interlift 2015 dashboard. The left sidebar is the same as in Figure 5.40. The main area displays a table titled 'ALARMAS EN CURSO' with columns: REF ASCENSOR, DESCRIPCIÓN, LOCALIDAD, CÓDIGO POSTAL, FECHA Y HORA DE ALTA, FECHA Y HORA DE CIERRE, and CLASE ALARMA. A dropdown menu is open over the 'Grupo' column, showing options like 'Torre Navisa S...', 'Introducir Fecha de Cierre', 'Cerrar incidencias activas', 'Archivar incidencias cerradas', 'Datos del Ascensor', 'Preventive', and 'Status - Remote Control'. The table contains 2 rows of data. Below the table, there is a section for 'S DE LLAMADAS DE TEST EN CURSO' with a table containing 1 row of data. At the bottom, there is a 'GUARDAR COMO PDF' button.

Figura 5.41. Decimocuarto paso del flujo de navegación de casos de uso.

15. El usuario archivará la incidencia.

The screenshot shows the 'interlift 2015' dashboard. The left sidebar contains navigation options: ASCENSORES, DASHBOARDING, ESTADÍSTICAS, ADMINISTRACIÓN, INFORMES, DESCARGAS, and MI CONFIGURACIÓN. The main area displays a table of incidents under the 'AVERÍAS EN CURSO' section. A dropdown menu is open over the 'Archivar Incidencia' option, listing actions like 'Introducir Fecha de Cierre', 'Cerrar incidencias activas', 'Archivar Incidencia', 'Archivar incidencias cerradas', 'Datos del Ascensor', 'Preventive', and 'Status - Remote Control'. The 'Archivar Incidencia' option is highlighted with a red box.

REF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	FECHA Y HORA DE ALTA	FECHA Y HORA DE CIERRE	CÓDIGO ERROR
Test_MAY	Sim Manuel- 8934076100144523577			25/09/2017 11:42		06 04 00: Contactor pegado
002015	testui-interlift 2015 - ecoGO	Augsburg	86100	16/11/2015 17:08		02 07 00: Circuito de seguridad de cerrojos abierto, durante un movimiento

Figura 5.42. Decimoquinto paso del flujo de navegación de casos de uso.

16. El usuario introducirá un valor y confirmará el archivo de la incidencia.

The screenshot shows the 'interlift 2015' dashboard with a confirmation dialog box open. The dialog is titled 'Archivar Incidencia' and contains a yellow warning icon, the text 'Archivar Incidencia', and an 'Observaciones:' field with the value 'incidencia'. At the bottom of the dialog are two buttons: 'ARCHIVAR INCIDENCIA' (highlighted with a red box) and 'CANCELAR'. The background shows the same incident table as in Figure 5.42.

Figura 5.43. Decimosexto paso del flujo de navegación de casos de uso.

17. El usuario navegará a la página de “Registros” en el menú lateral izquierdo.

sigma4lifts interlift 2015

Sem25/06/2021 13:45 Te encuentras en INICIO >> DASHBOARDING >> EN CURSO CERRAR SESIÓN

Empres: Sigma4Lifts
usuario: Eduardo Díaz Azevedo
Perfil: Administrador MP

Grupo: NAVISA

AVERÍAS EN CURSO						
REF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	FECHA Y HORA DE ALTA	FECHA Y HORA DE CIERRE	CÓDIGO ERROR
Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 11:42		06 04 00: Contactor pegado
002015	testul-Interlift 2015 - ecoGO	Augsburg	86100	16/11/2015 17:08		02 07 00: Circuito de seguridad de cerrojos abierto, durante un movimiento

2 elementos encontrados

ALARMAS EN CURSO						
REF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	FECHA Y HORA DE ALTA	FECHA Y HORA DE CIERRE	CLASE ALARMA
0 elementos encontrados						

ERRORES DE LLAMADAS DE TEST EN CURSO						
REF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	FECHA Y HORA DE ALTA	FECHA Y HORA DE CIERRE	MODELO ENLACE / TELEFONO
002015	testul-Interlift 2015 - ecoGO	Augsburg	86100	16/12/2018 13:00		LINK/345901000222533

1 elementos encontrados

GUARDAR COMO PDF

Figura 5.44. Decimoséptimo paso del flujo de navegación de casos de uso.

18. El usuario comprobará que el último registro es la incidencia archivada.

sigma4lifts interlift 2015

Sem25/06/2021 13:46 Te encuentras en INICIO >> DASHBOARDING >> REGISTROS CERRAR SESIÓN

Empres: Sigma4Lifts
usuario: Eduardo Díaz Azevedo
Perfil: Administrador MP

Grupo: NAVISA

AVERÍAS - ALARMAS - ERRORES DE LLAMADAS DE TEST							
TIPO DE MENSAJE	REF. ASCENSOR	DESCRIPCIÓN	LOCALIDAD	CÓDIGO POSTAL	FECHA Y HORA DE ALTA	FECHA Y HORA DE CIERRE	INFORMACIÓN ADICIONAL
Alarma	00002	Torre Navisa SIM Externa	Sevilla	41006	15/06/2021 13:38	15/06/2021 13:44	10 Alarma [S4.021 timestamp]
Llamada test	002015	testul-Interlift 2015 - ecoGO	Augsburg	86100	16/12/2018 13:00		Cross-345901000222533 [SIGMA4]
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 11:42		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 09:49		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 18:11		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 15:35		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 14:53		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 13:23		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 13:19		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 12:58		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 12:54		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 12:52		06 04 00: Contactor pegado
Avería	Test_MAY	Sim Manuel-8934076100144523577			25/09/2017 12:49		06 04 00: Contactor pegado

Figura 5.45. Decimoctavo paso del flujo de navegación de casos de uso.

Todo esto podría resumirse en el siguiente diagrama de secuencia:

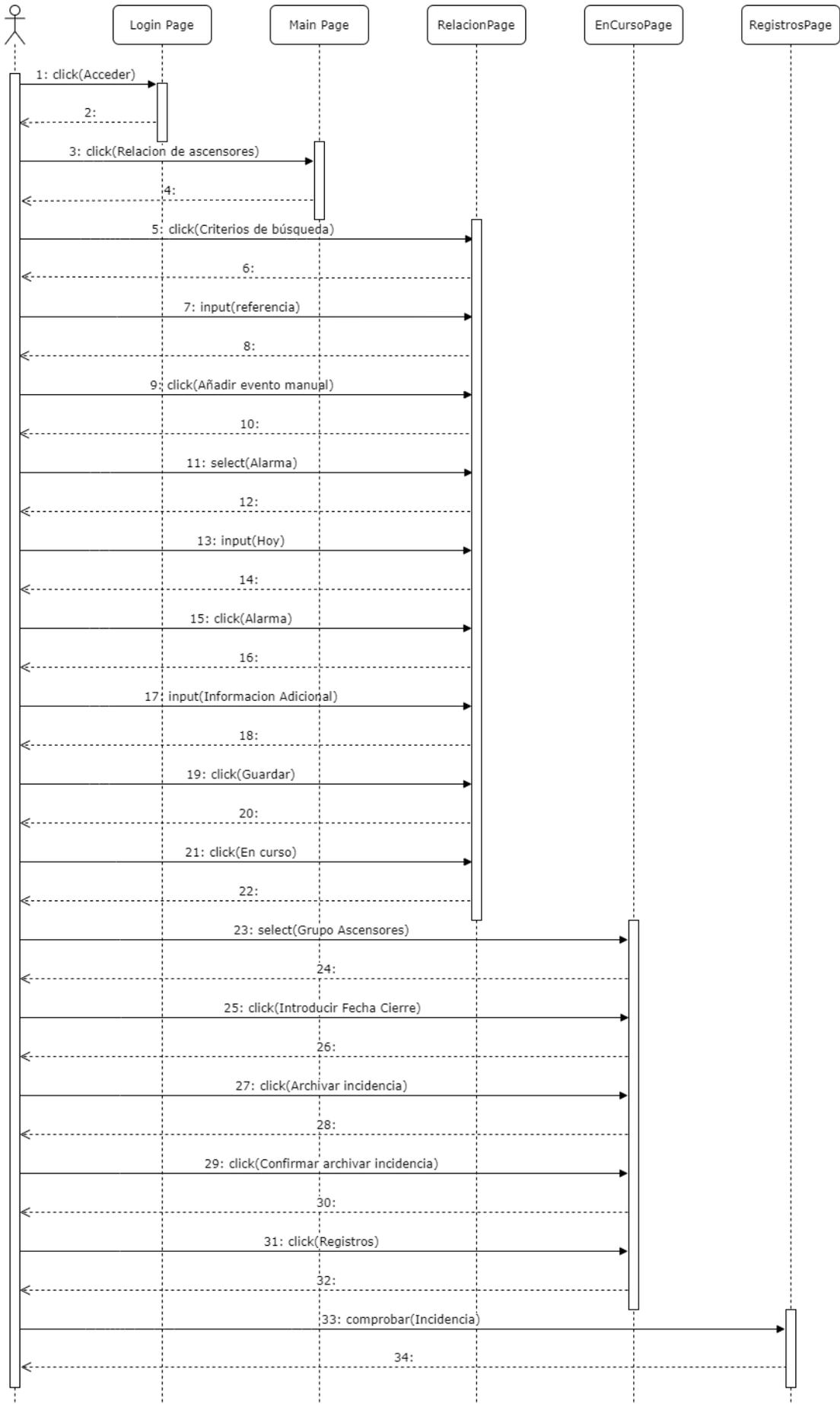


Figura 5.46. Diagrama de secuencia para la automatización de caso de uso.

5.3.2. Flujo de navegación de Selenium

Siguiendo los casos anteriores, vamos a ver el diagrama de secuencia equivalente en nuestro proyecto de automatización. En este caso omitiremos la parte del inicio de sesión para reducir el tamaño del diagrama. Además, obviaremos los métodos de los PageObjects para quitar ruido.

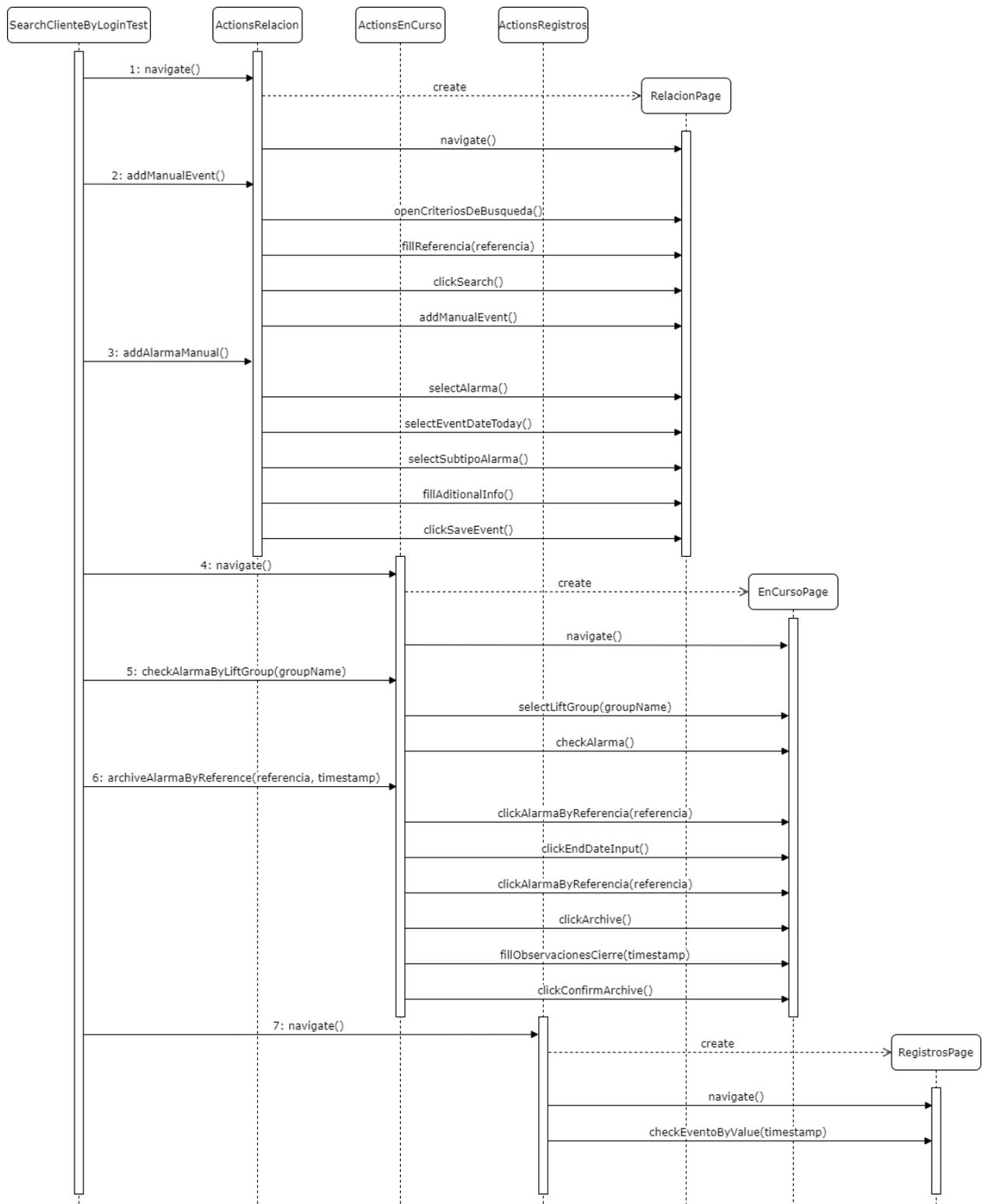


Figura 5.47. Diagrama de secuencia para la automatización de caso de uso.

A pesar de la longitud de la prueba, gracias a la previa tarea de síntesis y la aplicación del patrón de diseño, nos quedaría algo como esto:

Código 5.5. AddAlarmaToLiftTest.java

```
public class AddAlarmaToLiftTest extends BaseTest {

    String testDescription = "Add alarma to a lift test.";

    @Test
    public void addAlarmaToLiftTest() {
        setUpExtentTest(testDescription);
        acciones.loginPage().login(data.getUser());
        acciones.relacionDeAscensoresPage().navigate();
        acciones.relacionDeAscensoresPage()
            .addManualEvent(lift.getReferencia());
        acciones.relacionDeAscensoresPage().addAlarmaManual();
        acciones.enCursoPage().navigate();
        acciones.enCursoPage().checkAlarmaByLiftGroup(lift.getGrupo());
        String timestamp = new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss");
        acciones.enCursoPage().archiveAlarmaByLiftReference(
            data.getLift().getReferenciaAscensor(), timestamp);
        acciones.registrosPage().navigate();
        acciones.registrosPage().checkEventoByValue(timestamp);
        acciones.logout();
    }
}
```

6. CONCLUSIONES

6.1. Conclusiones del proyecto

La creación de un proyecto de automatización de pruebas de regresión ha supuesto mucho más que la implementación del mismo: conocer la aplicación web a probar, aprender a desplegarla o identificar que casos de prueba son realmente útiles para encontrar errores y regresiones ha sido un verdadero desafío.

Con este proyecto me he dado cuenta de la importancia que tiene en la industria del software la gestión de la calidad y de lo amplio que puede llegar a ser ese terreno. Tanto es así que el papel del ingeniero de calidad de software es imprescindible para cualquier proyecto con un mínimo de rigor.

El mayor reto, sin duda, ha sido la parte de configuración del entorno de pruebas. Algo que en principio parecía fácil, ha supuesto un verdadero reto por la cantidad de detalles implicados. Sin embargo, me ha aportado un conocimiento muy valioso en materia de virtualización.

Durante este trabajo he tenido que aplicar gran parte de los conocimientos aprendidos durante la carrera, pero sobre todo mejorar la capacidad de autosuficiencia y autoaprendizaje. Nuevas tecnologías como Maven, Selenium y Vagrant me han ayudado a conseguir el objetivo de este trabajo: automatizar casos de prueba en un entorno de desarrollo de la manera más mantenible y robusta posible.

6.2. Líneas de trabajo futuras

La puesta en marcha del framework de automatización sólo ha permitido explotar una mínima parte de lo que es capaz de ofrecer una herramienta como lo es Selenium. Es por ello que se podría crear una estrategia de pruebas mucho más amplia y aumentar los casos de prueba de regresión para asegurar el correcto funcionamiento de la aplicación web.

Otro aspecto que podría mejorar sería el uso de Selenium Grid, que a diferencia del Selenium WebDriver, éste permite el uso de distintos navegadores. Es bastante común que haya errores específicos de ciertos navegadores, por lo que tener acceso a varios de ellos aportaría datos muchos más reales sobre el estado de la web.

Incluso haciendo el uso de Selenium Grid, esto no deja de ser una herramienta de frontend. Sería una buena idea añadir testing de la parte de backend para asegurar que el esqueleto de la aplicación funcione como debe. Una ventaja que tienen los casos de prueba de backend respecto a los de frontend es que son mucho más estables. Esto se debe a que la mayor parte de los cambios en el software de una aplicación se aplican a la parte de la interfaz de usuario.

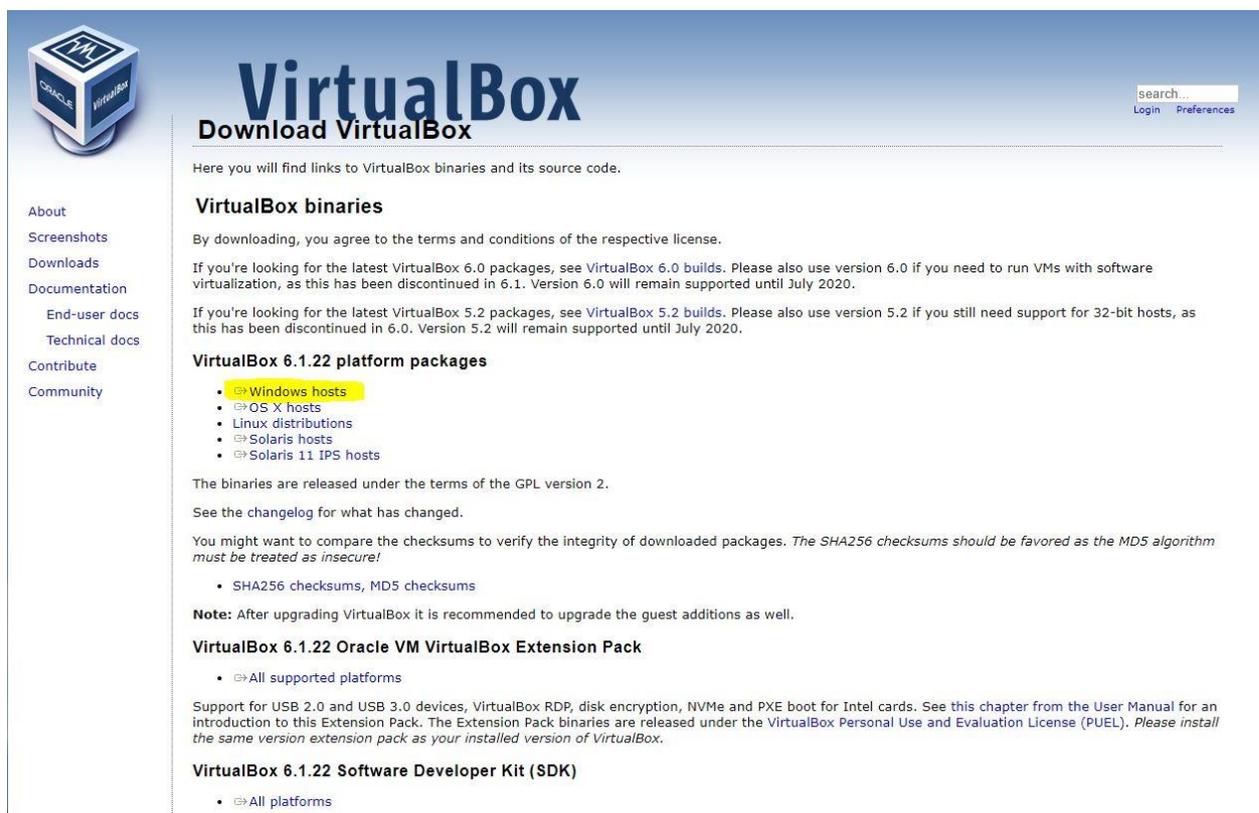
Para finalizar sería posible añadir casos de pruebas no funcionales de estrés, carga, volumen o seguridad. Aunque estos tipos de prueba puedan ser más costosos, pueden prevenir problemas de calidad potencialmente catastróficos luego de la salida a producción.

ANEXO A: ENTORNO DE PRUEBAS

En este anexo vamos a explicar como desplegar el entorno completo de pruebas en Windows. Para ello usaremos la herramienta Vagrant, que con ayuda de VirtualBox virtualizaremos una máquina Linux donde se ejecutará la aplicación web y tendremos disponible el código del proyecto de pruebas.

1. Instalar VirtualBox

Descargaremos la última versión de VirtualBox para Windows. Para ello iremos a la página oficial <https://www.virtualbox.org/wiki/Downloads> y seleccionaremos Windows hosts dentro de platform packages.



The screenshot shows the 'Download VirtualBox' page on the Oracle VM VirtualBox website. The page title is 'VirtualBox Download VirtualBox'. On the left, there is a navigation menu with links for 'About', 'Screenshots', 'Downloads', 'Documentation', 'End-user docs', 'Technical docs', 'Contribute', and 'Community'. The main content area includes a search bar, a login link, and a preferences link. The text on the page states: 'Here you will find links to VirtualBox binaries and its source code.' Under the heading 'VirtualBox binaries', it says 'By downloading, you agree to the terms and conditions of the respective license.' It provides information about the latest VirtualBox 6.0 packages and VirtualBox 5.2 packages. Under the heading 'VirtualBox 6.1.22 platform packages', there is a list of links: 'Windows hosts' (highlighted in yellow), 'OS X hosts', 'Linux distributions', 'Solaris hosts', and 'Solaris 11 IPS hosts'. Below this, it mentions 'The binaries are released under the terms of the GPL version 2.' and 'See the changelog for what has changed.' It also includes a note about comparing checksums to verify integrity, stating 'The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!'. There is a list of 'SHA256 checksums, MD5 checksums'. A 'Note' states: 'After upgrading VirtualBox it is recommended to upgrade the guest additions as well.' Under the heading 'VirtualBox 6.1.22 Oracle VM VirtualBox Extension Pack', there is a link for 'All supported platforms'. It provides support information for USB 2.0 and USB 3.0 devices, VirtualBox RDP, disk encryption, NVMe and PXE boot for Intel cards. Under the heading 'VirtualBox 6.1.22 Software Developer Kit (SDK)', there is a link for 'All platforms'.

Figura A.1. Descarga VirtualBox.

Aceptamos todos los diálogos y finalizamos la instalación.

2. Instalar Vagrant

Nos dirigiremos a la página oficial de Vagrant <https://www.vagrantup.com/downloads> y elegiremos en la sección de descargas Windows 64-bit.

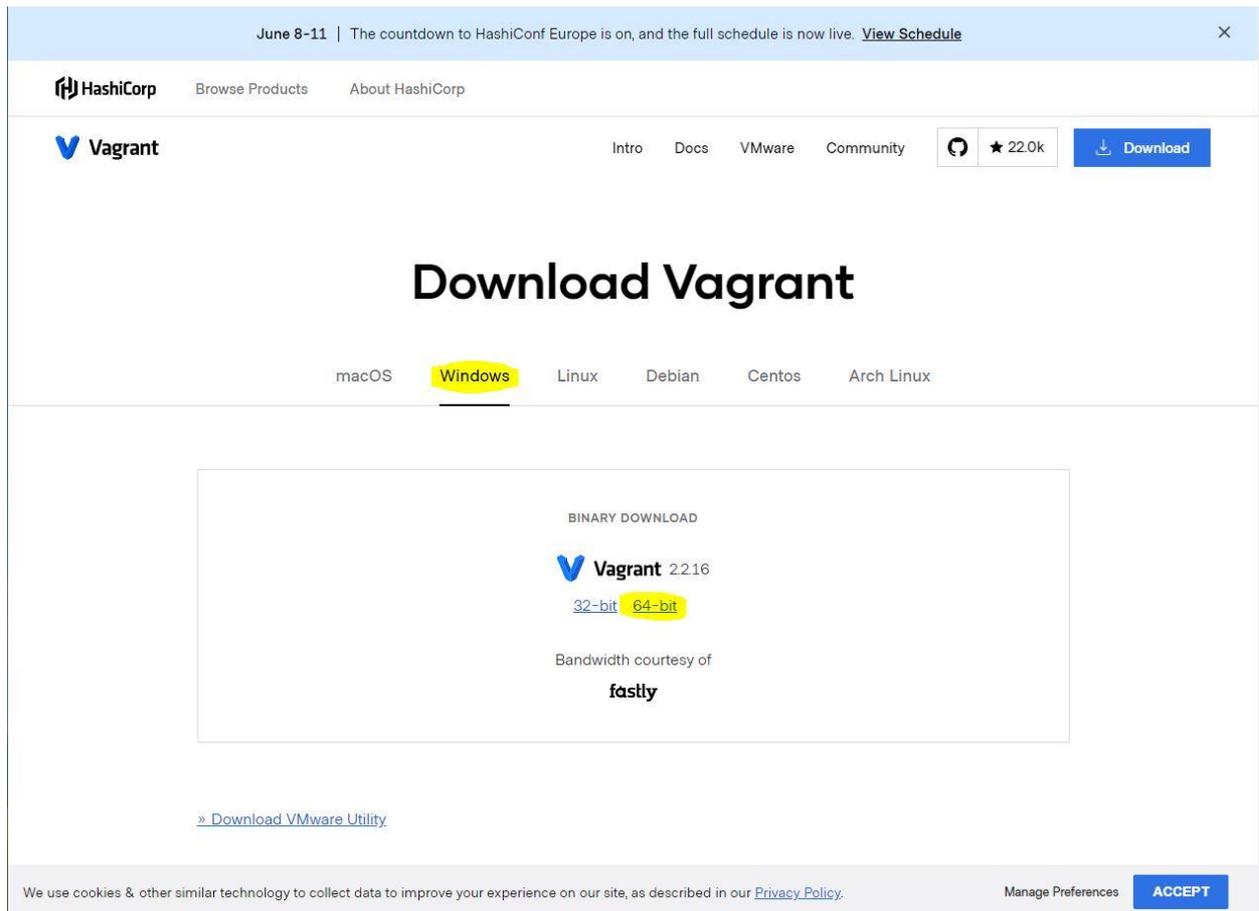


Figura A.2. Descarga Vagrant.

Al igual que con VirtualBox, aceptamos todos los diálogos del instalador y finalizamos. También tenemos que instalar un plugin de vagrant. Para ello, abrimos la terminal y escribimos el siguiente comando:

```
$ vagrant plugin install vagrant-reload
```

Este plugin sirve para reiniciar la máquina virtual tras instalar todo el entorno por primera vez. Se utiliza en la última línea del Vagrantfile, que veremos a continuación en el Código A.1.

3. Desplegar el entorno de pruebas

Abriremos la terminal y ejecutaremos los siguientes comandos:

```
$ mkdir deploy_dir
$ cd deploy_dir
$ echo > Vagrantfile
$ echo > setup.sh
```

Una vez creados los dos ficheros, Vagrantfile deberá contener el siguiente código:

Código A.1. Vagrantfile

```
# encoding: utf-8
# -*- mode: ruby -*-
# vi: set ft=ruby :
# Box / OS
VAGRANT BOX = 'bento/debian-10.6'
# Memorable name for your
VM NAME = 'TFG'
# VM User - 'vagrant' by default
VM USER = 'debian'
# Host folder to sync
HOST PATH = '.'
# Where to sync to on Guest - 'vagrant' is the default user name
GUEST PATH = '/home/' + VM USER
# # VM Port - uncomment this to use NAT instead of DHCP
# VM PORT = 8080
Vagrant.configure(2) do |config|
  # Vagrant box from Hashicorp
  config.vm.box = VAGRANT BOX
  # Actual machine name
  config.vm.hostname = VM NAME
  # Set VM name in Virtualbox
  config.vm.provider "virtualbox" do |v|
    v.name = VM NAME
    v.memory = 4096
    v.gui = true
    v.cpus = 2
    v.customize ["modifyvm", :id, "--vram", "128"]
  end
  #DHCP - comment this out if planning on using NAT instead
  config.vm.network "forwarded port", guest: 49001, host: 49001
  # # Port forwarding - uncomment this to use NAT instead of DHCP
  # config.vm.network "forwarded port", guest: 80, host: VM PORT
  # Sync folder
  config.vm.synced folder HOST PATH, GUEST PATH
  # Disable default Vagrant folder, use a unique path per project
  config.vm.synced folder '.', '/home/'+VM USER+'', disabled: true
  # Install testing environment
  config.vm.provision "shell", path: "setup.sh"
  #vagrant plugin install vagrant-reload
  config.vm.provision :reload
end
```

Es importante que le demos suficiente potencia a la máquina para que no sea una molestia trabajar en ella. Le hemos asignado 4GB de RAM y 2 CPUs. También le hemos puesto 128Mb a la memoria de vídeo para que la pantalla sea fluida y acepte resoluciones como 1920x1080. Por defecto viene en 8Mb y no es práctico en absoluto.

Si nos fijamos bien, lo que realiza la configuración de la máquina virtual es el contenido de setup.sh. Para desplegar un entorno de pruebas completo en una máquina virtual necesitaremos: instalar herramientas básicas, instalar un IDE con el código de pruebas y desplegar la aplicación web. Vamos a repasar detenidamente todo lo necesario para el despliegue antes de mostrar el contenido de setup.sh.

3.1. Instalación de herramientas básicas

Lo primero de todo, Vagrant nos ofrece una máquina virtual carente de escritorio y GUI. En lugar de eso, tendremos una interfaz por línea de comandos como la de la Figura A.3.

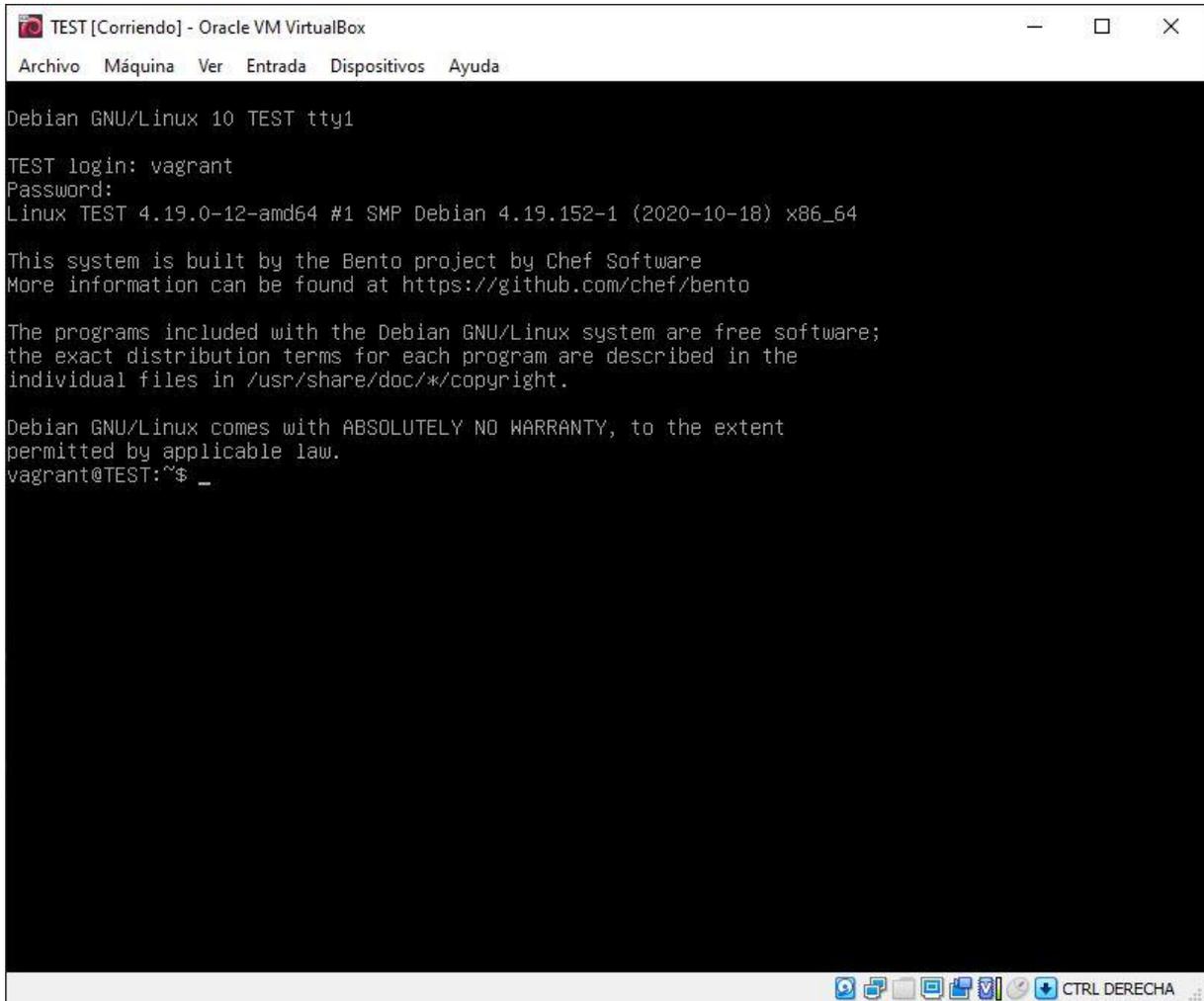


Figura A.3. Máquina virtual sin GUI.

Como es de esperar, no es práctico desarrollar código en un entorno de estas características. Para solventarlo podemos instalar entornos de escritorio disponibles para Debian. Primero probamos instalando xfce4, pero tiene una interfaz bastante obsoleta y apenas trae aplicaciones instaladas. Tras investigar un poco, decidí utilizar task-gnome-desktop. Su interfaz es más limpia y trae aplicaciones básicas instaladas como Firefox, editores de ficheros, etc. En la Figura A.4 vemos el aspecto que tendrá nuestro entorno.

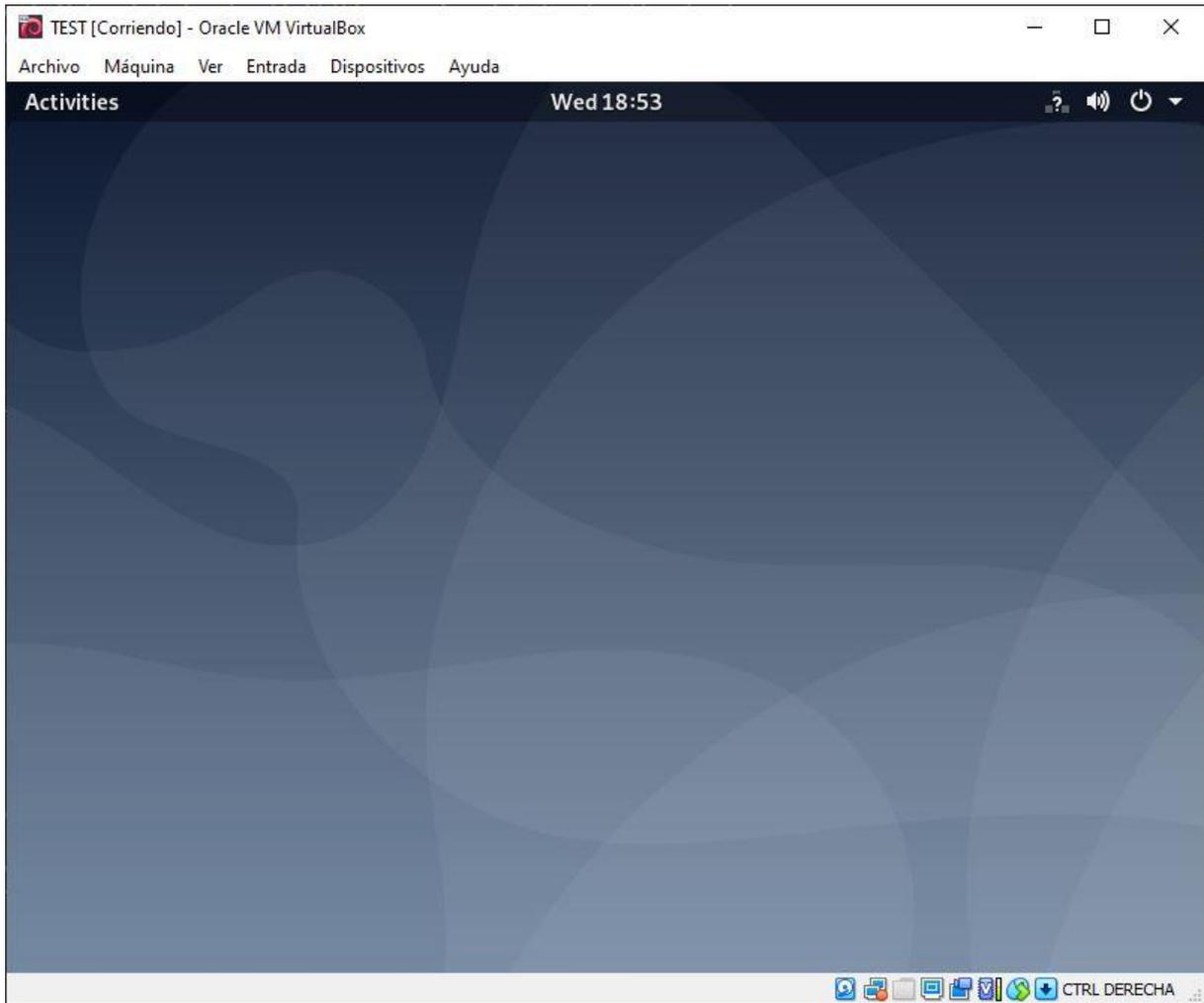


Figura A.4. Máquina virtual con GUI.

La interfaz gráfica no es la única herramienta que tendremos que instalar. Vamos a listar brevemente todo lo necesario para el entorno:

- JDK y JRE: sin ellos no podremos compilar y ejecutar el proyecto porque está escrito en Java.
- Curl: lo utilizaremos para descargar el conector de la base de datos con la aplicación.
- Unzip: con esta herramienta podremos descomprimir archivos descargados como Galleon, necesario en el proceso de instalación de WildFly (luego se explicará con más detalle).
- Mariadb: será la base de datos utilizada por la aplicación web. Sin ella no es posible el despliegue.
- Git: herramienta de control de versiones utilizada en el proyecto de automatización.
- Maven: herramienta de construcción del proyecto de automatización.
- Snapd: es un sistema de despliegue y manejo de paquetes, necesario para la instalación del IDE.

Para la instalación de todas estas herramientas y servicios, tenemos de ejemplo el código A.2.

Código A.2. Instalación de las herramientas básicas.

```
sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o DPkg::options::="--force-confdef" -o DPkg::options::="--force-confold" install grub-pc
sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o DPkg::options::="--force-confdef" -o DPkg::options::="--force-confold" dist-upgrade
```

```

sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::options::="--force-
confdef" -o Dpkg::options::="--force-confold" update
sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::options::="--force-
confdef" -o Dpkg::options::="--force-confold" install task-gnome-desktop
sudo sed -i 's/allowed users=.*$/allowed users=anybody/'
/etc/X11/Xwrapper.config
sudo apt install default-jre -y
sudo apt install default-jdk -y
sudo apt install curl -y
sudo apt install unzip -y
sudo apt install mariadb-server -y
sudo apt install git -y
sudo apt install maven -y
sudo apt autoremove --purge snapd
sudo apt install -y snapd

```

Si nos fijamos, las primeras líneas hacen referencia al modo no interactivo. Esto es porque esos comandos populan diálogos en los que hay que hacer decisiones utilizando las flechas del teclado. Como eso no es automatizable en un script, se lanzan en modo no interactivo para evitar ese problema. En la Figura A.5 tenemos un ejemplo de un comando interactivo.

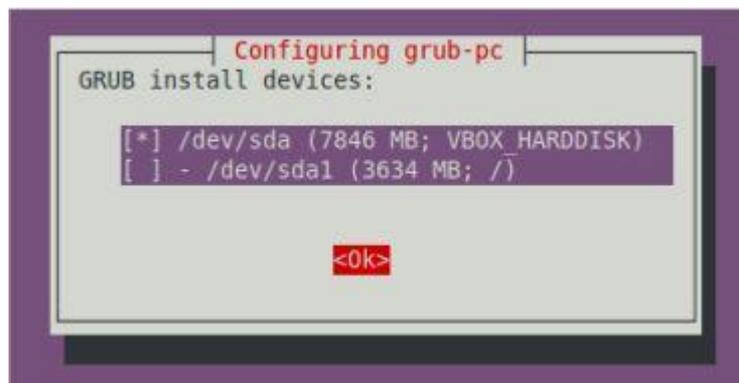


Figura A.5. Comando interactivo.

3.2. Instalación del IDE y códigos

Primero tendremos que crear el directorio donde IntelliJ guardará todos los proyectos, situado bajo el home del usuario: `/home/vagrant/IdeaProjects`. Para el proyecto de automatización clonaremos el código con git y le daremos permisos al usuario. Si no hacemos esto último, pertenecerá al usuario root y no se podrán editar los ficheros. Para el código de la aplicación web lo descargaremos del servidor del departamento y lo ubicaremos en el sitio correspondiente.

También tendremos que crear un directorio donde guardemos todos los ficheros descargados necesarios en el proceso de despliegue. Estará situado en `/home/debian/TFG`.

También usaremos snap para instalar los IntelliJ y Eclipse. Todos estos pasos están reflejados en el código A.3.

Código A.3. Instalación del IDE y código del proyecto.

```

sudo snap install --classic intelliJ-idea-community
sudo snap install --classic eclipse
sudo mkdir -p /home/debian/TFG

```

```
sudo mkdir -p /home/vagrant/IdeaProjects
cd /home/vagrant/IdeaProjects
git clone https://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:x-oauth-
basic@github.com/gameduser/sigma-web-test.git
sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/sigma.jsf-nuevo.tar.bzip2
sudo tar -xjf sigma.jsf-nuevo.tar.bzip2
sudo chown -R vagrant /home/vagrant/IdeaProjects/
cd /home/debian/TFG
```

3.3. Despliegue de Sigma4Lifts

Para desplegar la aplicación utilizaremos la herramienta WildFly. Con ella podremos arrancar y conectar la página web con la base de datos. Lo primero es crear la base de datos.

3.3.1. Creación de la base de datos

La base de datos está sostenida por mariadb. Sus datos provienen de un fichero SQL inicial que descargamos desde la página web del departamento, donde tenemos algunos usuarios, ascensores, empresas, etc. Para crearla, seguiremos los siguientes pasos:

Código A.4. Creación de la base de datos.

```
sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/sigma desarrollo.sql.bzip2
sudo bzip2 -d sigma desarrollo.sql.bzip2
sudo mv sigma desarrollo.sql.bzip2.out sigma desarrollo.sql
sudo systemctl enable mariadb
sudo cat <<EOF | sudo mysql
CREATE DATABASE sigma desarrollo;
CREATE USER 'sigma'@'localhost' IDENTIFIED BY 'sigmadb';
GRANT ALL PRIVILEGES ON sigma desarrollo.* TO 'sigma'@'localhost';
EOF
sudo mysql sigma desarrollo < sigma desarrollo.sql
```

Ahí podemos ver que el proceso es descargar la información comprimida del servidor del departamento, descomprimirla en el lugar adecuado y volcarla en la base de datos. Además, con la línea marcada en rojo le estamos diciendo a la máquina que inicie la base de datos siempre con cada arranque.

3.3.2. Configuración de Sigma4Lifts

Hay ciertos ficheros con parámetros necesarios para el funcionamiento de la aplicación. Los tenemos disponibles en el servidor del departamento, por lo que solo tendremos que bajarlos y colocarlos en el lugar correspondiente. Podemos ver el resultado en el código A.5.

Código A.5. Configuración de la aplicación

```
sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/etc_sigma.tar.bzip2
```

```
sudo tar xjf etc sigma.tar.bzip2
sudo cp -r etc/sigma /etc/sigma
```

3.3.3. Instalación de WildFly

Como hemos dicho antes, WildFly es la herramienta de despliegue utilizada en este entorno. Para instalarla utilizaremos Galleon, una herramienta de provisionamiento diseñada para crear y mantener distribuciones de software compuestas por uno o más productos. Primero descargaremos el archivo comprimido de galleon y lo ubicaremos en la carpeta correspondiente. Tras ello instalaremos WildFly y definiremos sus usuarios. Podemos ver el resultado en el código A.6.

Código A.6. Instalación de WildFly con Galleon.

```
sudo wget
https://github.com/wildfly/galleon/releases/download/4.2.8.Final/galleon-
4.2.8.Final.zip
sudo unzip galleon-4.2.8.Final.zip -d /opt
sudo mv /opt/galleon-4.2.8.Final /opt/galleon
sudo /opt/galleon/bin/galleon.sh install wildfly:current --dir=/opt/wildfly
sudo groupadd -r wildfly
sudo useradd -r -g wildfly -d /opt/wildfly -s /sbin/nologin wildfly
sudo chown -RH wildfly: /opt/wildfly
sudo mkdir -p /etc/wildfly
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.conf /etc/wildfly/
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/launch.sh
/opt/wildfly/bin/
sudo chmod +x /opt/wildfly/bin/*.sh
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.service
/etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable wildfly
sudo systemctl start wildfly
sudo /opt/wildfly/bin/add-user.sh -u admin -p admin -g
PowerUser,BillingAdmin, -e
```

3.3.4. Despliegue del conector de la base de datos

Para que la base de datos pueda comunicarse con la aplicación es necesario un conector. Tenemos uno disponible en la página oficial de mariadb. Lo descargaremos y lo desplegaremos con WildFly:

Código A.7. Despliegue del conector de la base de datos.

```
curl -o mariadb-java-client-2.6.2.jar --insecure
https://downloads.mariadb.com/Connectors/java/connector-java-2.6.2/mariadb-
java-client-2.6.2.jar
sudo /opt/wildfly/bin/jboss-cli.sh --connect --commands='deploy mariadb-
java-client-2.6.2.jar --name=mariadb --runtime-name=mariadb-java-client-
2.6.2.jar'
sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
user=admin --password=admin --commands='data-source add --jndi-
name=java:/sigma/datasource --name=sigmadb --connection-
url=jdbc:mysql://127.0.0.1:3306/sigma desarrollo --driver-
class=org.mariadb.jdbc.Driver --driver-name=mariadb-java-client-2.6.2.jar --
user-name=sigma --password=sigmadb --statistics-enabled --background-
validation --valid-connection-checker-class-
```

```
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker
--exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter'
```

3.3.5. Despliegue de la Sigma4Lifts

Por último tenemos que desplegar la aplicación web. Para ello necesitamos el archivo WAR de la aplicación. Se trata de un archivo JAR utilizado para distribuir una colección de JavaServer Pages, servlets, clases Java, archivos XML, bibliotecas de tags y páginas web estáticas (HTML y relacionados) que juntos constituyen una aplicación web.

Dicho archivo puede conseguirse ejecutando el comando package de Maven en el directorio raíz del código. Esto generará el archivo WAR dentro de la carpeta target. Sin embargo, nosotros usaremos un archivo WAR disponible en el repositorio del departamento.

Explicado esto, los pasos a seguir serían descargar el WAR, colocarlo en la carpeta correspondiente y desplegarlo con ciertos comandos. Podemos ver el resultado en el código A.6.

Código A.6. Despliegue de Sigma4Lifts.

```
sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/war/sigma.war
sudo cp sigma.war /opt/wildfly/standalone/deployments
sudo /opt/wildfly/bin/jboss-cli.sh --connect --user=admin --password=admin -
-commands='/system-property=sigma.base:add(value=/etc/sigma) '
sudo /opt/wildfly/bin/jboss-cli.sh --connect --user=admin --password=admin -
-commands='/socket-binding-group=standard-sockets/socket-binding=http:write-
attribute(name=port,value="{jboss.https.port:49001}") '
sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
command='/subsystem=security:write-attribute(name=initialize-jacc,
value=false) '
systemctl stop wildfly
systemctl start wildfly
sleep 10
sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
command='/subsystem=elytron/policy=jacc:add(jacc-policy={}) '
sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
command='/subsystem=undertow/application-security-domain=other:add(security-
domain=ApplicationDomain,integrated-jaspi=false) '
```

3.4. Arrancar el entorno

Ya hemos visto todos los pasos necesarios para la creación del entorno de pruebas. Si lo juntamos todo con algunas modificaciones en el fichero setup.sh, tendríamos el siguiente contenido:

Código A.7. setup.sh

```
#!/bin/sh

tools installation() {
    sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o DPkg::options::="--
force-confdef" -o DPkg::options::="--force-confold" install grub-pc
    sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o DPkg::options::="--
force-confdef" -o DPkg::options::="--force-confold" dist-upgrade
```

```

    sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o DPkg::options::="--force-confdef" -o DPkg::options::="--force-confold" update
    sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o DPkg::options::="--force-confdef" -o DPkg::options::="--force-confold" install task-gnome-desktop
    sudo sed -i 's/allowed users=.*$/allowed users=anybody/'
/etc/X11/Xwrapper.config
    sudo apt install default-jre -y
    sudo apt install default-jdk -y
    sudo apt install curl -y
    sudo apt install unzip -y
    sudo apt install mariadb-server -y
    sudo apt install git -y
    sudo apt install maven -y
    sudo apt autoremove --purge snapd
    sudo apt install -y snapd
}

ide and code installation() {
    sudo snap install --classic eclipse
    sudo snap install --classic intellij-idea-community
    sudo mkdir -p /home/debian/TFG
    sudo mkdir -p /home/vagrant/IdeaProjects
    cd /home/vagrant/IdeaProjects
    git clone https://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:x-oauth-basic@github.com/gameduser/sigma-web-test.git
    sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/sigma.jsf-nuevo.tar.bzip2
    sudo tar -xjf sigma.jsf-nuevo.tar.bzip2
    sudo chown -R vagrant /home/vagrant/IdeaProjects/
    cd /home/debian/TFG
}

database creation() {
    sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/sigma desarrollo.sql.bzip2
    sudo bzip2 -d sigma desarrollo.sql.bzip2
    sudo mv sigma desarrollo.sql.bzip2.out sigma desarrollo.sql
    sudo systemctl enable mariadb
    sudo cat <<EOF | sudo mysql
CREATE DATABASE sigma desarrollo;
CREATE USER 'sigma'@'localhost' IDENTIFIED BY 'sigmadb';
GRANT ALL PRIVILEGES ON sigma desarrollo.* TO 'sigma'@'localhost';
EOF
    sudo mysql sigma desarrollo < sigma desarrollo.sql
}

application configuration() {
    sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/etc sigma.tar.bzip2
    sudo tar xjf etc sigma.tar.bzip2
    sudo cp -r etc/sigma /etc/sigma
}

wildfly installation() {
    sudo wget
https://github.com/wildfly/galleon/releases/download/4.2.8.Final/galleon-4.2.8.Final.zip
    sudo unzip galleon-4.2.8.Final.zip -d /opt

```

```

sudo mv /opt/galleon-4.2.8.Final /opt/galleon
sudo /opt/galleon/bin/galleon.sh install wildfly:current --
dir=/opt/wildfly
sudo groupadd -r wildfly
sudo useradd -r -g wildfly -d /opt/wildfly -s /sbin/nologin wildfly
sudo chown -RH wildfly: /opt/wildfly
sudo mkdir -p /etc/wildfly
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.conf
/etc/wildfly/
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/launch.sh
/opt/wildfly/bin/
sudo chmod +x /opt/wildfly/bin/*.sh
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.service
/etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable wildfly
sudo systemctl start wildfly
sudo /opt/wildfly/bin/add-user.sh -u admin -p admin -g
PowerUser,BillingAdmin, -e
}

database connection deployment() {
    curl -o mariadb-java-client-2.6.2.jar --insecure
https://downloads.mariadb.com/Connectors/java/connector-java-2.6.2/mariadb-
java-client-2.6.2.jar
    sudo /opt/wildfly/bin/jboss-cli.sh --connect --commands='deploy mariadb-
java-client-2.6.2.jar --name=mariadb --runtime-name=mariadb-java-client-
2.6.2.jar'
    sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
user=admin --password=admin --commands='data-source add --jndi-
name=java:/sigma/datasource --name=sigmadb --connection-
url=jdbc:mysql://127.0.0.1:3306/sigma desarrollo --driver-
class=org.mariadb.jdbc.Driver --driver-name=mariadb-java-client-2.6.2.jar --
user-name=sigma --password=sigmadb --statistics-enabled --background-
validation --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker
--exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter'
}

sigma deployment() {
    sudo wget --user eduardodiaz --password ed54010
https://everest.us.es/code/war/sigma.war
    sudo cp sigma.war /opt/wildfly/standalone/deployments
    sudo /opt/wildfly/bin/jboss-cli.sh --connect --user=admin --
password=admin --commands='/system-property=sigma.base:add(value=/etc/sigma) '
    sudo /opt/wildfly/bin/jboss-cli.sh --connect --user=admin --
password=admin --commands='/socket-binding-group=standard-sockets/socket-
binding=http:write-attribute(name=port,value="{jboss.https.port:49001}") '
    sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
command='/subsystem=security:write-attribute(name=initialize-jacc,
value=false) '
    systemctl stop wildfly
    systemctl start wildfly
    sleep 10
    sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
command='/subsystem=elytron/policy=jacc:add(jacc-policy={}) '

```

```
sudo /opt/wildfly/bin/jboss-cli.sh --connect --controller=127.0.0.1 --
command='/subsystem=undertow/application-security-domain=other:add(security-
domain=ApplicationDomain,integrated-jaspi=false) '
}

application deployment() {
    database creation
    application configuration
    wildfly installation
    database connection deployment
    sigma deployment
    database connection deployment
    sudo systemctl restart wildfly
    sudo sleep 10
}

tools installation
ide_and_code_installation
application_deployment
```

Con el Vagrantfile y setup.sh en el mismo directorio, ejecutaremos el comando:

```
$ vagrant up
```

Al hacerlo se levantará la máquina virtual y comenzará el proceso de instalación. La primera vez tardará varios minutos. Durante ese tiempo la máquina estará sin GUI. No hay que preocuparse por eso. Una vez finalice, se reiniciará ya con la interfaz gráfica y todo instalado. Es recomendable cambiar la región porque por defecto viene en inglés y el teclado no coincide totalmente. También cambiar la resolución para tener más espacio en las ventanas. Para comprobar que la aplicación se ha instalado correctamente, abriremos el navegador Firefox e ingresaremos el siguiente link:

<http://127.0.0.1:49001/sigma>

Deberá aparecernos la página de login de la aplicación como la mostrada en la Figura A.6. Para hacer login, S4L020 y 1234 son el usuario y clave respectivamente.



Figura A.6. Despliegue correcto de Sigma4Lifts en la máquina virtual.

Por último, vamos a comprobar que el IDE y el código están correctamente instalados. Abriremos la barra de búsqueda para ver si tenemos instalado IntelliJ IDEA como en la figura A.7.

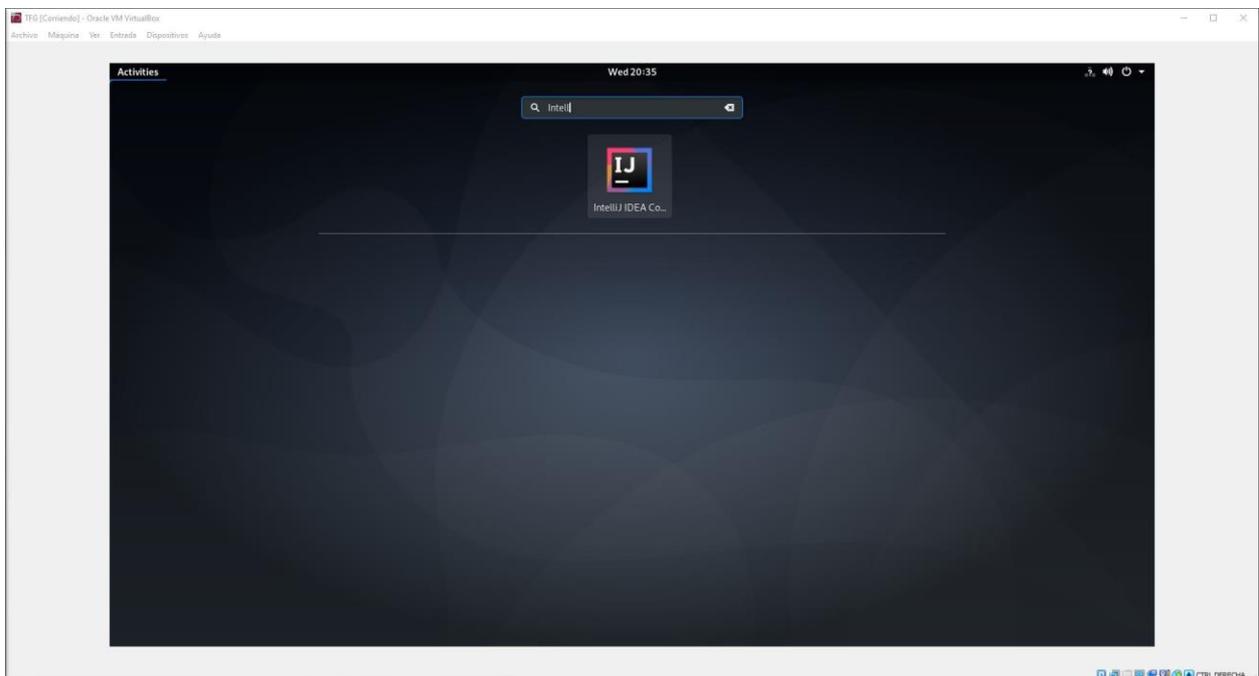


Figura A.7. IntelliJ IDEA instalado correctamente.

Lo abrimos y nos aparecerá un diálogo preguntándonos por los términos y condiciones y la compartición de datos para analíticas. Los pasamos y se nos abrirá el entorno. Ahí seleccionaremos “Open” para comprobar que tengamos el código clonado en la máquina.

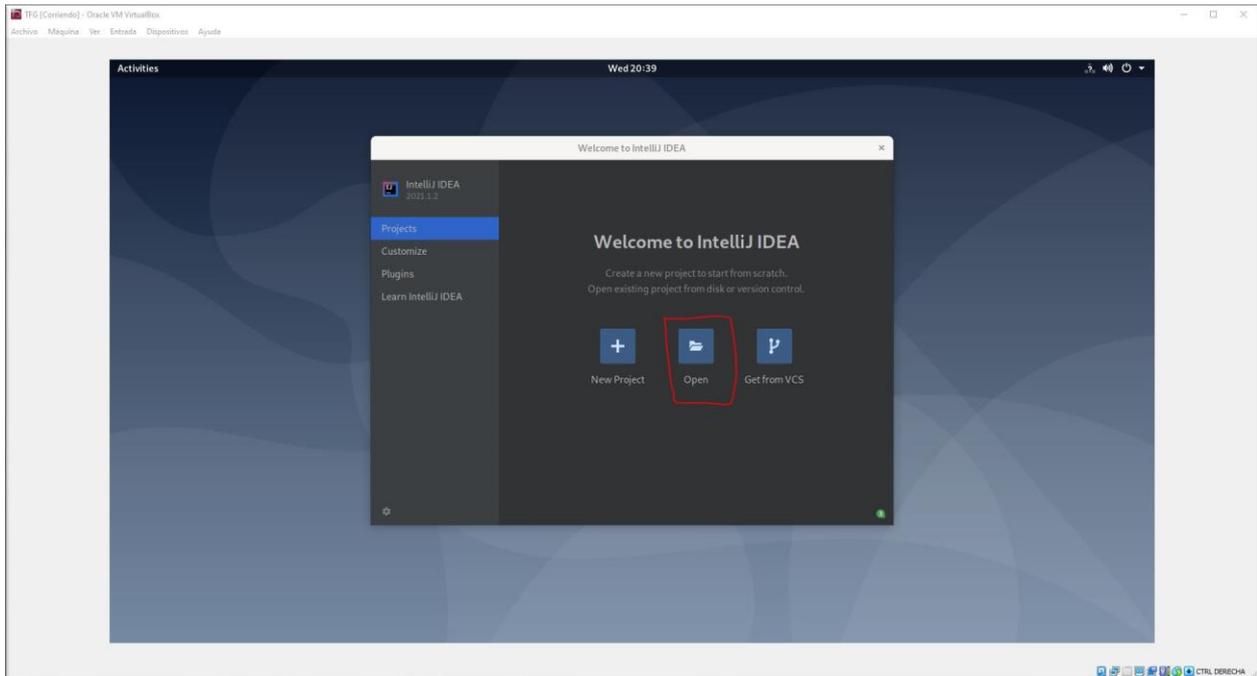


Figura A.8. Abrir proyecto en IntelliJ IDEA.

Tras hacer click nos preguntará qué proyecto queremos abrir. Buscaremos el proyecto sigma-web-test bajo el directorio IdeaProjects explicado en puntos anteriores como se muestra en la figura A.9.

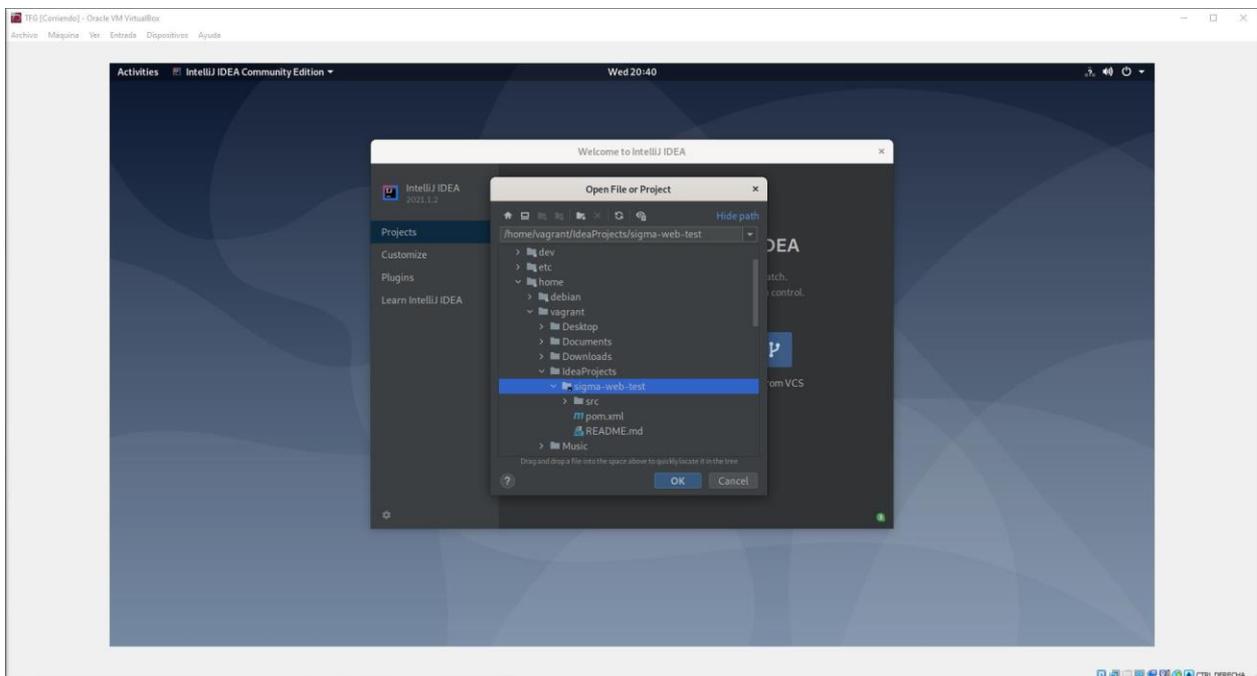


Figura A.9. Abrir proyecto de automatización sigma-web-test.

Aceptamos el diálogo de confiar en los proyectos bajo dicho directorio y nos aparecerá el código en la columna izquierda. Este proyecto necesita el SDK 15 para funcionar, por lo que comprobaremos si está en esa versión. Para ello iremos a File > Project Structure. Ahí abriremos el desplegable del SDK y si no tenemos la versión 15 haremos click en Add SDK > Download SDK como vemos en la figura A.10.

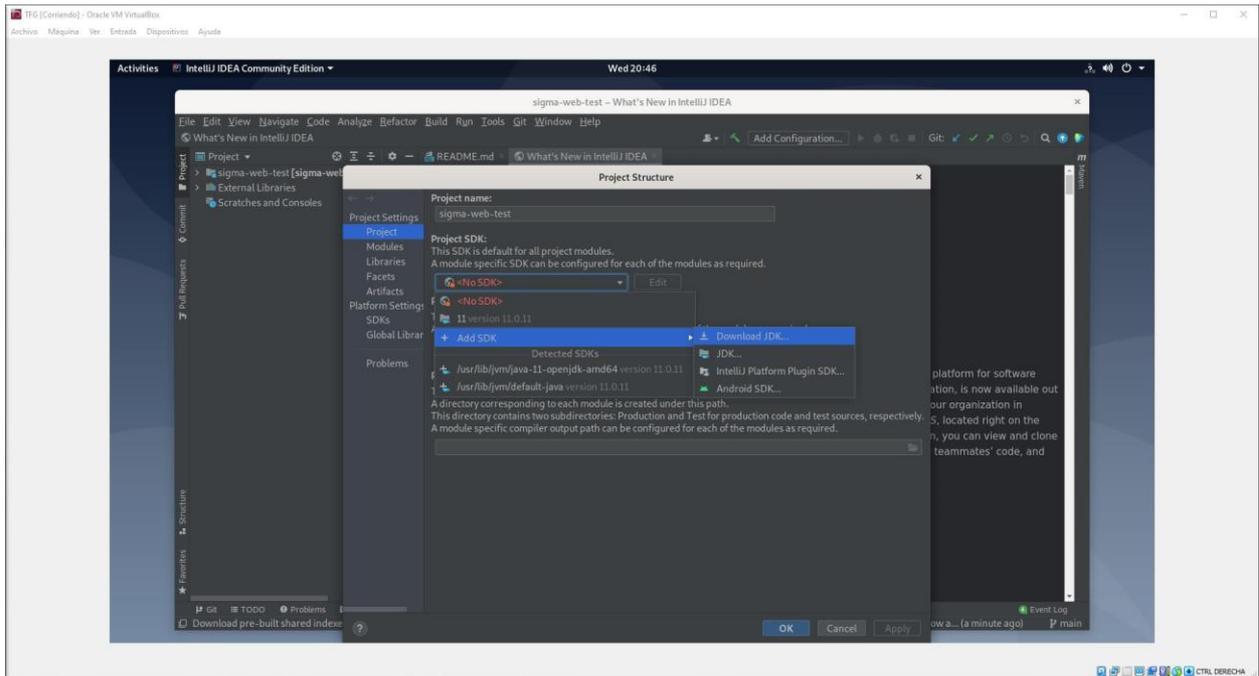


Figura A.10. Descargar SDK.

En la ventana emergente elegiremos cualquiera de las versiones 15. Nosotros usaremos la de coretto-15. Una vez hecho esto ya tenemos listo el entorno. Para ejecutarlo revisar el punto 4.3.

REFERENCIAS

- [1] Colaboradores de Wikipedia, «Pruebas de software» *Wikipedia, La enciclopedia libre*, 2021 mayo 26.
- [2] Colaboradores de Wikipedia, «Automatización de pruebas» *Wikipedia, La enciclopedia libre*, 30 enero 2021.
- [3] Colaboradores de Wikipedia, «Caja blanca» *Wikipedia, La enciclopedia libre*, 13 julio 2019.
- [4] Colaboradores de Wikipedia, «Caja negra» *Wikipedia, La enciclopedia libre*, 23 septiembre 2020.
- [5] Colaboradores de Wikipedia, «Prueba unitaria» *Wikipedia, La enciclopedia libre*, 8 febrero 2021.
- [6] Colaboradores de Wikipedia, «Pruebas de integración» *Wikipedia, La enciclopedia libre*, 31 agosto 2019.
- [7] Colaboradores de Wikipedia, «Pruebas de humo» *Wikipedia, La enciclopedia libre*, 6 febrero 2021.
- [8] Colaboradores de Wikipedia, «Pruebas de regresión» *Wikipedia, La enciclopedia libre*, 5 noviembre 2020.
- [9] Colaboradores de Wikipedia, «Selenium» *Wikipedia, La enciclopedia libre*, 11 mayo 2021.
- [10] Colaboradores Wikipedia, «Maven» *Wikipedia, La enciclopedia libre*, 10 octubre 2020.
- [11] Colaboradores de Wikipedia, «TestNG» *Wikipedia, La enciclopedia libre*, 1 julio 2020.
- [12] Colaboradores de Wikipedia, «Log4j» *Wikipedia, La enciclopedia libre*, 27 julio 2020.
- [15] Colaboradores de Wikipedia, «Calidad de Software» *Wikipedia, La enciclopedia libre*, 22 marzo 2021.