Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática Mención Tecnologías de la Información

Estudio de herramientas de automatización de pruebas software en aplicaciones web

Autor:
D. Jorge García Ruiz

Tutores:
Dra. Yania Crespo González-Carvajal
D. Juan José Juste Delgado

Agradecimientos

La etapa universitaria es algo nuevo, duro y diferente a lo que nos tenemos que enfrentar si queremos están en posesión de un título universitario. Y ahora que me encuentro en la fase final, tengo que agradecer a todas aquellas personas que me apoyaron en momentos difíciles, ya que, si no hubiese sido por ellos, esto proyecto no hubiese visto la luz.

A mis padres, José Miguel y Lucía, gracias a ellos, pude estudiar la titulación que deseaba. Además de toda la confianza, ilusión y compresión que tuvieron en mí. También a mi hermano Miguel, que siempre ha estado ahí animando y sacándome sonrisas en momentos adversos.

A mis tíos y primos, en especial, a Luis y María José,

por la convivencia con ellos y por todos aquellos momentos, tanto buenos como malos, durante estos cuatro años.

A mis abuelos, en especial, a mi abuelo paterno Pepe, que nos dejó el año pasado, pero su ilusión, apoyo, motivación y paciencia ha sido y será una de las fuentes guía que tendré presente siempre.

A todos mis compañeros de clase, su ayuda y motivación ha sido básica para seguir el camino hacia adelante.

A todos mis amigos, siempre han estado ahí, sacando sonrisas y apoyando en los momentos más duros, cuando más falta hacía.

A mi tutora Yania Crespo, tutora de la Universidad, por su ayuda, apoyo y comprensión durante estos meses de realización de este proyecto.

A Juan José Juste, tutor de Everis, sus consejos, ayuda, motivación y dedicación durante este tiempo.

A todos mis compañeros de Everis, por su apoyo y ánimo diario durante estos últimos meses.

A todos vosotros sólo os puedo decir muchas gracias.

Resumen

Este proyecto tiene como finalidad realizar un estudio sobre la automatización de pruebas de aceptación funcionales sobre una aplicación web.

Para ello, se define un plan de pruebas que se implementa y ejecuta con diferentes herramientas de automatización y se realiza un informe comparativo de las mismas.

De esta forma, se establece cuál de ellas es más usable o se adapta mejor a un entorno dado, dependiendo de unas características o requisitos iniciales tanto de la aplicación como del equipo de pruebas o desarrollo.

Finalmente, se realiza un análisis sobre las ventajas que tiene la automatización de pruebas y el valor que aporta para las metodologías de desarrollo ágiles.

Abstract

This project aims to conduct a study on the automation of functional acceptance tests on a web application.

To do this, a test plan is defined that is implemented and executed with different automation tools and a comparative report of them is made.

In this way, it is established which is more usable or better suited to a given environment, depending on some initial characteristics or requirements of both the application and the test or development team.

Finally, an analysis is made of the advantages of automation of tests and the value that advances for agile development methodologies.

Tabla de contenidos

Tabla de Figuras

Índice de Tablas

Capítulo I: Introducción

1.1. Contexto y Motivación.

La realización de pruebas en cualquier proyecto que contenga uno o múltiples desarrollos, es una de las fases básicas para garantizar que la aplicación cumple los requisitos marcados por el usuario, consiguiendo así una funcionalidad "aceptable".

En definitiva, si realizamos unas buenas pruebas de nuestros desarrollos obtendremos una solución a la petición del usuario acorde a la funcionalidad que solicitó.

En muchas ocasiones, la fase de pruebas no se realiza con toda la dedicación que correspondería para garantizar que todo funciona correctamente.

La causa principal de esta situación es la ausencia del tiempo que se dispone para destinar a las pruebas por lo que a veces, causa a las empresas problemas con el cliente debido a que el funcionamiento no era el esperado y por tanto con una calidad por debajo de la esperada. Una buena solución a estos problemas es la automatización de pruebas.

La automatización de pruebas es el proceso que consiste en mecanizar la ejecución de las pruebas que se van a realizar a un software, con el fin de obtener informes y, por tanto, un resultado tras la ejecución de las mismas y poder realizar la comparación con el resultado esperado.

"Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove."

Martin Fowler, Chief Scientist, ThoughtWorks

La motivación de la realización de este proyecto viene dada por la propuesta de este proyecto por la empresa Everis y por la posibilidad de comprobar lo útil y ágil que puede ser la realización de un plan de pruebas y automatizarlo, de esta forma se podrán conseguir unos resultados más afines a los que desea el cliente.

1.2. Objetivos.

Con el presente trabajo fin de grado se pretenden conseguir los siguientes objetivos:

- Realizar la automatización de un plan de pruebas funcionales y de aceptación en un entorno web, como prueba de concepto, con las diferentes herramientas de automatización que existen en el mercado, tanto de uso gratuito como de pago. Seleccionadas con un determinado criterio que se comentará más adelante en la presente memoria.

- Descubrir las ventajas de la automatización y la importancia de las mismas.

- Realizar un estudio comparativo de las herramientas utilizadas para poder concluir qué herramienta se adapta mejor según el tipo de herramienta y contexto.

De esta forma, comprobaremos, de primera mano, lo importante que es realizar un correcto plan de pruebas para comprobar el núcleo funcional de la aplicación. Sobre todo, si la aplicación está sometida a constantes cambios.

1.3. Estructura del documento.

Este documento describe la realización de un Trabajo Fin de Grado que contiene la siguiente disposición de capítulos y secciones.

Capítulo I. Introducción. En este capítulo se realiza una descripción del entorno, objetivos y alcance del proyecto.

Capítulo II. Planificación: Se detallará la planificación de tareas y estimación de tiempo en realizar las mismas. A su vez, se estimarán los posibles riesgos que se pueden dar en la realización de las mismas.

Capítulo III. Entorno de Pruebas. Este capítulo describe el entorno y el plan de pruebas que se va a realizar para probar la aplicación web.

Capítulo IV. Herramientas de automatización de pruebas. Se describirá en diferentes secciones cada una de las herramientas en cuestión para lanzar el plan de pruebas generado en el capítulo dos.

Capítulo V. Resultados y Comparativa. En este capítulo explicará detalladamente el estudio realizado de cada una de las herramientas y junto con una confrontación de los resultados obtenidos en el análisis de cada una de ellas.

Capítulo VI. Gestión del Proyecto. Se establecerá un seguimiento del proyecto detallando todos los riesgos que se dieron durante la realización del mismo.

Capítulo VII. Conclusiones y Trabajo Futuro. En este capítulo figuran las conclusiones extraídas al realizar el trabajo fin de grado y se detallan como trabajo futuro las aplicaciones a gran y pequeña escala de este estudio.

Referencias y Bibliografía. En esta sección se hará mención a todas las referencias de documentos o páginas web, de las que se ha extraído información para razonar y argumentar procedimientos y conclusiones que figuran en este proyecto

Anexos. En este apartado se detallará toda la información complementaria sobre las actividades y razonamientos del documento. En concreto:

Anexo I: En esta sección, se detallará un manual de usuario de la funcionalidad de la aplicación, ya que complementará el estudio realizado de la misma porque otorga el suficiente conocimiento funcional de la aplicación.

Anexo II: En este apartado se detallará como se realiza la instalación de Ruby, para poder implementar los planes de pruebas en las correspondientes herramientas.

Anexo III: En esta sección, se detallarán algunos aspectos importantes sobre la configuración que tienen que tener los navegadores para que se puedan ejecutar correctamente los planes de pruebas implementados.

Anexo IV: En este apartado se detallará la información que se añadirá en los CDs.

Capítulo II: Planificación

La metodología que se ha seguido para planificar la realización de este Trabajo Fin de Grado ha sido SCRUM. Debido a que tiene las siguientes características [22]:

- Ágil: La división del trabajo en pequeñas unidades funcionales (sprints) permite mantener fácilmente una política de entregas frecuentes de software que ofrecen una visión clara del estado del proceso y permite la posibilidad de realizar modificaciones a lo largo del ciclo de vida del proyecto.

- Simple: Se centra especialmente en facilitar el desarrollo rápido, por lo que su complejidad se ha tratado de reducir al máximo.

- Flexible: Todo el desarrollo se contempla como un ciclo de iteraciones continuas de desarrollo, lo que facilita la introducción de modificaciones espontáneas, con la colaboración del cliente, mejorando continuamente el proceso.

- Colaborativa: El planteamiento, desde el punto de vista de la organización del equipo, resulta bastante horizontal, otorgando a cada uno de los individuos que forman parte del equipo de desarrollo un elevado grado de autonomía y organización en su trabajo. Además, la relación colaborativa con el cliente facilita y a la vez, agiliza y garantiza un producto acorde al que este desea como producto final del proceso.

- Reduce costes: Al estar en constante relación con el cliente facilita el trabajo y la adecuación del producto final a las necesidades de éste. De forma que quede satisfecho.

Por tanto, por las razones expuestas anteriormente, es una metodología ideal para realizar este trabajo fin de grado.

A continuación, hay que definir una serie de detalles característicos de dicha metodología como son los roles y la disposición de tareas. Cabe destacar que los roles que se han seleccionado para este proyecto no son, en su totalidad, los que esta metodología emplea, debido a que, por causas evidentes, solo se ha realizado dicho proyecto por una persona, dando lugar a roles que no tienen ocupación dentro de este proyecto o que están embebidos en otros.

Los roles que forman parte de esta metodología aplicada a este proyecto, son los que podemos encontrar en la Tabla 1. [4]:

Rol	Persona Encargada	Responsabilidades
Equipo de desarrollo	Jorge García Ruiz	Realizar los incrementos del producto en cada sprint.
Product Owner	Jorge García Ruiz	Representar las necesidades de todo el que tiene interés en el proyecto y, por tanto, en el producto resultante.
Scrum Master	Jorge García Ruiz	Responsable del proceso Scrum, de su correcta implementación y la obtención del máximo beneficio de la misma.
Stakeholders	Yania Crespo Juan José Juste	Tienen interés en el resultado del proyecto, bien porque lo ha financiado, lo utilizará o será afectado por él.

Tabla 1. Roles y Responsabilidades

2.1. Caracterización de los riesgos.

Según el PM-BOOK: *"Un riesgo es un evento o una condición de inciertos que, si ocurren, tienen un efecto positivo o negativo sobre los objetivos del proyecto"* [3].

También conocemos las dos características principales de un riesgo:

- Incertidumbre: Es la probabilidad que hay de que ocurra el riesgo o no.
- Pérdida: Si el riesgo ocurre, las consecuencias que genera.

A continuación, vamos a establecer los tipos de riesgos que se pueden dar en este proyecto, con su consiguiente clasificación [3]:

- Riesgos de Proyecto: Afectan al plan de proyecto. Si estos suceden, pueden sufrir aumentos considerables tanto la planificación como los costes estimados al proyecto.
- Riesgos Técnicos: Amenazan a la calidad, a la planificación y por tanto a aspectos técnicos del desarrollo como la arquitectura, diseño y especificaciones.
- Riegos de Negocio: Afectan a la viabilidad del sistema o producto que se está desarrollando.

Seguidamente, vamos a establecer el rango de probabilidad de que ocurran los riesgos:

- Muy Alto: Cuando la probabilidad es mayor al 75%.
- Alto: La probabilidad está entre 50% y el 75%.
- Medio: La probabilidad está entre 25% y 50%.
- Bajo: Cuando la probabilidad es inferior al 25%.

De igual forma el rango de clasificación de pérdidas, si suceden los riesgos:

- Catastrófico: si se da, los objetivos del proyecto no se cumplirían.
- Crítico: si se produce, la calidad del producto final no sería la esperada.
- Marginal: si sucede, afecta a objetivos de poca importancia.
- Despreciable: si ocurre, no tiene ninguna repercusión, ya que son problemas menores.

Como estrategia frente a los riesgos que ocurran tendremos tres posibles opciones para afrontarlos:

- Impedir el riesgo.
- Transferir el riesgo, de forma que algo o alguien se ocupe del mismo.
- Aceptar el riesgo, en este caso encontramos dos situaciones:
  - Mitigarlo, tomando decisiones rápidas para reducir la probabilidad o impacto.
  - Definir un plan de contingencia, de forma que, si se da el riesgo tendremos definidos el plan de acciones o metodología a realizar para afrontarlo.

2.2. Definición de Riesgos.

A continuación, se detallan los riesgos, clasificados según la información que se muestra en el punto anterior, con su correspondiente plan de acción o contingencia en caso de que estos aparezcan [3].

Formulario de gestión de riesgos	
Identificador: R-01	**Título:** Fallo de planificación de las tareas
Tipo: Riesgo de Proyecto	**Probabilidad:** Media
Consecuencia: Catastrófica. No se podrá realizar la entrega final del proyecto en la fecha marcada.	
Valoración de riesgos	
Descripción: No se podrá entregar la versión final del proyecto en la fecha marcada debido a una planificación no adecuada**.**	
Contexto: Este riesgo puede darse a lo largo del proyecto, en la planificación de cada sprint.	
Análisis: Implicaría una replanificación del proyecto para conseguir que los retrasos de entregas no afecten a tareas posteriores.	
Planificación de Riesgos	
Estrategia: Prevención del riesgo	**Plan de acción del riesgo:** se realizará un seguimiento exhaustivo de los hitos marcados a lo largo del proyecto, además se dedicará el tiempo necesario para realizar la estimación inicial.

Tabla 2. Riesgo 01.

Formulario de gestión de riesgos	
Identificador: R-02	**Título:** Pérdida de información
Tipo: Riesgo de Proyecto	**Probabilidad:** Media
Consecuencia: Catastrófica.	
Valoración de riesgos	
Descripción: Pérdida de información en el proyecto**.**	
Contexto: Este riesgo puede darse a lo largo del proyecto.	
Análisis: Puede generar grandes retrasos y problemas en el proyecto.	
Planificación de Riesgos	
Estrategia: Prevención del riesgo	**Plan de acción del riesgo:** se realizarán copias de seguridad del código fuente de aplicación web que se ejecuta en la máquina virtual proporcionada por la Universidad, de la automatización del plan de pruebas y de los documentos generados.

Tabla 3. Riesgo02.

Formulario de gestión de riesgos	
Identificador: R-03	**Título:** Falta de experiencia en las herramientas a utilizar
Tipo: Riesgo Técnico	**Probabilidad:** Alta
Consecuencia: Crítico. El proyecto puede que no tenga la calidad esperada.	
Valoración de riesgos	
Descripción: Algunas de las herramientas a utilizar necesitan un lenguaje específico de programación en los casos de prueba.	
Contexto: Se puede producir en las iteraciones que corresponda la realización de dichos casos de prueba.	
Análisis: Puede tener una repercusión de retraso en el proyecto y con una calidad inferior a la esperada.	
Planificación de Riesgos	
Estrategia: Reducción del riesgo	**Plan de acción del riesgo:** Se tendrá en cuenta en la planificación, y se estimará de una forma más holgada para poder familiarizarse con las nuevas herramientas.

Tabla 4. Riesgo03.

Formulario de gestión de riesgos	
Identificador: R-04	**Título:** Falta de tiempo por trabajo en empresa
Tipo: Riesgo de Proyecto	**Probabilidad:** Alta
Consecuencia: Crítico. El proyecto puede que no tenga la calidad esperada.	
Valoración de riesgos	
Descripción: La dedicación de tiempo no es exclusiva al TFG, si no que se tiene que repartir con la jornada laboral en la empresa Everis.	
Contexto: Se puede producir a lo largo del proyecto.	
Análisis: Puede tener una repercusión de retraso en el proyecto y con una calidad inferior a la esperada.	
Planificación de Riesgos	
Estrategia: Reducción del riesgo	**Plan de acción del riesgo**: Se estimarán con holgura las tareas, teniendo en cuenta el horario laboral. A su vez, se intentará aprovechar el tiempo al máximo en la medida de lo posible.

Tabla 5. Riesgo04.

Formulario de gestión de riesgos	
Identificador: R-05	**Título:** Falta de comunicación con los tutores del proyecto
Tipo: Riesgo de Proyecto	**Probabilidad:** Baja
Consecuencia: Crítico. El proyecto puede que no tenga la calidad esperada.	
Valoración de riesgos	
Descripción: Este TFG consta de dos tutores: uno de la universidad y otro de la empresa Everis. La falta comunicación y disposición a la hora de resolver dudas puede dar lugar a un proyecto incompleto	
Contexto: Se puede producir a lo largo del proyecto.	
Análisis: Puede tener una repercusión de retraso en el proyecto y con una calidad inferior a la esperada.	
Planificación de Riesgos	
Estrategia: Protección del riesgo	**Plan de acción del riesgo:** Se ha establecido un protocolo de comunicación a través de correos electrónicos y reuniones pactadas.

Tabla 6. Riesgo05.

Formulario de gestión de riesgos	
Identificador: R-06	**Título:** Error en el análisis de los requisitos.
Tipo: Riesgo técnico.	**Probabilidad:** Media
Consecuencia: Crítico. El proyecto puede que no tenga la calidad esperada.	
Valoración de riesgos	
Descripción: Se realiza un análisis incompleto o erróneo de requisitos.	
Contexto: Se puede producir en la iteración donde haya dicha tarea.	
Análisis: Puede tener una repercusión de retraso en el proyecto y con una calidad inferior a la esperada.	
Planificación de Riesgos	
Estrategia: Protección del riesgo	**Plan de acción del riesgo:** Revisar y asegurarse que los requisitos son correctos antes de la realización del plan de pruebas**.**

Tabla 7.Riesgo06.

Formulario de gestión de riesgos	
Identificador: R-07	**Título:** Baja por Enfermedad
Tipo: Riesgo de Proyecto	**Probabilidad:** Baja
Consecuencia: Marginal. El proyecto puede que sufra algún retraso, pero sin repercusiones importantes	
Valoración de riesgos	
Descripción: Caer enfermo por alguna enfermedad.	
Contexto: Se puede producir a lo largo del proyecto	
Análisis: Puede tener una repercusión de retraso en el proyecto.	
Planificación de Riesgos	
Estrategia: Protección del riesgo	**Plan de acción del riesgo:** Evaluar el tiempo perdido, y en caso de que sea necesario replanificar.

Tabla 8. Riesgo07.

Formulario de gestión de riesgos	
Identificador: R-08	**Título:** Problemas con el software y versiones de terceros.
Tipo: Riesgo de Proyecto	**Probabilidad:** Media
Consecuencia: Crítico. El proyecto puede que no tenga la calidad esperada.	
Valoración de riesgos	
Descripción: El uso y descarga de los componentes de terceros puede que contenga fallos y no funcione correctamente.	
Contexto: Se puede producir a lo largo del proyecto	
Análisis: Puede tener una repercusión de retraso en el proyecto y con una calidad inferior a la esperada.	
Planificación de Riesgos	
Estrategia: Protección del riesgo	**Plan de acción del riesgo:** Se ha establecido descargar software de las páginas oficiales para evitar este tipo de problemas en la medida de lo posible.

Tabla 9.Riesgo08.

Formulario de gestión de riesgos	
Identificador: R-09	Título: Fallos de hardware y software en la máquina local.
Tipo: Riesgo de Proyecto	**Probabilidad:** Baja
Consecuencia: Crítico. El proyecto puede que no tenga la calidad esperada.	
Valoración de riesgos	
Descripción: La realización del TFG se realizará en el ordenador del alumno, puede que los componentes fallen.	
Contexto: Se puede producir a lo largo del proyecto.	
Análisis: Puede tener una repercusión de retraso en el proyecto y con una calidad inferior a la esperada.	
Planificación de Riesgos	
Estrategia: Protección del riesgo	**Plan de acción del riesgo:** Se realizará un estudio de las piezas o software dañado y se sustituirán los afectados.

Tabla 10.Riesgo09.

Formulario de gestión de riesgos	
Identificador: R-10	**Título:** Disponibilidad del Servidor
Tipo: Riesgo de Proyecto	**Probabilidad:** Baja
Consecuencia: Crítico. El proyecto puede que no tenga la calidad esperada.	
Valoración de riesgos	
Descripción: El servidor en el que se aloja la página web a probar se cae. Impidiendo la ejecución del plan de pruebas con las diferentes herramientas de automatización.	
Contexto: Se puede producir a lo largo del proyecto.	
Análisis: Puede tener una repercusión de retraso en el proyecto y con una calidad inferior a la esperada.	
Planificación de Riesgos	
Estrategia: Protección del riesgo	**Plan de acción del riesgo:** En caso de caerse se revisarán las causas y se tomarán medios adecuados para su solución sin que tengan una repercusión de tiempo elevada en el proyecto.

Tabla 11.Riesgo10.

2.3. Product Backlog

En la tabla 12, se muestra una lista priorizadas de todas las tareas, con sus respectivas horas hombre estimadas de esfuerzo en su realización, es, por tanto, el product backlog realizado para abordar cada una de las tareas correspondientes a este proyecto.

Tarea/Actividad	Duración Estimada
Descripción de la aplicación WEB	15
Planificación	20
Análisis de Requisitos, y estudio de la aplicación	20
Análisis de la aplicación y definición del plan de pruebas	25
Elaboración del Smoke - Test	10
Análisis de cada herramienta de automatización	15*3
Elaboración de los casos de prueba con el software definido de cada herramienta.	25*3
Ejecución del plan de pruebas.	10*3
Obtención de resultados de cada herramienta	10*3
Comparativa de herramientas	20
Conclusiones	15
Trabajo futuro	15
Estructura del documento	5
Elaboración de la documentación	40
Total	**365**

Tabla 12. Product Backlog.

En algunas tareas figura un factor multiplicativo ya que hay 3 herramientas de automatización, todas las unidades figuran en horas.

2.4. Metodología y Sprints.

En este apartado se va a detallar como trabajamos a lo largo de este proyecto. La realización de un trabajo fin de grado tiene una estimación de 300 horas de trabajo según la guía docente. Por lo que es un factor que tenemos que tener en cuenta a la hora de estimar cada una de las tareas de las que se compone el proyecto.

Partiendo de esta premisa, se ha estimado que la duración total del proyecto son 365 horas. Además, teniendo en cuenta la jornada laboral, que la dedicación máxima con la que se podrá contar de lunes a jueves será de dos horas diarias, el viernes se dedicarán tres horas y el fin de semana es cuando se dedicará más tiempo a la realización de dicho TFG, el sábado será un total de seis horas y el domingo cinco horas, sumando un total de veintidós horas semanales.

Esta metodología ágil tiene una realización de unas tareas en las que se tiene un resultado, un entregable después de cada realización de dicha tarea.

Nosotros en este proyecto hemos establecido realizar un sprint cada 15 días, o lo que es lo mismo 44 horas de trabajo, obteniendo un total de nueve sprints. Después de cada sprint, se realizará un Sprint Review donde se verán las tareas que estaban destinadas a entregarse, si en algún caso no se entrega alguna de ellas se revisará en el siguiente sprint, y así sucesivamente.

En la Figura 1, podemos ver la gráfica de Burndown charts, esto es el listado de las tareas pendientes representadas por el total de horas de duración de las mismas y el número de iteraciones de forma que se podrá ver en la gráfica una función descendente, que nos indicará el número de horas que se van empleando para la realización de cada una de las tareas indicadas en cada sprint. En el capítulo VII, podremos ver de forma más detallada las tareas que se van a realizar por cada sprint. En esta figura, solo se muestran ocho de los nueve sprints que se han estimado para la realización de este proyecto. Si nos fijamos en el gráfico, en el último sprint no se llegan a cumplir todas las horas, quedan trece horas, por esta razón se realiza otro sprint para completar las actividades pendientes al anterior sprint y como margen de tiempo que podremos tener en cuenta a lo largo del desarrollo del proyecto.

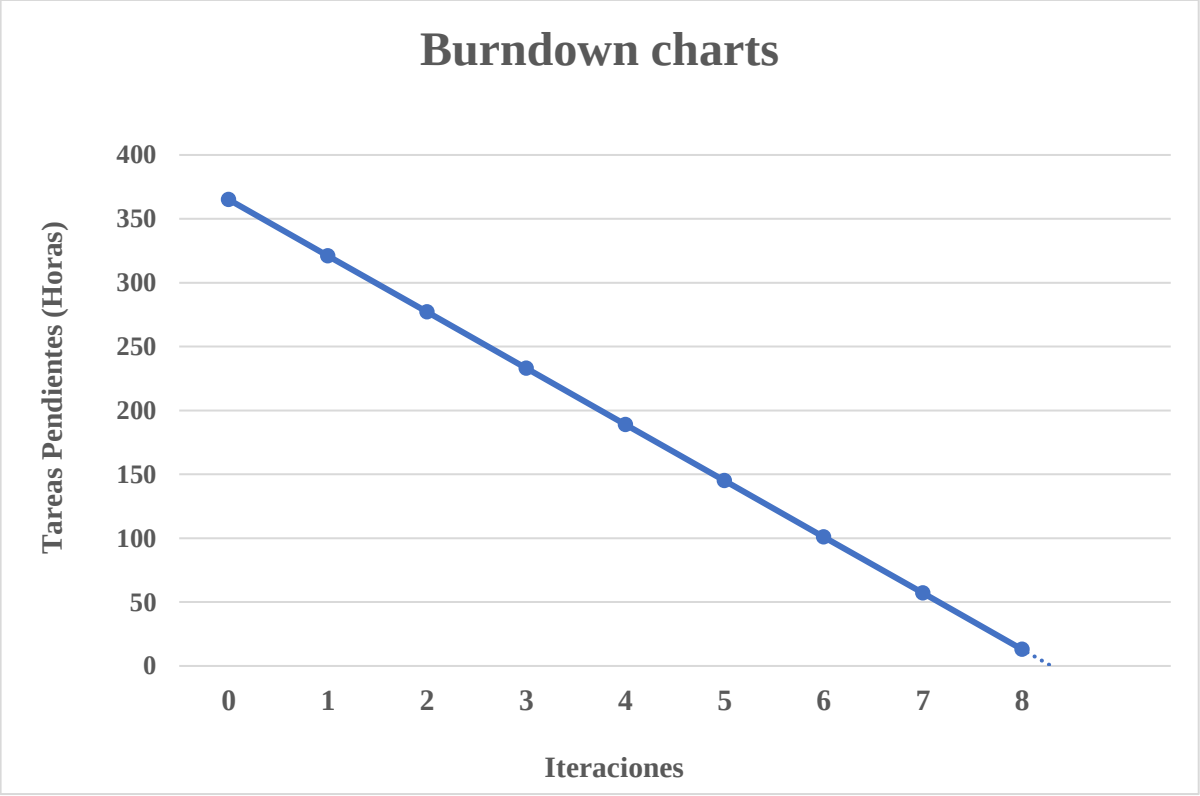

Figura 1. Burndown charts

En la tabla 13, en resumen, podemos ver la planificación estimada, fecha de inicio, fecha final de proyecto, numero de iteraciones y horas de dedicación en cada sprint.

Tiempos de Planificación	
Horas Estimadas de Duración del Proyecto: 365h	*N.º de Sprints: 9*
Duración de cada Sprint: 44h	
Fecha de Inicio del Proyecto: 30/01/2017	*Fecha de Finalización: 03/06/2017*

Tabla 13. Tiempos de Planificación

Capítulo III: Aplicación Web Objeto de las Pruebas

3.1.Descripción

La aplicación sobre la que vamos a trabajar y automatizar las pruebas está alojada en una máquina perteneciente al domino de aulas.inf.uva.es de la Universidad de Valladolid.

La dirección web de acceso es la: http://virtual.lab.inf.uva.es:28032.

Se trata de una máquina virtual otorgada en la asignatura de Planificación y Gestión de Plataformas Informáticas del primer cuatrimestre de cuarto curso.

La funcionalidad de esta página, es la de un gestor de incidencias cuyos requisitos se establecieron en la práctica de la asignatura antes citada.

Dentro de la plataforma encontraremos cuatro roles establecidos para la gestión de las mismas:

- Administrador. Este rol se encargará de dar de alta a los nuevos roles y proyectos en el sistema y de la administración de cada uno de ellos.

- Técnico. Se encargará de solucionar y revisar las incidencias que previamente un cliente generó y el supervisor validó. Podrá ver y generar un listado de las incidencias que tiene asignadas y las que hay en el sistema con el estado de cerradas por la organización.

- Supervisor. Este es el rol que será único en el sistema. Y podrá ver listados de todas las incidencias con el tipo de estado que hay definido en el sistema: aceptadas, rechazadas y cerradas de forma definitiva, esto es, las que fueron revisadas y cerradas por el supervisor.

- Cliente. Es el rol que detectará las incidencias y las dará de alta en el sistema. También podrá generar un listado con todas las incidencias abiertas.

A través de estos se irán registrando, tratando, aceptando, solucionando y rechazando cada una de las incidencias que se van dando de alta en el sistema.

A su vez los objetivos del proceso de gestión de incidencias en este sitio web son:

- Restablecer el funcionamiento normal tan pronto como sea posible.
- Minimizar el impacto negativo en el negocio.
- Garantizar que los niveles de calidad de servicio acordados se mantienen.

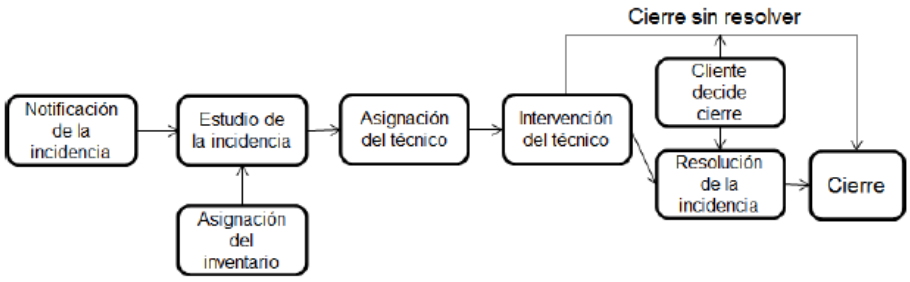

Figura 2.Ciclo de vida de una incidencia.

En la Figura 2, podemos contemplar el ciclo de vida de una incidencia dentro de la aplicación. Esta irá cambiando de estado según vaya pasando por los diferentes roles, que se detallaron anteriormente, y las acciones que estos lleven a cabo. Los estados posibles que puede tener una incidencia en esta plataforma son: solicitada, aceptada, rechazada, en curso, cerrada con y sin solución y cerrada de forma definitiva, esto es, que fue revisada y cerrada por el supervisor.

3.2. Análisis y Arquitectura

Desde el punto de vista técnico vamos a repasar una serie de características de dicha aplicación. Comenzaremos diciendo que está desarrollada en PHP y la base de datos está alojada en el servidor de la Universidad de Valladolid con dominio: jair.lab.inf.uva.es

Comenzaremos con la parte de la arquitectura, donde podemos ver en la Figura 3, correspondiente al diagrama de despliegue, la disposición física de cada uno de los elementos de la aplicación, esto lo tendremos que tener en cuenta cuando vayamos a automatizar las pruebas. Además, podemos ver que es una computación cliente-servidor de tres niveles, pudiendo comprobar como en la parte del servidor tendremos el servidor por un lado y por el otro el servidor de BBDD.

En la Figura 4, podremos observar el diagrama de entidad relación, de cada una de las tablas de las que consta nuestra aplicación, es interesante valorarlo ya que es donde se almacenará toda la información de nuestra aplicación.

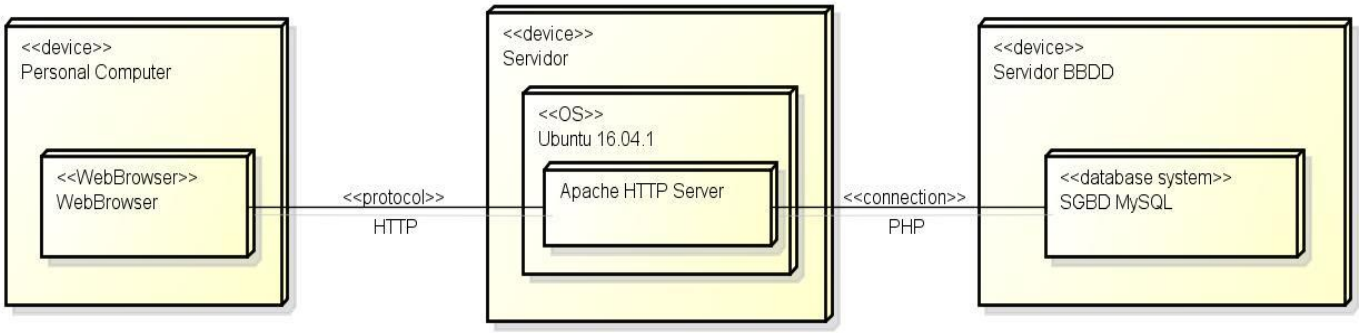

Figura 3. Diagrama de Despliegue

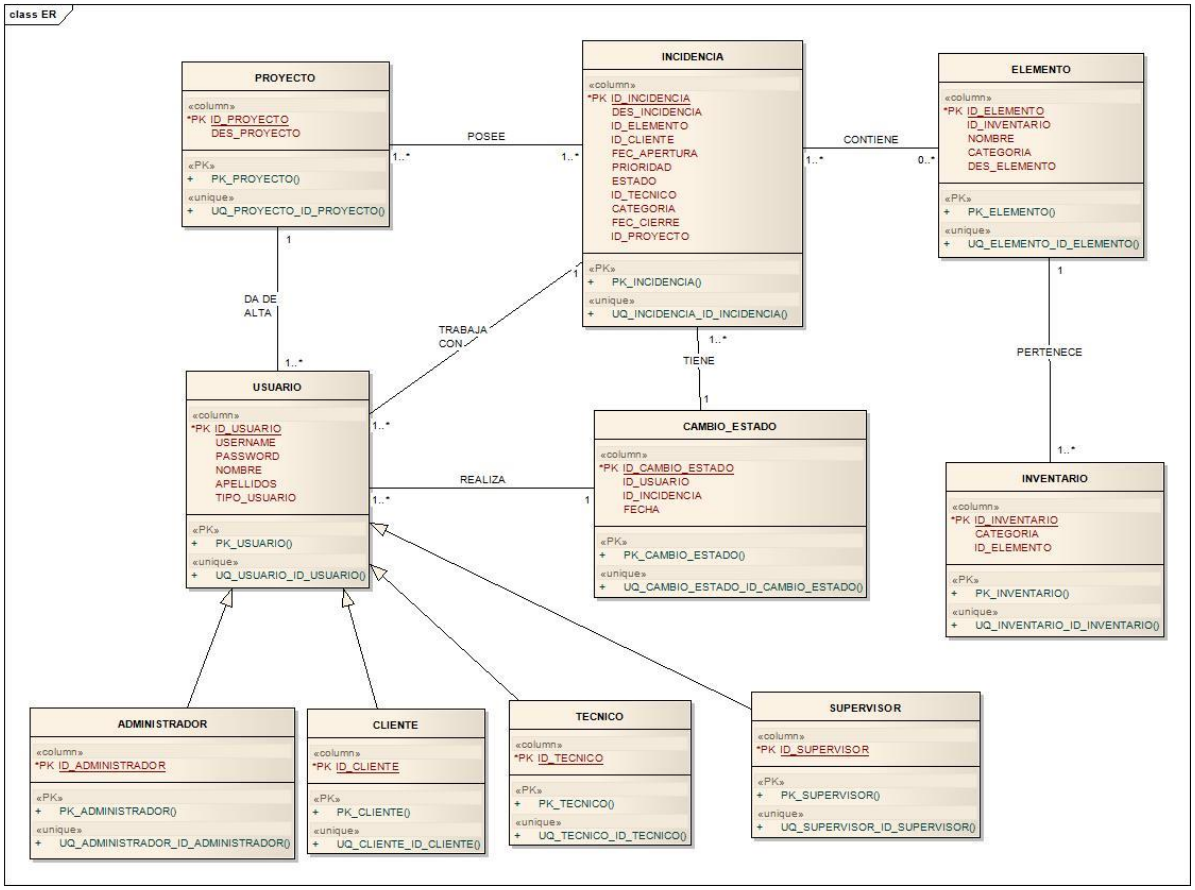

Figura 4. Diagrama Entidad Relación

Desde el punto de vista de análisis, se van describir cada uno de los casos de uso que tenemos en la aplicación web que nos servirán para la realización del plan de pruebas:

UC-01	Iniciar Sesión
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario inicia sesión en él
Actor	Usuario
Precondición	El actor Usuario tiene que estar registrado en el sistema
Secuencia Normal	1. El sistema pide al usuario que introduzca su Username y Password. 2. El actor Usuario introduce los datos con los que está registrado 3. El sistema comprueba los datos introducidos y permite el acceso al sistema
Excepciones	3.1 Si los datos son incorrectos el sistema notificará del error y el caso de uso continúa en el paso 2
Postcondición	El usuario está identificado y tiene acceso al sistema

Tabla 14.Caso de uso 001.

UC-002	Cerrar Sesión
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario cierra sesión
Actor	Usuario
Precondición	El usuario tiene que haber iniciado sesión en el sistema
Secuencia Normal	1. El usuario cierra sesión 2. El sistema comprueba la sesión de usuario y cierra la misma
Excepciones	
Postcondición	El usuario está registrado, pero no tiene acceso al sistema

Tabla 15. Caso de uso 002.

UC-03	Registrar Cliente
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Administrador registra un nuevo usuario de tipo Cliente en el sistema
Actor	Administrador
Precondición	El usuario tiene que estar registrado en el sistema y haber iniciado sesión en el mismo
Secuencia Normal	1. El actor Administrador selecciona la opción de añadir un nuevo usuario 2. El sistema carga los datos que tienen que ser introducidos para registrar un nuevo usuario 3. El actor Administrador introduce todos los datos en el sistema 4. El sistema registra al nuevo usuario de tipo cliente con los datos introducidos
Excepciones	4.1. Los datos introducidos son incorrectos, el sistema notificará del error y el caso de uso continúa en el paso 3
Postcondición	El usuario cliente se registra correctamente en el sistema

Tabla 16. Caso de uso 003

UC-04	Registrar Técnico
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando, El usuario Administrador registra un nuevo usuario de tipo Técnico en el sistema
Actor	Administrador
Precondición	El usuario tiene que estar registrado en el sistema y haber iniciado sesión en el mismo

Secuencia Normal	1. El actor Administrador selecciona la opción de añadir un nuevo usuario 2. El sistema carga los datos que tienen que ser introducidos para registrar un nuevo usuario 3. El actor Administrador introduce todos los datos en el sistema 4. El sistema registra al nuevo usuario de tipo técnico con los datos introducidos
Excepciones	4.1. Los datos introducidos son incorrectos, el sistema notificará del error y el caso de uso continúa en el paso 3
Postcondición	El usuario técnico se registra correctamente en el sistema

Tabla 17. Caso de uso 004.

UC-05	Registrar Supervisor
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario Administrador registra un nuevo usuario de tipo Supervisor en el sistema
Actor	Administrador
Precondición	El usuario tiene que estar registrado en el sistema y haber iniciado sesión en el mismo
Secuencia Normal	1. El actor Administrador selecciona la opción de añadir un nuevo usuario 2. El sistema carga los datos que tienen que ser introducidos para registrar un nuevo usuario 3. El actor Administrador introduce todos los datos en el sistema 4. El sistema registra al nuevo usuario de tipo supervisor con los datos introducidos
Excepciones	4.1. Los datos introducidos son incorrectos, el sistema notificará del error y el caso de uso continúa en el paso 3
Postcondición	El usuario supervisor se registra correctamente en el sistema

Tabla 18. Caso de uso 005.

UC-06	Alta Proyecto
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario Administrador da de alta un nuevo proyecto
Actor	Administrador
Precondición	El usuario tiene que estar registrado en el sistema y haber iniciado sesión en el mismo
Secuencia Normal	1. El actor Administrador selecciona la opción de dar de alta un nuevo proyecto 2. El sistema carga los datos iniciales del nuevo proyecto 3. El actor Administrador confirma los datos del nuevo proyecto 4. El sistema da de alta correctamente el nuevo proyecto
Excepciones	
Postcondición	El usuario está registrado, pero no tiene acceso al sistema

Tabla 19. Caso de uso 006.

UC-007	Asignar Técnico
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario Supervisor asigne un técnico a una incidencia generada
Actor	Supervisor
Precondición	El actor Supervisor tiene que estar registrado en el sistema y haber iniciado sesión en el mismo El actor Técnico tiene que estar dado de alta en el sistema La incidencia tiene que estar generada en el sistema La incidencia tiene que tener el estado de aceptada
Secuencia Normal	1. El sistema muestra las incidencias generadas al actor Supervisor 2. El actor Supervisor selecciona una incidencia 3. El actor Supervisor asigna un técnico a la incidencia generada 4. El sistema guarda la incidencia con el técnico que le ha sido asignado
Excepciones	
Postcondición	La incidencia generada tiene asignada un técnico

Tabla 20.Caso de uso 007.

UC-008	Analizar incidencia
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario Supervisor analiza una incidencia generada
Actor	Supervisor
Precondición	El actor Supervisor tiene que estar registrado en el sistema y haber iniciado sesión en el mismo La incidencia tiene que estar generada en el sistema
Secuencia Normal	1. El sistema muestra la nueva incidencia generada 2. El actor Supervisor selecciona dicha incidencia generada y la examina 3. El actor Supervisor acepta o rechaza la incidencia generada. 4. El sistema guarda el nuevo estado de la incidencia generada.
Excepciones	
Postcondición	La incidencia generada ha sido analizada y tiene un nuevo estado: aceptada o rechazada

Tabla 21. Caso de uso 008.

UC-009	Asignar elementos afectados
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario Supervisor asigna el o los elementos afectados a la incidencia
Actor	Supervisor
Precondición	El actor Supervisor tiene que estar registrado en el sistema y haber iniciado sesión en el mismo La incidencia tiene que estar generada en el sistema La incidencia tiene que tener el estado de aceptada El elemento o los elementos tienen que estar registrados en uno de los inventarios del sistema
Secuencia Normal	1. El sistema muestra elementos de un inventario por categoría 2. El actor Supervisor selecciona uno o varios elementos de dicho inventario 3. El sistema asigna el o los elementos a la incidencia y pide la confirmación de dicha asignación. 4. El actor Supervisor confirma la asignación 5. El sistema registra la asignación de el o los elementos afectados a la incidencia
Excepciones	4.1 El actor Supervisor no confirma la asignación y el caso de uso continúa en el paso 2
Postcondición	La incidencia generada tiene asignada un técnico

Tabla 22. Caso de uso 009.

UC-010	Generar incidencia
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario Cliente quiere generar una nueva incidencia
Actor	Cliente
Precondición	El actor Cliente tiene que estar registrado en el sistema
Secuencia Normal	1.El actor Cliente inicia sesión en el sistema 2. El actor Cliente elige la opción de generar Incidencia 3. El sistema le muestra por pantalla un formulario con los datos a rellenar 4. El actor Cliente rellena el formulario con los datos de su incidencia 5. El actor Cliente envía el formulario
Excepciones	4.1 Si al rellenar el formulario el sistema encuentra algún error lo notifica y se vuelve al paso 4
Postcondición	La incidencia se registra en el sistema

Tabla 23. Caso de uso 010.

UC-011	Aceptar incidencia
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor acepta una incidencia generada
Actor	Supervisor
Precondición	El actor Supervisor tiene que estar registrado en el sistema La incidencia tiene que estar registrada en el sistema
Secuencia Normal	1.El actor Supervisor inicia sesión en el sistema 2. El sistema muestra por pantalla las incidencias nuevas generadas 3. El Cliente Supervisor estudia la incidencia elegida y la acepta
Excepciones	
Postcondición	La incidencia cambia de estado a aceptada

Tabla 24. Caso de uso 011.

UC-012	Rechazar incidencia
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor rechaza una incidencia generada
Actor	Supervisor
Precondición	El actor Supervisor tiene que estar registrado en el sistema. La incidencia tiene que estar registrada en el sistema
Secuencia Normal	1.El actor Supervisor inicia sesión en el sistema 2. El sistema muestra por pantalla las incidencias nuevas generadas 3. El Cliente Supervisor estudia la incidencia elegida y la rechaza
Excepciones	
Postcondición	La incidencia cambia de estado a rechazada

Tabla 25. Caso de uso 012.

UC-013	Ver listado de incidencias por cliente
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor vea el listado de las incidencias solicitadas por un cliente.
Actor	Supervisor
Precondición	El actor Supervisor tiene que estar registrado en el sistema. El actor Cliente tiene que estar registrado en el sistema. Las incidencias tienen que estar registradas en el sistema.
Secuencia Normal	1. El actor Supervisor inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias. 3. El actor Supervisor selecciona que se muestren las incidencias de un cliente determinado.
Excepciones	
Postcondición	Se muestra un listado de las incidencias generadas por cliente

Tabla 26. Caso de uso 013.

UC-014	Ver listado todas las incidencias activas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor vea el listado de las incidencias que tienen el estado activa.
Actor	Supervisor
Precondición	El actor Supervisor tiene que estar registrado en el sistema. Las incidencias tienen que estar registradas en el sistema.
Secuencia Normal	1. El actor Supervisor inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias. 3. El actor Supervisor selecciona que se muestren las incidencias activas.
Excepciones	
Postcondición	Se muestra un listado de las incidencias activas

Tabla 27 Caso de uso 014.

UC-015	Ver listado de incidencias por técnico
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor vea el listado de las incidencias tratadas por un técnico.
Actor	Supervisor
Precondición	El actor supervisor tiene que estar registrado en el sistema. Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema.
Secuencia Normal	1. El actor Supervisor inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias. 3. El actor Supervisor selecciona que se muestren las incidencias de un técnico determinado.
Excepciones	
Postcondición	Se muestra un listado de incidencias por técnico

Tabla 28. Caso de uso 015.

UC-016	Ver listado de todas las incidencias cerradas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor vea el listado de las incidencias con el estado cerrada.
Actor	Supervisor, Técnico
Precondición	El actor supervisor tiene que estar registrado en el sistema. Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema. Las incidencias tienen que tener el estado cerrada.
Secuencia Normal	1. El actor Supervisor/Técnico inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias. 3. El actor Supervisor/Técnico selecciona que se muestren las incidencias con el estado cerrada.
Excepciones	
Postcondición	Se muestra un listado de todas las incidencias cerradas

Tabla 29. Caso de uso 016.

UC-017	Cerrar incidencia definitivamente
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor vea el listado de las incidencias con el estado cerrada.
Actor	Supervisor
Precondición	El actor supervisor tiene que estar registrado en el sistema. Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema. El actor Cliente tiene que estar registrado en el sistema. Las incidencias tienen que tener el estado cerrada.
Secuencia Normal	1. El actor Supervisor inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias. 3. El actor Supervisor selecciona que se muestren las incidencias con el estado cerrada (con o sin solución). 4. El actor Supervisor comprueba la incidencia y la cierra definitivamente.
Excepciones	
Postcondición	Se cierra definitivamente la incidencia

Tabla 30. Caso de uso 017.

UC-018	Ver listado de incidencias abiertas y asignadas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Supervisor vea el listado de las incidencias con el estado abierto y asignado.
Actor	Supervisor
Precondición	
Secuencia Normal	1. El actor Supervisor inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias. 3. El actor Supervisor selecciona que se muestren las incidencias con el estado abierta y asignada a un técnico.
Excepciones	
Postcondición	Se muestra un listado de las incidencias abiertas y asignadas

Tabla 31. Caso de uso 018.

UC-019	Introducir datos sobre la evolución de incidencias
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Técnico vaya introduciendo datos sobre la evolución de las incidencias que tiene asignadas.
Actor	Técnico
Precondición	Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema. Las incidencias tienen que tener el estado asignadas al técnico.
Secuencia Normal	1. El actor Técnico inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias que tiene asignadas. 3. El actor Técnico irá añadiendo información sobre la incidencia.
Excepciones	
Postcondición	Se introducen nuevos datos sobre la evolución de las incidencias

Tabla 32. Caso de uso 019.

UC-020	Introducir comentarios
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Técnico vaya introduciendo comentarios sobre las incidencias que tiene asignadas.
Actor	Técnico
Precondición	Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema. Las incidencias tienen que tener el estado asignadas al técnico.
Secuencia Normal	1. El actor Técnico inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias que tiene asignadas. 3. El actor Técnico introducirá comentarios sobre la incidencia.
Excepciones	
Postcondición	Se introducen comentarios en la incidencia

Tabla 33. Caso de uso 020.

UC-021	Cambiar estado de incidencia
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Técnico cambie el estado de las incidencias que tiene asignadas.
Actor	Técnico
Precondición	Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema. Las incidencias tienen que tener el estado asignadas al técnico.
Secuencia Normal	1. El actor Técnico inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias que tiene asignadas. 3. El actor Técnico cambiará de estado de la incidencia.
Excepciones	
Postcondición	Se cambia el estado de la incidencia

Tabla 34. Caso de uso 021.

UC-0022	Cerrar incidencia con solución
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Cliente/Técnico cierra una incidencia que tiene solución
Actor	Cliente, Técnico
Precondición	El actor Cliente tiene que estar registrado en el sistema. Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema.
Secuencia Normal	1. El actor Cliente/Técnico inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias 3. El actor Cliente/Técnico selecciona la incidencia que quiere resolver 4. El actor Cliente/Técnico selecciona la opción cerrar con solución
Excepciones	
Postcondición	La incidencia cambia de estado a cerrada con solución

Tabla 35. Caso de uso 022.

UC-0023	Cerrar incidencia sin solución
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Cliente/Técnico cierra una incidencia que no tiene solución
Actor	Cliente, Técnico
Precondición	El actor Cliente tiene que estar registrado en el sistema. Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema.
Secuencia Normal	1. El actor Cliente/Técnico inicia sesión en el sistema. 2. El sistema muestra por pantalla las incidencias 3. El actor Cliente/Técnico selecciona la incidencia que quiere resolver 4. El actor Cliente/Técnico selecciona la opción cerrar sin solución
Excepciones	
Postcondición	La incidencia cambia de estado a cerrada sin solución

Tabla 36. Caso de uso 023.

UC-024	Ver listado de incidencias generadas y abiertas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Cliente vea las incidencias que ha generado y están abiertas
Actor	Cliente
Precondición	El actor Cliente tiene que estar registrado en el sistema. La incidencia tiene que estar generada en el sistema
Secuencia Normal	1. El actor Cliente inicia sesión en el sistema. 2. El actor Cliente elige la opción de mostrar listado de incidencias 3. El sistema muestra por pantalla las incidencias. 4. El actor cliente selecciona que se muestren las incidencias con el estado abierta
Excepciones	
Postcondición	Se visualizará el listado generado por pantalla.

Tabla 37. Caso de uso 024.

UC-025	Ver listado de incidencias generadas y cerradas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Cliente vea las incidencias que ha generado y están cerradas
Actor	Cliente
Precondición	El actor Cliente tiene que estar registrado en el sistema. La incidencia tiene que estar generada en el sistema
Secuencia Normal	1. El actor Cliente inicia sesión en el sistema. 2. El actor Cliente elige la opción de mostrar listado de incidencias 3. El sistema muestra por pantalla las incidencias. 4. El actor cliente selecciona que se muestren las incidencias con el estado cerrada
Excepciones	
Postcondición	Visualizar por pantalla el listado con las características generadas

Tabla 38. Caso de uso 025.

UC-026	Consultar información de una incidencia
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Cliente/Técnico/Supervisor quiere mostrar la información sobre una incidencia
Actor	Cliente, Técnico, Supervisor
Precondición	El actor Cliente tiene que estar registrado en el sistema. El actor Supervisor tiene que estar registrado en el sistema Las incidencias tienen que estar registradas en el sistema. El actor Técnico tiene que estar registrado en el sistema.
Secuencia Normal	1. El actor Cliente/Técnico/Supervisor inicia sesión en el sistema. 2.El actor Cliente/Técnico/Supervisor elige la opción de ver listado de incidencias 3. El sistema muestra por pantalla las incidencias 4. El actor Cliente/Técnico/Supervisor selecciona la incidencia 5. El actor Cliente/Técnico selecciona la opción mostrar la información 6. El sistema muestra la información de la incidencia
Excepciones	
Postcondición	Se mostrará información relacionada con la incidencia por pantalla.

Tabla 39. Caso de uso 026.

UC-027	Eliminar usuario.
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Administrador quiere mostrar la información sobre una incidencia
Actor	Administrador
Precondición	El actor Administrador debe estar dado de alta en el sistema.
Secuencia Normal	1. El actor administrador inicia sesión en el sistema. 2. El actor administrador elige la opción “eliminar” de listado de usuarios. 3. En la lista resultante desaparece el usuario eliminado.
Excepciones	
Postcondición	Se elimina el usuario correctamente.

Tabla 40.Caso de uso 027.

En la Figura 5, se puede apreciar el diagrama de casos de uso de la aplicación, los casos de uso coloreados en azul, son los que se implementarán en el plan de pruebas acotado, ya que con su funcionalidad se cubre el núcleo de la aplicación.

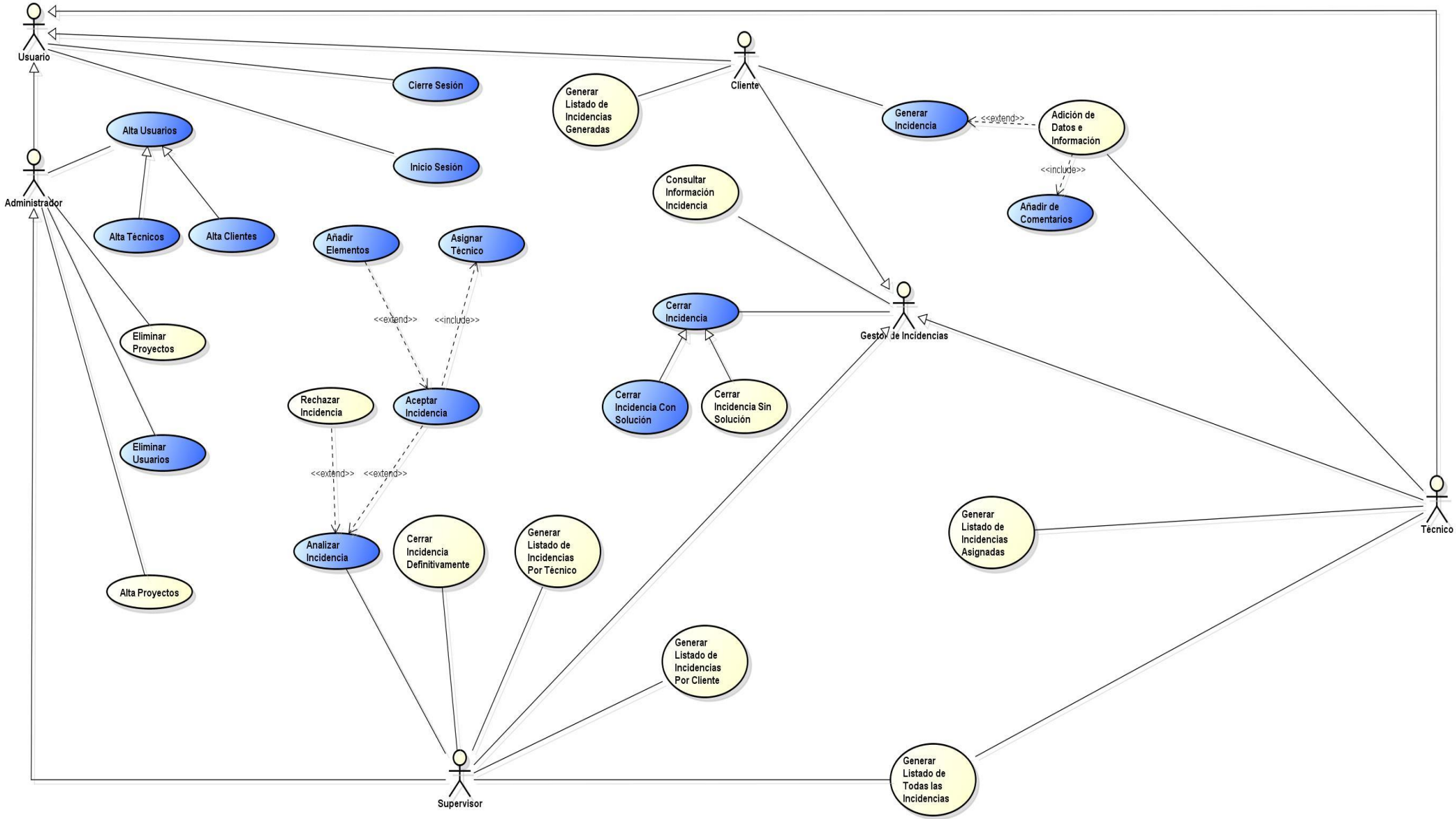

Figura 5. Diagrama de Casos de Uso.

Capítulo IV: Entorno de Pruebas

A lo largo de este capítulo definiremos el entorno de estudio, el plan de pruebas que se va ejecutar en las diferentes herramientas de automatización y una comparativa de las herramientas a utilizar.

Antes de comenzar con esta sección, vamos a explicar una serie de conceptos que harán que entendamos mejor cada uno de los apartados y capítulos posteriores.

Las pruebas del software consisten en la validación o confirmación del comportamiento de un programa *dinámicamente* a través de un grupo *finito* de casos de prueba, debidamente seleccionados del, típicamente, ámbito de ejecuciones infinito, en relación al *comportamiento esperado* [5]. Caben destacar los términos marcados en cursiva:

Dinámicamente: en este contexto significa que, para la realización de las pruebas es necesario ejecutar el programa para los datos de entrada.

Finito: hay que establecer un límite en el conjunto de pruebas a realizar, ya que la realización de pruebas en software complejos puede tender a infinitos casos de prueba, por tanto, hay que establecer un equilibrio ya que los recursos y el tiempo son limitados.

Comportamiento esperado: debe ser posible decidir cuando la salida observada de la ejecución del programa es aceptable o no, de otra forma el esfuerzo de la prueba es inútil. El comportamiento observado puede ser revisado contra las expectativas del usuario, contra una especificación o contra el comportamiento anticipado por requerimientos implícitos o expectativas razonables.

Por tanto, a la hora de realizar de un plan de pruebas debemos distinguir los tipos de pruebas que hay, ya que deberemos adaptarlo a las necesidades que tengamos en ese momento y la funcionalidad que queramos probar. No olvidemos que el objetivo principal de ésta, es revelar el mayor número posible de fallos, por lo que tendremos que elegir el tipo o tipos de prueba que intenten "romper" nuestro desarrollo, de esta forma pretenderemos conseguir un sistema que cumpla las expectativas del cliente para un "propósito marcado". Es decir, que el producto resultante sea aceptable para los objetivos para los que se planteó [7].

La técnica de análisis y pruebas al que se somete a un software, desarrollado o en proceso de desarrollo, para comprobar que cumple con la especificación y funcionalidad marcada por el cliente es conocido como el proceso de Verificación y Validación (V&V) [6].

La Verificación: consiste en la comprobación de que estamos construyendo el producto de acuerdo con su determinación. Esto es, que cumple todos los requisitos funcionales y no funcionales establecidos por el cliente.

La Validación: consiste en la comprobación de que el producto satisface la especificación marcada por el cliente que describe un proceso más general, debido a que en los documentos de requisitos de software no siempre reflejan los deseos o necesidades del cliente.

Dentro de este proceso, tenemos dos tipos procedimientos complementarios [6]:

- Las inspecciones software: se ocupan de validar la documentación, diagramas y código fuente del software. Se pueden realizar en cualquier etapa del desarrollo y son estáticas ya que no es necesaria la ejecución de código.

- Pruebas de software: como ya se definió anteriormente, es un proceso que se encarga de verificar y validar la ejecución del código teniendo en cuenta unas entradas marcadas determinadas, junto con las salidas de esa ejecución y el resultado esperado.

En la Figura 6, se muestra, de forma gráfica, como las inspecciones y las pruebas software son procedimientos complementarios en el proceso de desarrollo de software.

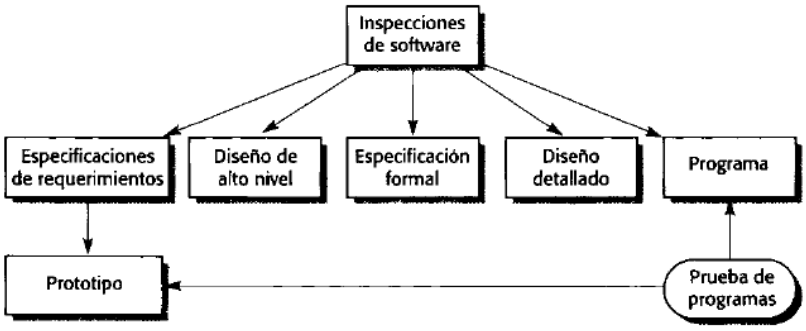

Figura 6. Verificación y Validación. Estáticos y Dinámicos [6].

Teniendo en cuenta todos estos conceptos y procesos de realización de pruebas podemos hacer una clasificación con los diferentes tipos de pruebas.

La clasificación la haremos dependiendo del nivel dentro de la aplicación, por tanto, las pruebas se organizan en [1] [6]:

Pruebas Unitarias: con su ayuda podremos verificar el funcionamiento de un módulo concreto de la aplicación. Para obtener buenos casos de prueba unitarios se usan los diagramas de estado o grafos de flujo.

Pruebas de Integración: con su ayuda verificaremos la interacción entre los componentes que forman la aplicación. Pueden estar guiadas la funcionalidad de cada uno de los componentes o integradas. En este último caso hay dos subgrupos: ascendente(Bottom-Up) y descendentes (Top-Down).

Dentro este tipo de pruebas, también encontramos las Pruebas de Regresión, estas pruebas se tienen en cuenta cuando hay integración continua, ya que este tipo de pruebas no es necesario que sean realizadas, si no que pueden ser pruebas que ya hayamos realizado en otros momentos, y por tanto, en estos casos, las utilizaríamos nuevamente para comprobar que las últimas modificaciones que hemos realizado en nuestro código no han modificado la funcionalidad o han "roto" parte de código que anteriormente funcionaba de forma correcta.

En la Figura 7. Vemos representado un árbol donde podríamos establecer las pruebas que tendríamos que realizar para verificar el funcionamiento de nuestra aplicación. Las hojas corresponderían con las pruebas unitarias, después cada uno de los nodos siguientes corresponde con las pruebas de integración entre el nodo anterior y el superior. La forma de realizarlo es la que nos crea la bifurcación entre las pruebas Bottom-Up y Top-Down. Si realizamos Bottom-Up, se realizan, primeramente, las pruebas unitarias, como se comentó anteriormente, en este caso D, E, después tenemos las pruebas de integración con los nodos superiores, por tanto, las pruebas serían B con D y C con E y por último A con B y A con C.

Si por el contrario realizamos un Top-Down, tenemos que realizar una integración en profundidad, A, B, D, C, E y seguidamente en anchura, A, B, C, D, E.

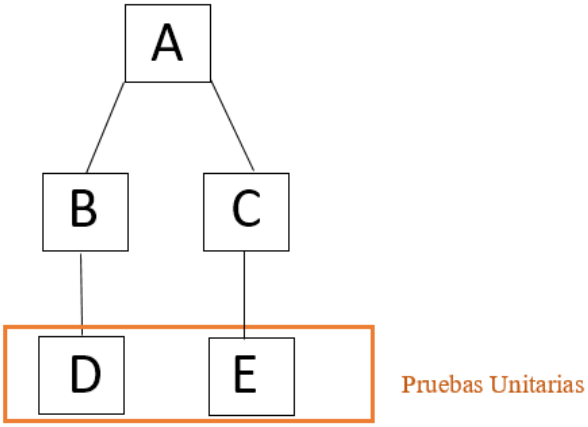

Figura 7. Ejemplo de Pruebas de Integración.

Tanto en las pruebas unitarias como en las pruebas de integración se verificará que se cumple con las especificaciones del cliente, esto es, con los requisitos definidos.

Atendiendo a esto, también se realizan los dos tipos de pruebas: de caja negra y las de caja blanca.

Las pruebas de caja negra es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software.

En las pruebas de caja negra, nos centramos solamente en las entradas y salidas del sistema, sin preocuparnos de la estructura interna del programa.

Para obtener el detalle de cuáles deben ser esas entradas y salidas, nos basamos únicamente en los requerimientos de software y especificaciones funcionales.

Sin embargo las pruebas de caja blanca, se basan en la estructura interna de la aplicación, de forma que se comprueba que cada una de las funcionalidades implementadas se cumplen correctamente y que son alcanzables cada uno de los posibles resultados que se obtienen, en base a cada una entradas determinadas, son "lógicos", esto es, que hemos contemplado que iba a ser así el comportamiento de la aplicación en estos casos, contemplándose en la implementación del código de cada uno de los módulos que componen nuestra aplicación.

Pruebas de Sistema: con este tipo de pruebas, vamos a comprobar la operativa de funcionamiento de nuestro sistema en su totalidad. En este tipo de pruebas se comprobarán los requisitos no funcionales: velocidad, seguridad, fiabilidad, rendimiento, portabilidad, etcétera.

Pruebas de Aceptación: validan el comportamiento del sistema según las especificaciones del cliente. Obteniendo como resultado exitoso que el usuario acepte el producto desarrollado.

En este tipo último tipo de pruebas encontramos a su vez, dos tipos:

- Pruebas Alfa: Se realizan con el cliente y el desarrollador o tester como observador. Pretendiendo así realizar una simulación en un entorno de producción.

- Pruebas Beta: Se realizan en el entorno de cliente sin ninguna observación.

Una vez que se somete a una aplicación a los diferentes tipos de prueba que pueden realizarse, pueden obtenerse errores.

El proceso de depuración de errores es una tarea que es simple, ya que cuando obtenemos un fallo, no tiene por qué estar cerca de donde nos falló el programa o prueba en cuestión.

De hecho, el objetivo de la realización de las pruebas es la obtención de errores. Estos errores deben depurarse.

En la Figura 8, se muestra el proceso detección, depuración y corrección de errores. Todo parte de la detección de un error o anomalía en los resultados de las pruebas realizados, a partir de aquí tendremos que localizar el error, diseñar la reparación, repararlo y nuevamente volver a realizar pruebas en el programa. Cabe destacar que este proceso es cíclico, de forma que se realizará hasta que todos los defectos sean solucionados y, por tanto, después de la ejecución de las pruebas marcadas sea exitoso.

El proceso de depuración puede llevar al diseño de nuevos casos de prueba como parte de la formulación de la hipótesis en busca de la causa.

Las herramientas utilizadas en este proceso, suelen encontrarse en los IDEs *(Integrated Development Environment*) utilizados para realizar la programación de nuestra aplicación. Estas tienen una operativa de ejecución "paso a paso". De forma que, permiten al desarrollador acotar la zona del error y poder detectar donde se produce el fallo de una forma más fácil.

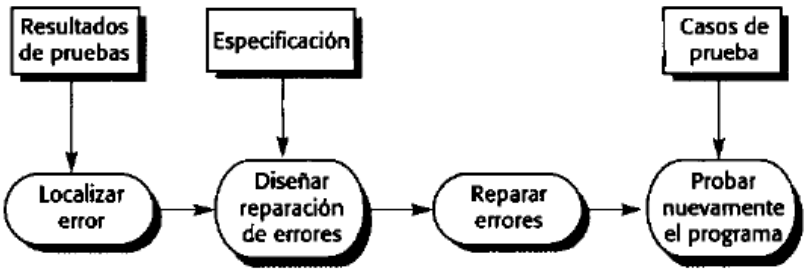

Figura 8. Proceso de Depuración [6].

En la Figura 8, podemos ver la pirámide de las pruebas automatizadas. Es un contexto desarrollado por Mike Cohn en el libro publicado Succeeding with Agile. Software [2]. Ahí se plantea esta pirámide, que como podemos observar lo que nos indica es que en la base se encuentran los tipos de pruebas que menos coste y más número de pruebas se realizan, que son las pruebas unitarias. De igual forma, en el vértice o en la parte más alta, encontramos las pruebas que menos se realizan, que son las pruebas beta, ya que es necesario montar un entorno de producción y reunirse con el cliente y tester para poder comprobar funcionalmente si se adapta a las especificaciones del cliente, por esta razón, son las que más coste tienen. También la posición de la pirámide nos indica que en la parte baja o en la base vamos a tener más ROI *(Return On Investment)*, esto es, donde más partido o beneficio vamos a sacar en base a la inversión tanto temporal o económica realizada, de forma análoga, en la parte más alta o vértice nos encontraremos con el caso opuesto al ROI de la base.

También, dentro de la pirámide vemos una clasificación de qué tipo de pruebas se realizan por categoría, esto es, las pruebas de aceptación, sistema, integración, de componentes y unitarias, se encargarán los desarrolladores y las pruebas de validación, las Alpha y Beta, se encargará el equipo de QA y los desarrolladores.

Como conclusión al ver esta pirámide, es que conviene hacer una valoración y un balance inicial de las pruebas que tenemos que hacer en la aplicación que queramos probar, teniendo en cuenta el coste, tanto de tiempo como económico.

También nos muestra el caso ideal de la automatización de las pruebas, ya que si seguimos ese patrón realizaremos una comprobación de la funcionalidad de la aplicación completa y desde el mejor punto de vista de coste para el proyecto.

Por esta razón, si no aplicamos este patrón, nos encontraremos con el "cono de helado" o anti-patrón, que podemos verlo en la Figura 10. En él se puede apreciar que donde más tiempo y recursos se emplean en las pruebas manuales. Este es el caso típico que nos encontramos cuando no se realiza la automatización de las pruebas de un software. De esta forma, invertiremos mucho tiempo en realizar este tipo de pruebas y no nos servirán para garantizar una funcionalidad correcta de toda la aplicación.

En los diferentes subapartados de este capítulo trataremos todo lo que se ha especificado.

En primer lugar, se hará un análisis y estudio del plan de pruebas, donde a través de los casos de uso se definirán todos los casos de prueba que hemos tratado para que se pueda probar la aplicación.

Seguidamente, se hará un Smoke-Test, esto es, un plan de pruebas acotado en el que se implementarán todos aquellos casos de uso forma parte de la funcionalidad del núcleo de la aplicación, probando así la correcta ejecución del núcleo de nuestra aplicación.

Por último, se especificarán los requisitos que se han tenido en cuenta, a la hora de seleccionar cada una de las herramientas que hay en el mercado.

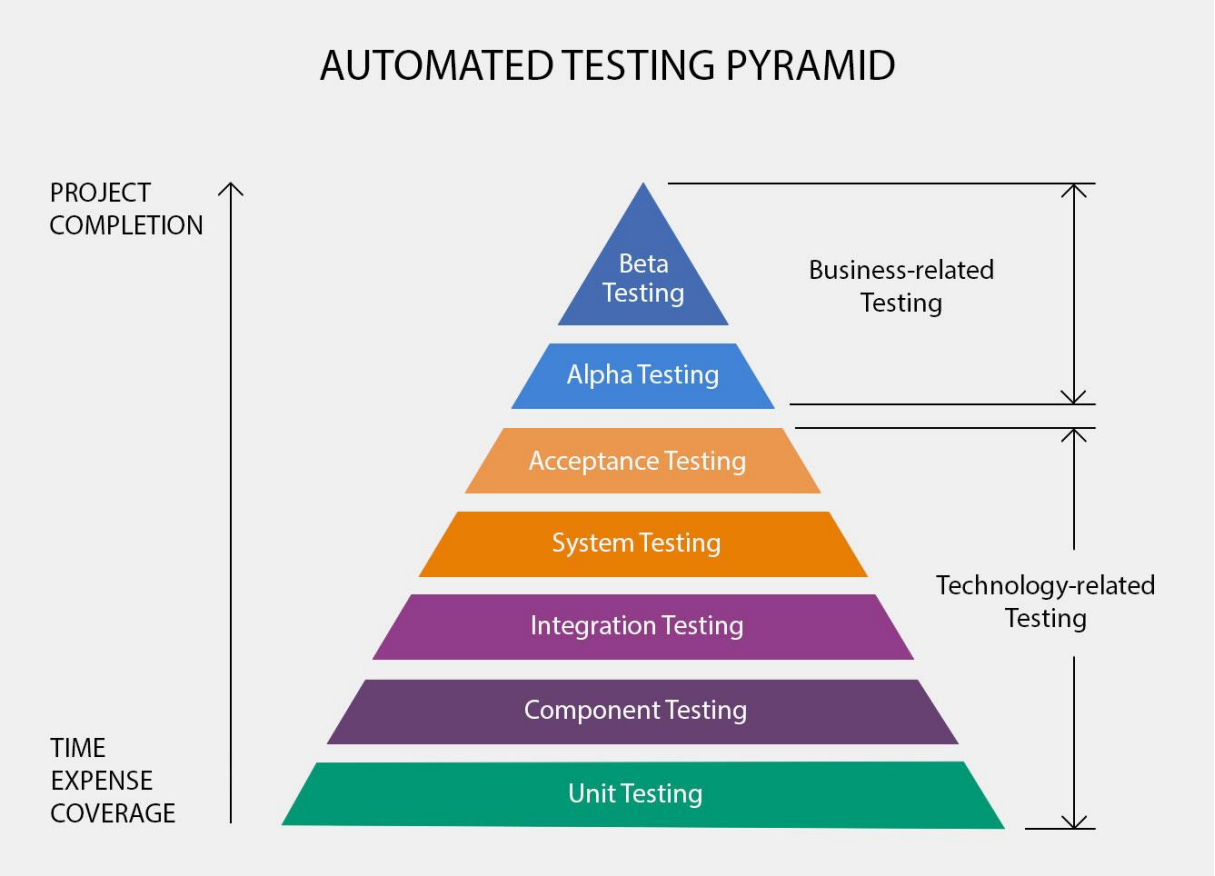

Figura 9. Pirámide de Automatización de Pruebas [14]

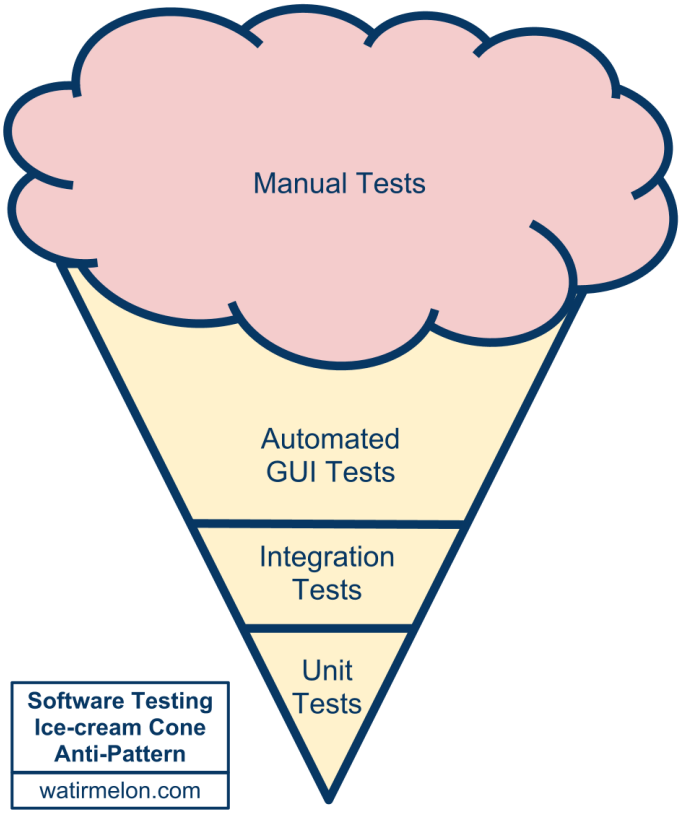

Figura 10. Cono de helado pruebas Software [13]

4.1. Análisis y Estudio del Plan de Pruebas

Tal y como se explicó con anterioridad en este documento, vamos a definir el plan de pruebas de la aplicación.

Hay que tener en cuenta que no se entrará en detalle riguroso, es decir no se probarán todas las funcionalidades de cada uno de las opciones de los filtros de generación de listados, ni todas las páginas ni funcionalidades del sistema.

Antes de nada, vamos a reflexionar sobre qué tipo de pruebas vamos a realizar.

Atendiendo a las definiciones y especificaciones que hemos detallado anteriormente lo que vamos a realizar son pruebas de aceptación funcionales [9].

La decisión de optar por este tipo de pruebas, es porque éstas cumplen con lo que realmente se quiere probar en esta aplicación, esta aplicación web ya está construida y queremos probar las especificaciones que marca un cliente, y a su vez, son funcionales ya que con el plan de pruebas que se va a realizar se va a comprobar si dicha página cumple las especificaciones marcadas y, por tanto, la funcionalidad de la misma.

También este plan de pruebas nos podría servir como pruebas de regresión, ya que, en caso que deseemos añadir un nuevo módulo, con más operativas de gestión de incidencias o simplemente un cambio de funcionalidad, con este plan de pruebas nos serviría para verificar que la aplicación sigue funcionando después de las modificaciones o si, por el contrario, se ha "roto" algo y debe ser corregido.

Los requisitos de la aplicación se han descrito y analizado en el Capítulo anterior en términos de la descripción de los casos de uso y el modelo conceptual.

Por tanto, para definir el plan de pruebas, vamos a realizar un análisis de requisitos de la aplicación, de esta forma obtendremos las especificaciones del cliente en esta aplicación y podremos realizar un plan de pruebas acorde al tipo de pruebas que deseamos.

En la Figura 5, los casos de uso que figuran en azul, son aquellos que se van a incluir en nuestro plan de pruebas acotado (Smoke-Test) mientras que el conjunto de todos ellos correspondería con el plan de pruebas general de la aplicación.

Por tanto, el proceso de negocio se muestra en el diagrama de actividad de la Figura 11.

Este núcleo del proceso del negocio se realiza en la aplicación mediante un subconjunto de todos los casos de uso, y será la base para el plan de pruebas

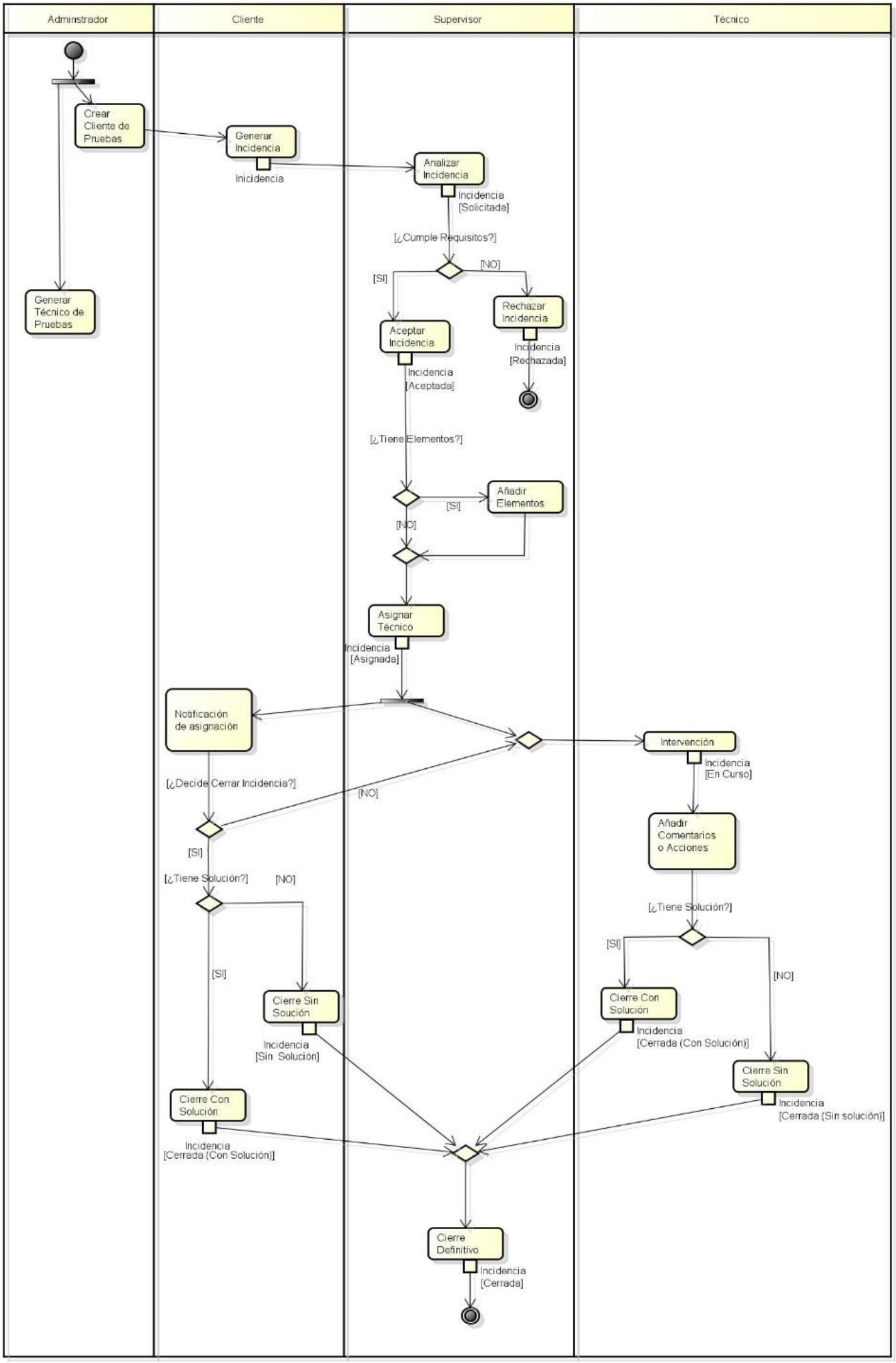

Figura 11. Diagrama de Actividad núcleo de aplicación.

Analizados todos los casos de uso vamos a definir el plan de pruebas, basado en los anteriores casos de uso de la aplicación, separado por cada uno de los roles/actores del sistema, de esta forma se verá de una forma más clara, los casos de uso involucrados con cada rol y con la numeración que se le ha otorgado:

Código Bloque:		**Descripción Bloque:**		
ADMIN		*Casos de Prueba para el Rol Administrador*		
Código Caso Prueba	**Título Caso Prueba**	**Descripción Caso Prueba**	**Criticidad**	**Código Caso Uso**
PAF - 001	**Inicio Sesión**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario introduce su login y password de forma correcta. **Resultado**: El usuario accede a la página principal de la aplicación.	ALTA	**UC-001**
PAF - 002	**Alta Usuario**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: 1. El usuario pulsará en el botón para añadir nuevos usuarios. 2. Completará cada uno de los campos del formulario correctamente. **Resultado**: El usuario será persistente en el sistema y se le podrá visualizar en la tabla de los usuarios de la aplicación.	ALTA	**UC-003** **UC-004** **UC-005**

PAF - 003	**Alta Proyecto**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: 1. El usuario pulsará en el botón para añadir nuevos proyectos. 2. Completará cada uno de los campos del formulario correctamente. **Resultado**: El proyecto será persistente en el sistema y se le podrá visualizar en la tabla de los usuarios de la aplicación.	MEDIA	**UC-006**
PAF - 004	**Eliminar Usuarios**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: El usuario elegirá el usuario en cuestión, y pulsará en el botón de eliminar. **Resultado**: El usuario se borrará correctamente de la aplicación.	MEDIA	**UC-027**
PAF - 005	**Cerrar Sesión**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: El usuario saldrá de la aplicación gracias al botón de cierre de sesión. **Resultado**: El usuario saldrá de la aplicación y le redirigirá a la pantalla inicial de login.	ALTA	**UC-002**

Tabla 41. Plan de pruebas. Rol Administrador

Código Bloque:		Descripción Bloque:		
SUPER		*Casos de Prueba para el Rol Supervisor*		
Código Caso Prueba	**Título Caso Prueba**	**Descripción Caso Prueba**	**Criticidad**	**Código Caso Uso**
PAF - 007	**Inicio Sesión**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario introduce su login y password de forma correcta. **Resultado**: El usuario accede a la página principal de la aplicación.	ALTA	**UC-001**
PAF - 008	**Analizar Incidencia**	**Precondiciones**: El usuario debe estar registrado en el sistema, estar en la página principal de la aplicación. **Procedimiento**: El usuario visualizará las incidencias pendientes y revisará las características de la misma, para comprobar si cumplen con los requisitos marcados por la plataforma de gestión. **Resultado**: El usuario tendrá que decidir si aceptar o rechazar la incidencia, cuyo estado actual es Solicitada.	ALTA	**UC-008**
PAF - 009	**Aceptar Incidencia**	**Precondiciones**: El usuario debe estar registrado en el sistema, en la página principal y haber analizado la incidencia. **Procedimiento**: El usuario verificará que cumple las especificaciones marcadas por la aplicación y aceptará la incidencia. **Resultado**: La incidencia pasa a estado de Aceptada.	ALTA	**UC-011**
PAF - 010	**Rechazar Incidencia**	**Precondiciones**: El usuario debe estar registrado en el sistema, en la página principal y haber analizado la incidencia. **Procedimiento**: El usuario verificará que no cumple las especificaciones marcadas por la aplicación y aceptará la incidencia. **Resultado**: La incidencia pasa a estado de Rechazada.	ALTA	**UC-012**

PAF - 011	**Añadir Elementos**	**Precondiciones**: El usuario debe estar registrado en el sistema, en la página principal y haber analizado y aceptado la incidencia. **Procedimiento**: El usuario asignará los elementos asociados a la incidencia, dependiendo de la categoría de la misma. **Resultado**: Los elementos asignados figurarán dentro de la incidencia.	ALTA	**UC-009**
PAF – 012	**Asignar a un Técnico**	**Precondiciones**: El usuario debe estar registrado en el sistema, en la página principal y haber analizado y aceptado la incidencia. **Procedimiento**: El usuario seleccionará un técnico de los que están registrados en el sistema. **Resultado**: La incidencia cambia de estado a asignada. El técnico recibirá una incidencia.	ALTA	**UC-007**
PAF – 013	**Ver listado de incidencias por cliente**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario podrá generar un listado con las incidencias que ha generado un cliente con los diferentes estados posibles de la aplicación **Resultado**: Se observará el listado del cliente seleccionado, independientemente del estado que éstas tengan.	ALTA	**UC-013**

PAF – 014	**Ver listado de incidencias por Técnico**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario generará un listado con todas las incidencias que han sido asignadas a un técnico, con cada uno de los posibles estados. **Resultado**: Se observará el listado del técnico seleccionado, independientemente del estado que éstas tengan.	ALTA	**UC-015**
PAF – 015	**Ver todas las incidencias**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario generará un listado de todas las incidencias que hay en la aplicación independientemente del estado que tengan, el cliente que las haya registrado y el técnico que las haya tratado **Resultado**: Se observará el listado con todas las incidencias que se han registrado en el sistema.	ALTA	**UC-006**
PAF – 016	**Cerrar Definitivamente**	**Precondiciones**: El usuario debe estar registrado en el sistema. La incidencia debe estar: Dada de alta por un cliente. Aceptada Asignada a un técnico Cerrada con o sin solución. **Procedimiento**: El usuario revisará la incidencia y comprobará que fue cerrada con o sin solución, en este caso, la cerrará automáticamente. **Resultado**: La incidencia cambia al estado Cerrada.	ALTA	**UC-017**

Tabla 42. Plan de pruebas. Rol Supervisor

Código Bloque:		Descripción Bloque:		
CLIEN		*Casos de Prueba para el Rol Cliente*		
Código Caso Prueba	**Título Caso Prueba**	**Descripción Caso Prueba**	**Criticidad**	**Código Caso Uso**
PAF - 017	**Inicio Sesión**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario introduce su login y password de forma correcta. **Resultado**: El usuario accede a la página principal de la aplicación.	ALTA	**UC-001**
PAF - 018	**Generar Incidencia**	**Precondiciones**: El usuario debe estar registrado en el sistema y haber iniciado sesión. **Procedimiento**: 1. El usuario pulsará en el botón para generar la incidencia. 2. Completará cada uno de los campos del formulario correctamente. 3. Guardará la incidencia. **Resultado**: La incidencia generada se guardará y será persistente. El usuario podrá visualizarla cuando desee.	ALTA	**UC-010**

PAF - 019	**Ver listado de incidencias Generadas**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: 1. El usuario accederá al apartado de "Mis Incidencias" donde podrá consultar todas las incidencias generadas. 2. Podrá filtrarlas, si lo desea, por el estado de las mismas. **Resultado**: Se visualizará el listado esperado.	ALTA	**UC-006**
PAF - 020	**Cerrar Incidencia Con solución**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: 1. El usuario elegirá visualizar la incidencia en cuestión. 2. Pulsará en el botón cerrar sin solución. **Resultado**: La incidencia será persistente y cambiará de estado: "cerrada con solución".	MEDIA	**UC-022**
PAF - 021	**Cerrar Incidencia Sin Solución.**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: 1. El usuario elegirá visualizar la incidencia en cuestión. 2. Pulsará en el botón cerrar sin solución. **Resultado**: La incidencia será persistente y cambiará de estado: "cerrada con solución".	MEDIA	**UC-023**

Tabla 43. Plan de pruebas. Rol Cliente.

Código Bloque:		Descripción Bloque:		
TECNI		*Casos de Prueba para el Rol Técnico*		
Código Caso Prueba	**Título Caso Prueba**	**Descripción Caso Prueba**	**Criticidad**	**Código Caso Uso**
PAF - 022	**Inicio Sesión**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario introduce su login y password de forma correcta. **Resultado**: El usuario accede a la página principal de la aplicación	ALTA	**UC-001**
PAF - 023	**Analizar Incidencia**	**Precondiciones**: El usuario debe estar registrado en el sistema y haber iniciado sesión. **Procedimiento**: 1.El usuario accederá a la incidencia asignada. 2.Revisará cada uno de los campos del formulario en cuestión y verificará que son correctos, en base a requisitos determinados. **Resultado**: La incidencia cambiará al estado de asignada.	ALTA	**UC-008**
PAF-024	**Escribir comentarios**	**Precondiciones**: El usuario debe estar registrado en el sistema y haber iniciado sesión. Cada uno de los campos son correctos en base a los requisitos marcados **Procedimiento**: 1.El usuario accederá a la incidencia asignada. 2.Añadirá un comentario por parte de este rol en el que se detallará una solución o la solución que se hizo para solventar la incidencia. **Resultado**: La incidencia cambiará al estado de asignada.	ALTA	**UC-020**

PAF – 025	**Ver todas las incidencias**	**Precondiciones**: El usuario debe estar registrado en el sistema. **Procedimiento**: El usuario generará un listado de todas las incidencias que hay en la aplicación independientemente del estado que tengan, el cliente que las haya registrado y el técnico que las haya tratado. **Resultado**: Se observará el listado con todas las incidencias que se han registrado en el sistema.	ALTA	**UC-006**
PAF - 026	**Cerrar Incidencia Con solución**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: 3. El usuario elegirá visualizar la incidencia en cuestión. 4. Pulsará en el botón cerrar sin solución **Resultado**: La incidencia será persistente y cambiará de estado: "cerrada con solución".	ALTA	**UC-022**
PAF - 027	**Cerrar Incidencia Sin Solución.**	**Precondiciones**: El usuario debe estar registrado en el sistema y en la pantalla principal después de haber iniciado sesión. **Procedimiento**: 3. El usuario elegirá visualizar la incidencia en cuestión. 4. Pulsará en el botón cerrar sin solución. **Resultado**: La incidencia será persistente y cambiará de estado: "cerrada con solución".		**UC-023**

Tabla 44. Plan de Pruebas. Rol Técnico

Por último, como complemento de este estudio y plan de pruebas, también ha sido importante el uso del manual de usuario, que figura en el Anexo I de este documento. Gracias a esta información podremos realizar las pruebas según la funcionalidad con la que se ha enseñado a los usuarios de los diferentes roles a proceder con la realización de cada una de las acciones pertinentes de cada usuario, todas ellas figuran en la Figura 5, correspondiente con el diagrama de los casos de uso de la aplicación.

4.2. Plan de Pruebas Acotado (Smoke-Test).

En este punto, hay que detallar que, en vista a la realización de los casos de uso anteriores, y como antes hemos comentado, la casuística de realización de un plan de pruebas y, por tanto, de los casos de prueba a realizar en una aplicación tiende a infinito. Por esta razón, vamos a acotar este plan de pruebas, orientándolo a la parte más funcional de la aplicación, esto es, el núcleo de la aplicación. De esta forma, podremos verificar y validar que funciona todo tal y como especificó el cliente y que, desde el punto de vista de integración, comprobaremos que funciona la parte más elemental y básica de un gestor de incidencias.

Una vez definido estas premisas, comprobamos que acotando el test de pruebas a realizar y a ejecutar en las diferentes herramientas de automatización, seguimos cumpliendo los objetivos principales que habíamos detallado.

Como hemos comentado a lo largo de este punto, vamos a acotar el plan de pruebas, y como hicimos en el punto anterior hay que definirlo y que mejor forma de definirlo que como se representa en la Figura 4.

Esta figura muestra el ciclo de vida de una incidencia, o lo que es lo mismo los estados fases y roles por los que pasa una incidencia desde que es dada de alta hasta que es cerrada.

Este plan de pruebas, nos resulta más sencillo y rápido que el primer plan de pruebas realizado. Además, ambos son útiles y cumplen los objetivos iniciales que habíamos estimado.

El plan de pruebas a realizar será el siguiente.

1. Iniciaremos sesión con el administrador, creando dos usuarios tester. Un cliente y un Técnico. El supervisor y administrador no podemos crearles ya que ambos tienen que estar presentes en la aplicación, y como vimos en la sección de requisitos, solo puede haber un rol de cada uno.

2. Iniciaremos sesión con el cliente de pruebas y generaremos una incidencia, rellenando debidamente todos los campos que se nos indica en el formulario.

3. Seguidamente lo haremos con el supervisor, la analizaremos, y la aceptaremos asignándola al técnico de pruebas creado en el caso 1. Aunque podamos rechazarla también es una funcionalidad en la que se acabaría el ciclo de vida de esa incidencia. Lo que nos conviene e interesa es comprobar la funcionalidad del núcleo de la aplicación.

4. Ahora iniciaremos sesión con el técnico de pruebas, añadiremos un comentario de la incidencia asignada y elegiremos la forma de cerrarla, esto es: la cerraremos con o sin solución.

5. Nuevamente, iniciaremos sesión con el supervisor, veremos el listado de incidencias por asignadas por el técnico o cliente de pruebas, y cerraremos definitivamente la incidencia.

6. Por último, para dejar la aplicación como nos la encontramos, en la medida de lo posible, ya que no podemos eliminar la incidencia de prueba generada ya que es uno de los requisitos de la aplicación, se eliminarán los dos roles de prueba creados.

Con estos seis escenarios de funcionalidad, abordaremos el núcleo de la aplicación, satisfaciendo los objetivos marcados para probar la misma.

Este es el plan de pruebas que tendremos que implementar en las diferentes herramientas de automatización de pruebas, que se describen en el capítulo V, de este documento. Así conseguiremos probar y generar los entornos correspondientes a cada una de las herramientas, para ejecutarlos en nuestra aplicación, y podremos realizar una comparativa de las mismas.

4.3. Selección de las herramientas de automatización.

A continuación, vamos a realizar un marco comparativo de las herramientas que se van a utilizar para realizar la automatización de las pruebas de aceptación funcionales de la aplicación web descrita en el punto 3.1.

Los criterios que se han tenido en cuenta para la selección de dichas herramientas han sido:

- Popularidad, esto es, que las herramientas a seleccionar sean lo suficientemente conocidas para que sean válidos los resultados y, por tanto, sea más creíble para el cliente.
- Multiplataforma.
- Desarrollo de Pruebas de Aceptación funcionales o de regresión.
- Open Source
- Sencillo
- Ágil
- Frameworks de test de pruebas.
- Probar cualquier página web.

Las herramientas seleccionadas, que han cumplido los requisitos anteriormente marcados, son:

- Selenium WebDriver.
- Watir
- Capibara
- Cucumber

En el Capítulo V veremos cada una de estas herramientas con detalle, además se explicará cómo se realiza la instalación de las mismas, el desarrollo del plan de pruebas y la obtención de los resultados.

Capítulo V: Herramientas de automatización de pruebas

Antes de comenzar con este capítulo, cabe destacar la metodología que se ha seguido como realización de este punto.

En primer lugar, se realiza un estudio de la herramienta, instalándola, buscando información y comprobando su funcionalidad, es decir, nos familiarizamos con la misma y preparamos el entorno para el desarrollo, ejecución y prueba de cada uno de los tests.

Después, se ha partido del plan de pruebas realizado en el subapartado 4.3, de forma general se implementará en cada una de las herramientas teniendo en cuenta el lenguaje de programación que se elija y, además, cada uno de los navegadores que soportan.

Para el desarrollo de las pruebas, se realiza de una forma muy similar en todas las herramientas, ya que todas ellas se basan en funcionar de forma análoga a un usuario cuando utilizase la aplicación.

El DOM (Document Object Mode), es la interfaz de la plataforma que proporciona un conjunto estándar de objetos, cuya jerarquía se puede ver en la Figura 12, para representar documentos HTML, XML, XHTML.

A través de esta interfaz, accederemos a cada uno de los elementos que forman parte de nuestra interfaz de usuario de la aplicación web. El primer paso es acceder a ellos por alguna de sus propiedades, principalmente por: id, name y text, una vez que se han encontrado gracias a esos atributos, nosotros podemos manipularlas, y a la ejecución de esta manipulación en los elementos de cada una de las pantallas que encontremos para realizar la funcionalidad del núcleo de nuestra aplicación lo llamaremos automatización.

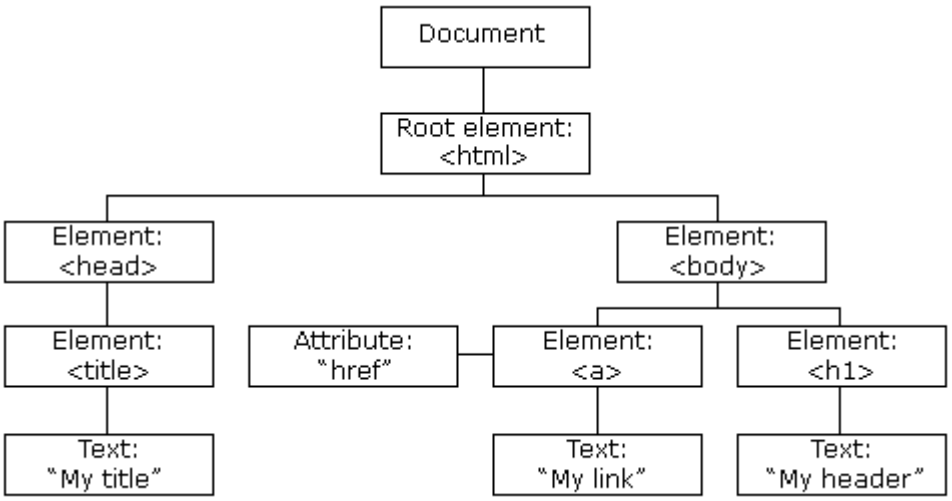

Figura 12. DOM Árbol de Objetos [24].

Una vez que lo hemos implementado, se probará y ejecutará y se extraerán una serie de conclusiones que se expondrán en el capítulo VI.

Por tanto, una vez definido el proceso o metodología a seguir, se verá reflejada, en los siguientes subapartados de este capítulo, donde en cada una de las herramientas que se han utilizado se hará hincapié en cada uno de los puntos que se han comentado anteriormente.

5.1. Selenium

Selenium es un conjunto de herramientas que automatizan el navegador. Cuyo propósito principal se basa en dos objetivos principales:

- Automatización de entornos web para poder ejecutar planes de pruebas programados previamente y poder así comprobar el comportamiento del entorno.
- Tareas de administración web.

Está formado principalmente por los siguientes componentes:

- **WebDriver** Selenium: Se puede controlar el navegador de forma nativa, ya sea desde local como a máquinas remotas.

En la Figura 13 podemos ver las utilidades a las que se destina este tipo de herramienta:

- Integración: se desarrolla para el mundo real y por lo tanto depende de proyectos reales con configuraciones reales:
  - Desarrollo Ágil: parte clave en la automatización del navegador.
  - Nube.
  - Building-Tools: Maven, Ant…
  - CI-Servidor: Jenkins (integración continua).
- Patrón: patrones de diseño como Page Object-Pattern.
- Webdriver API
- Programación: habilidades fundamentales para la automatización del navegador web.
  - IDE: NetBeans, Eclipse, ...
  - HTML
  - XPath: necesaria para la navegación sofisticada.
  - Test-Marco: TestNG, JUnit, ...
- Herramientas del navegador.
- Partes internas: nos permitirán realizar soluciones sofisticadas para problemas complejos, ya que no es necesario que conozcamos perfectamente el funcionamiento interno en el momento de su uso.
  - Navegador partes internas. Facilitan a los navegadores que no sean vistos como una caja negra
  - Drivers internos del navegador: el más utilizado Firefox.

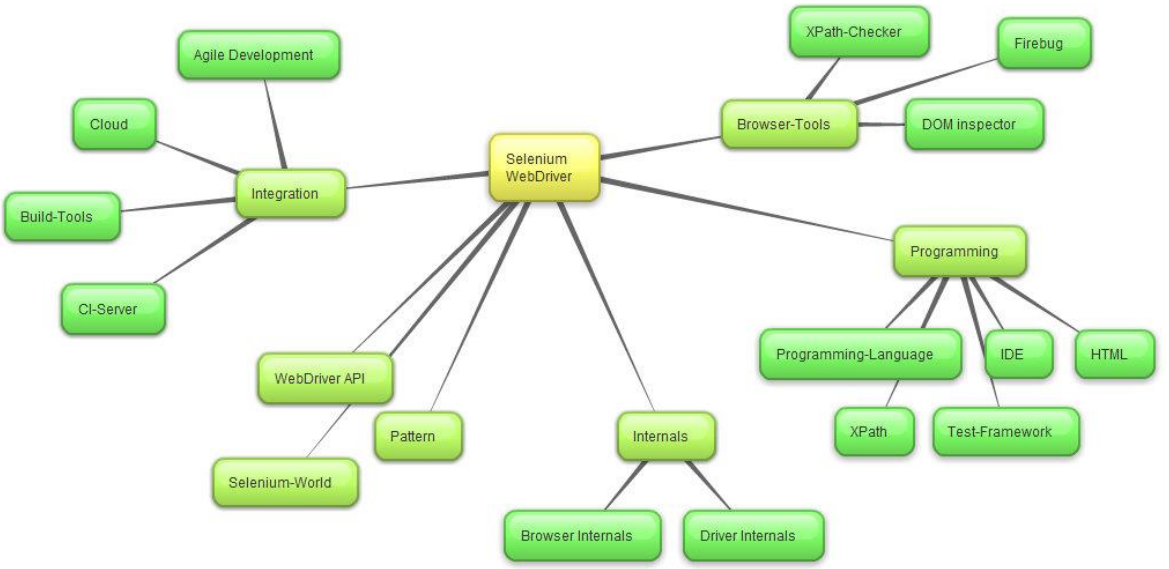

Figura 13.Selenium WebDriver mapa estructural [26]

- **Selenium IDE.** Es un entorno de desarrollo implementado como una extensión de Firefox y permite grabar, editar y depurar pruebas funcionales. En la Figura 14 podemos ver la interfaz de esta herramienta.

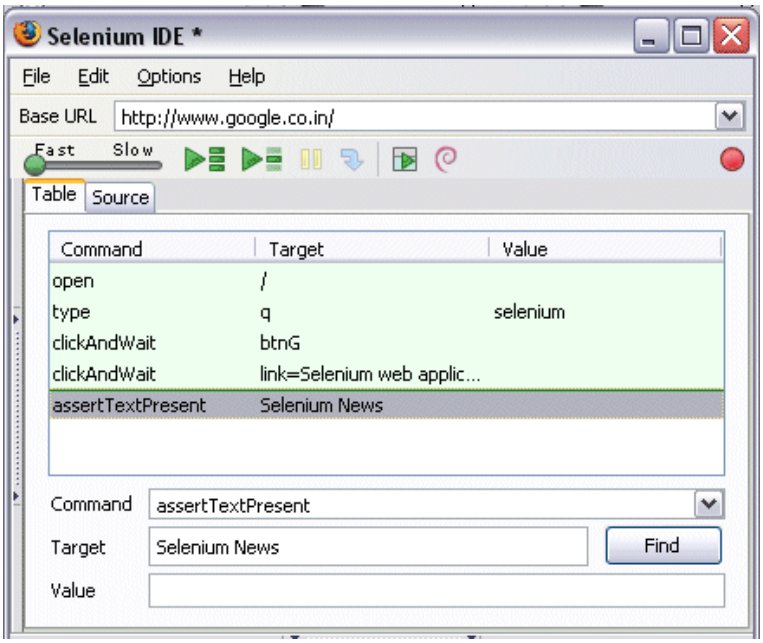

Figura 14. Interfaz Selenium IDE [11].

- **Selenium RC:** Es un sistema cliente / servidor que le permite controlar los navegadores web localmente o en otros equipos, utilizando casi cualquier lenguaje de programación y marco de pruebas.

- **Selenium Grid:** Es igual a Selenium control remoto a otro nivel mediante la ejecución de pruebas en muchos servidores al mismo tiempo, reduciendo el tiempo que se necesita para probar múltiples navegadores o sistemas operativos. En la Figura 15 podemos ver un diagrama de cómo es el sistema que se forma al usar esta herramienta.

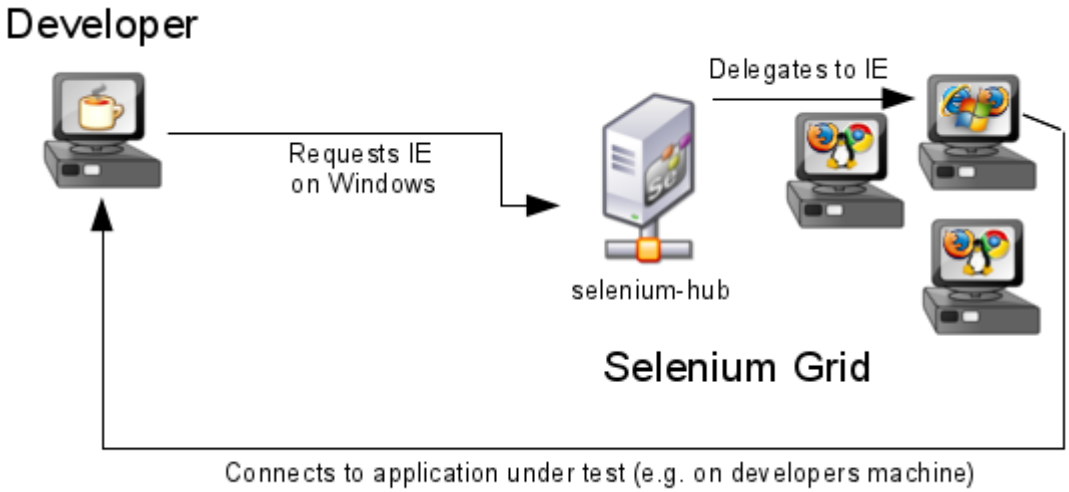

Figura 15. Selenium Grid [20].

A la hora de seleccionar la herramienta que nos interesa, tendremos que elegir entre dos Selenium WebDriver y Selenium IDE.

Selenium WebDriver, se utiliza para:

- Crear pruebas sólidas, basadas en el navegador suites de automatización de regresión y pruebas.
- Capacidad de escalar y distribuir guiones sobre muchos entornos.
- Es el sucesor del Selenium RC, ya que éste está actualmente obsoleto. Además, ahora también incluye una función de las capacidades de la red.

De igual forma, el Selenium IDE, se utiliza para:

- Crear secuencias de comandos para realizar la automatización de pruebas definidas, con el objetivo de detectar errores.
- Como complemento de Firefox para reproducir las interacciones con el navegador.

Por tanto, en base a las características y especificaciones iniciales de las herramientas de automatización seleccionaremos Selenium WebDriver, ya que es el que mejor se adapta a dichas especificaciones.

5.1.1. Instalación

Una vez que tenemos elegida la herramienta a utilizar necesitaremos instalar un IDE y Selenium. Como IDE se ha seleccionado Eclipse. El siguiente paso ahora es instalar Selenium, para ello tenemos que acceder a la página principal: http://www.seleniumhq.org/projects/webdriver/.

En la Figura 16, vemos la página principal de Selenium, en la que se indica en rojo el lugar en el que hay que hacer click para descargar Selenium.

Tal y como comentamos anteriormente, a la hora de descargar hay que elegir el lenguaje de programación que queremos.

En la Figura 17 se pueden apreciar los diferentes tipos de lenguajes que podemos descargar.

En nuestro caso, seleccionaremos el lenguaje de programación Java, la razón principal es que es el que más se usa para realizar este tipo de pruebas, además es junto con C# el que más conocemos.

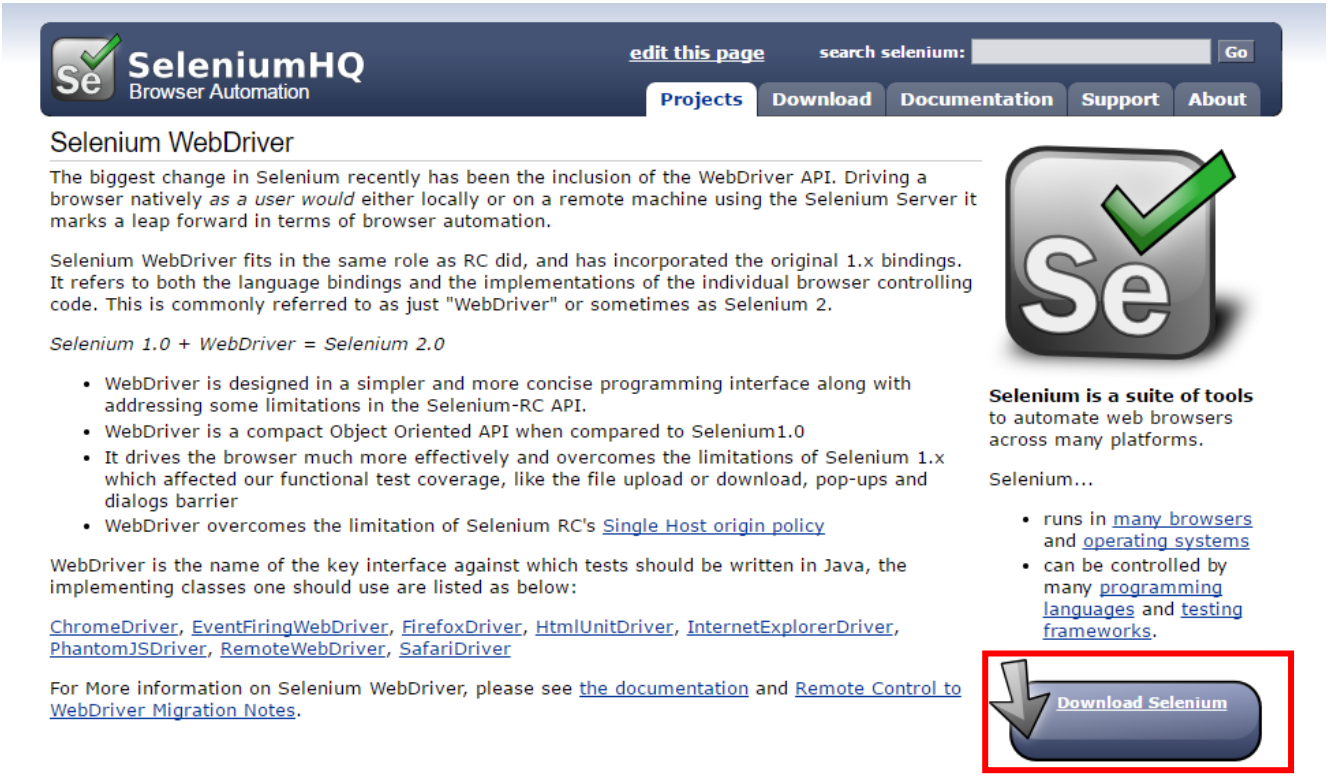

Figura 16. Página Principal Selenium.

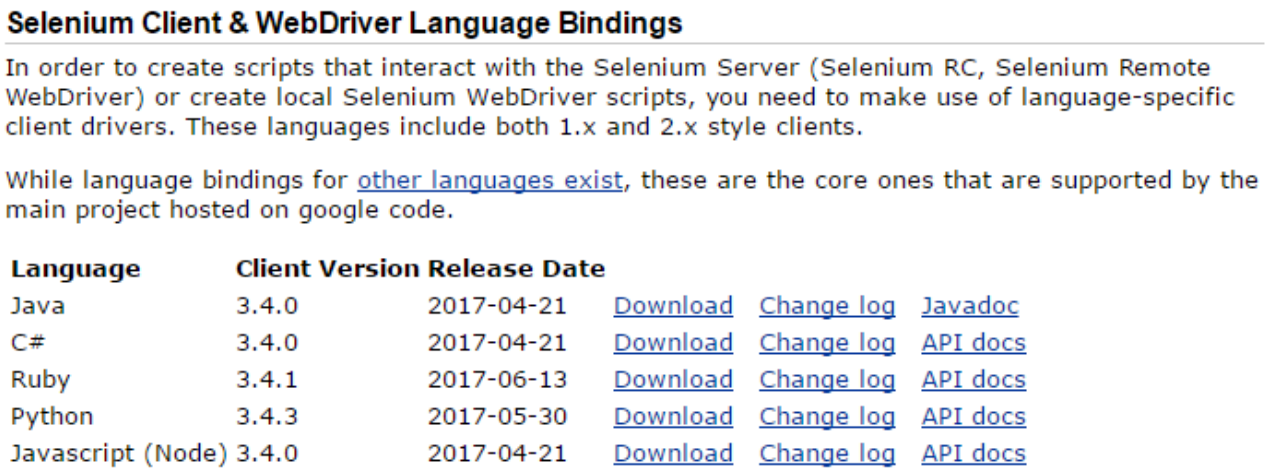

Figura 17. Lenguajes de Selenium.

Una vez que hemos descargado Selenium, tenemos que añadir todos los archivos jar al proyecto Java que crearemos en Eclipse. En la Figura 18 podemos observar la pantalla de Eclipse en la que tenemos que agregar estos ficheros. Para ello pulsaremos sobre la opción *Add External JARs* y buscaremos en nuestro equipo donde descargamos con anterioridad Selenium.

Por último, antes de crear una clase y comenzar con el desarrollo de las pruebas, cabe destacar que utilizaremos un Framework de pruebas Java: JUnit [10], también descargaremos de la página oficial y lo añadiremos en los ficheros jar en la misma pantalla y de forma análoga en la misma página que aparece en la Figura 16.

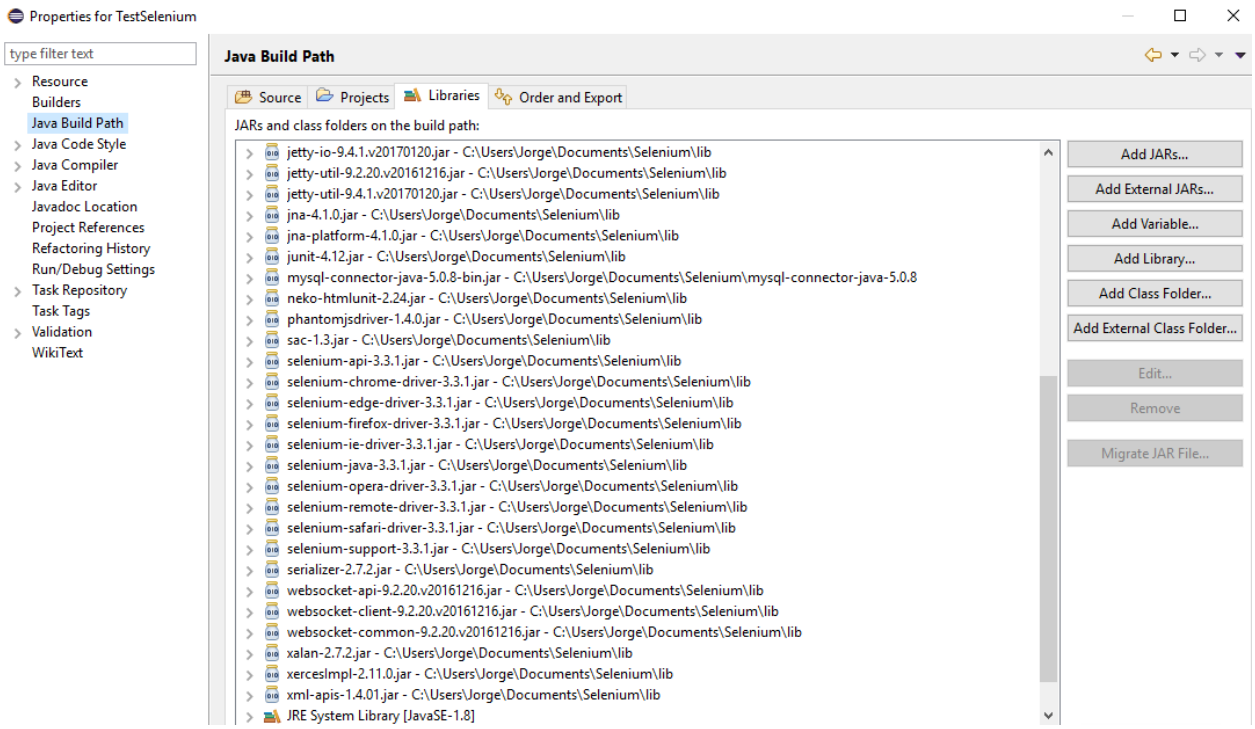

Figura 18. Adición JARs Proyecto Eclipse.

Una vez que hemos realizado todos estos pasos previos, tendremos una arquitectura como la que aparece en la Figura 19. En la que tendremos nuestros test, seguidamente una capa de Selenium WebDriver, luego una más para cada uno de los navegadores que admite esta herramienta y, por último, nuestra aplicación web.

Por último, tendremos que acceder a la página principal y seleccionar los drivers de los navegadores que vamos a utilizar para ejecutar en cada uno de ellos el plan de pruebas que hemos realizado.

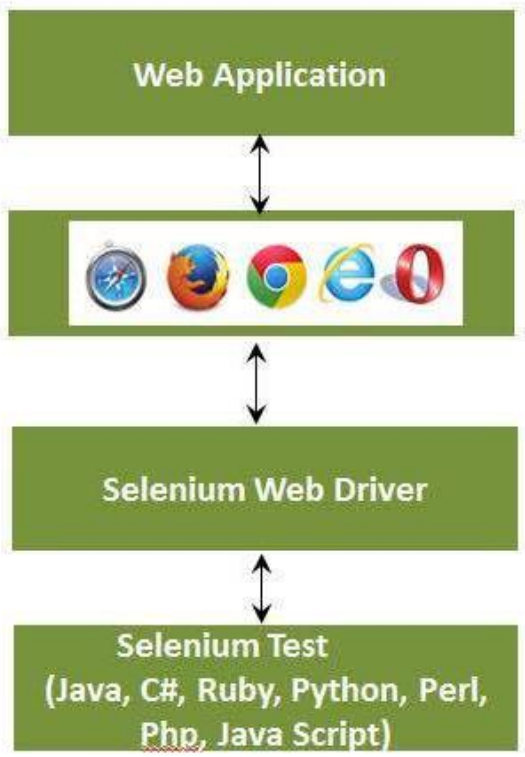

Figura 19. Arquitectura Selenium WebDriver [27].

En la Figura 20 se puede ver la parte de la página principal en la que figuran la tabla en la que se pueden descargar los drivers de cada navegador.

Browser					
Mozilla GeckoDriver	0.16.1	change log	issue tracker	Implementation Status	Released 2017-04-26
Google Chrome Driver	2.29	change log	issue tracker	selenium wiki page	Released 2017-04-04
Opera	2.27		issue tracker	selenium wiki page	Released 2017-04-04
Microsoft Edge Driver			issue tracker	Implementation Status	
GhostDriver	(PhantomJS)		issue tracker	SeConf talk	
HtmlUnitDriver	2.26		issue tracker		Released 2017-04-04
SafariDriver			issue tracker		
Windows Phone			issue tracker		
Windows Phone	4.14.028.10		issue tracker		Released 2013-11-23
Selendroid - Selenium for Android			issue tracker		
ios-driver			issue tracker		
BlackBerry 10			issue tracker		Released 2014-01-28

Figura 20. Drivers Navegadores.

Nosotros hemos tenido en cuenta el sistema operativo que estamos utilizando, que es Windows 10, por esta razón hemos seleccionado los navegadores:

- Internet Explorer
- Edge
- Google Chrome
- Opera
- Firefox

5.1.2. Desarrollo del plan de pruebas.

Para el desarrollo de las pruebas se ha utilizado Java, junto con el framework de JUnit, que sirve para desarrollar test de pruebas en Java.

Tal y como se vio en el punto 4.2 de este documento, se va a probar el núcleo de la aplicación web que se describió en el capítulo III.

Para realizarlo se ha generado la siguiente estructura de desarrollo, dentro de un proyecto Java:

- Clase TestWebPage: Se desarrolla en java el código relativo al plan de pruebas definido y acotado (Smoke test) en el punto 4.3 de este documento.
- Enum Browsers: Se definen todos los navegadores en relación con la herramienta
- Clase Utilities: Clase de utilidad en la que se desarrollan de forma genérica las acciones más habituales y que comparten todos los roles.

Se puede ver el código de la clase de TestWebPage en java, separada por los diferentes métodos de test o pruebas test. Estos métodos se diferencian claramente ya que se puede ver como tienen la nomenclatura de "@test" delante de ellos.

Se ha desarrollado por tanto un método o método test (de pruebas) por cada uno de los usuarios que hay y por supuesto, se ha realizado un assert correspondiente para validar a lo largo de la ejecución de las pruebas el correcto funcionamiento de cada uno de los pasos previos. El propio framework de JUnit nos garantiza y nos muestra los resultados de la ejecución de cada uno de los métodos.

La clase Utilities, como se describió anteriormente, el objetivo de la misma es simplificar y que sea una clase utilidad para poder utilizarla en los diferentes tipos de roles para las acciones que se llevan a cabo en el desarrollo del plan de pruebas sin problema.

Las acciones que son más usadas en los diferentes roles son: primero la elección del navegador, como podemos ver en el método *chooseBrowser* tiene una lógica de configuración en los diferentes navegadores, que como vemos en cada uno de ellos necesita un driver determinado, que se puede descargar, como hemos visto, anteriormente, en la Figura 18.

Otra funcionalidad típica es introducir texto en un campo, pulsar un botón, seleccionar una de las opciones del campo select, y cerrar la instancia de WebDriver y, por tanto, liberar los recursos que no se van a utilizar a continuación veremos el enum con los diferentes tipos de navegadores que vamos a utilizar a lo largo del desarrollo de las pruebas.

```java
package Selenium;
/**
*Enum con los diferentes tipos de navegadores soportados.
* @author Jorge
*
*/
public enum Browsers
{
	CHROME,
	IE,
	EDGE,
	FIREFOX,
	OPERA
}
```

Justo a continuación, Podemos ver la clase Utilities, que como bien se detalló, se utilizará para que las clases las utilicen en los métodos más habituales a lo largo de la automatización.

```java
package Selenium;

import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.ie.InternetExplorerDriver;
import org.openqa.selenium.opera.OperaDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;


/**
 * Clase que implementa de forma genérica las acciones de interacción con el entorno web
 * @author Jorge
 *
 */
public class Utilities {
	/**
	 * Tiempo (en segundos) de espera máxima para seleccionar un elemento.
	 */
	private int timeWait = 10; //Seg
	/**
	 * Título de la págnia inicial del entorno
	 */
	private final String initialPage= "Gestión de Incidencias";
	/**
	 * Getter, para obtener el nombre inicial de la página
	 * @return initialPage
	 */
	public String getInitialPage() {
		return initialPage;
	}
```

```
	/**
	 * Método que selecciona el navegador deseado
	 * @param navegador
	 * @return Webdriver
	 */
	public  WebDriver chooseBrowser(Browsers navegador){
		WebDriver driver = null;
		switch(navegador){

		case CHROME:
			System.setProperty("webdriver.chrome.driver",
"C:/Users/Jorge/Documents/Selenium/jar/chromedriver.exe");
			 driver  = new ChromeDriver();
			return driver;

		case IE:
			System.setProperty("webdriver.ie.driver",
"C:/Users/Jorge/Documents/Selenium/jar/IEDriverServer.exe");
			 driver  =  new InternetExplorerDriver();
			return driver;

		case EDGE:
			System.setProperty("webdriver.edge.driver",
"C:/Users/Jorge/Documents/Selenium/jar/MicrosoftWebDriver.exe");
			driver  =  new EdgeDriver();
			return driver;

		case OPERA:
			System.setProperty("webdriver.opera.driver",
"C:/Users/Jorge/Documents/Selenium/jar/operadriver_win64/operadriver.exe");
			driver = new OperaDriver();
			return driver;

		case FIREFOX:

	System.setProperty("webdriver.gecko.driver","C:/Users/Jorge/Documents/Seleni
um/geckodriver_64.exe");
		driver = new FirefoxDriver();
		return driver;
		}
		return driver;
	}

	/**
	 * Método que inicia sesión en el sistema
	 * @param user
	 * @param pass
	 * @param driver
	 * @throws InterruptedException
	 */
	public  void  logon(String  user,  String  pass,  WebDriver  driver)  throws
InterruptedException{
		WebElement username;
		username = driver.findElement(By.name("user"));
		username.clear();
		username.sendKeys(user);
		WebElement password;
		password = driver.findElement(By.name("pass"));
		password.sendKeys(pass);

		WebElement btnAceptar;
		btnAceptar = driver.findElement(By.id("btnAceptar"));
		btnAceptar.click();
```

```
            Thread.sleep(timeWait);
      }

      /**
       * Metodo que cierra sesión en el entorno
       * @param driver
       * @throws InterruptedException
       */
      public void logoff(WebDriver driver) throws InterruptedException{
            WebElement salir;
            salir = driver.findElement(By.id("salir"));
            salir.click();

            Alert alert = driver.switchTo().alert();
            alert.accept();

            Thread.sleep(timeWait);
      }

      /**
       * Metodo que inserta datos en formularios.
       * @param driver
       * @param idElemento
       * @param datos
       * @throws InterruptedException
       */
      public void insert(WebDriver driver, String idElemento, String datos) throws
InterruptedException{
            WebElement elemento;
            WebDriverWait wait = new WebDriverWait(driver, timeWait);

      wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(idElemento)))
;
            driver.findElement(By.id(idElemento)).sendKeys(datos);
            elemento = driver.findElement(By.id(idElemento));
            elemento.sendKeys(Keys.ENTER);

      }

      /**
       * Metodo que presiona el botón indicado.
       * @param driver
       * @param idElemento
       * @throws InterruptedException
       */
      public void pressBoton(WebDriver driver, String idElemento) throws
InterruptedException{


            WebDriverWait wait = new WebDriverWait(driver,timeWait);

      wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(idElemento)))
;
            driver.findElement(By.id(idElemento)).click();
      }
```

```
	/**
	 * Metodo que selecciona una opción dentro de un box select
	 * @param driver
	 * @param nameElemento
	 * @param value
	 * @throws InterruptedException
	 */
	public void selectValue(WebDriver driver, String nameElemento, String value)
throws InterruptedException{
		Select select= new Select(driver.findElement(By.name(nameElemento)));
		select.selectByVisibleText(value);
		Thread.sleep(timeWait);
	}

	/**
	 * Método que hace click en hipervinculos
	 * @param driver
	 * @param value
	 * @throws InterruptedException
	 */
	public void chooseOption(WebDriver driver, String value) throws
InterruptedException{
		WebElement option;
		option= driver.findElement(By.partialLinkText(value));
		option.click();
		Thread.sleep(timeWait);
	}


	/**
	 * Método que cierra el webdriver
	 * @param driver
	 */
	public void closeDriver(WebDriver driver) {
	 try{
		if(driver!=null)
			driver.close();

		}catch(Exception exc){
			throw exc;
		}


	}
}
```

Por último, a continuación, tenemos el código fuente del plan de pruebas acotado de toda la aplicación y desarrollado en Java.

```java
package Selenium;

import static org.junit.Assert.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.Select;
/**
 * Clase test del núcleo de la aplicación
 * @author Jorge
 *
 */
public class TestWebPage {
	/**
	 * WebDriver
	 */
	private static WebDriver driver;
	/**
	 * Dirección web de la página a probar
	 */
	private static final String dirWeb = "http://virtual.lab.inf.uva.es:28032/";

	/**
	 * Instancia de la clase utilities
	 */
	private static Utilities utilities;

	/**
	 * Constante de string vacío
	 */
	private static final String EMPTY_STRING = " ";


	/**
	 * Array sobre los posibles usuarios que puede crear el Administador
	 */
	private static String[] chooseRol;
	/**
	 * Login del rol Administrador
	 */
	private static String userAdmin;
	/**
	 * Contraseña del rol Administrador
	 */
	private static String passAdmin;
	/**
	 * Login del rol Supervisor
	 */
	private static String userSuper;
	/**
	 * Contraseña del rol Supervisor
	 */
	private static String passSuper;
	/**
```

```
	 * Login del rol cliente
	 */
	private static String userTestClient;
	/**
	 * Contraseña del rol cliente
	 */
	private static String passTestClient;
	/**
	 * Login del rol técnico
	 */
	private static String userTestTecnico;
	/**
	 * Contraseña del rol técnico
	 */
	private static String passTestTecnico;
	/**
	 * Día de hoy
	 */
	private static Date today;
	/**
	 * Comentarios que realiza el técnico al cerrar la incidencia
	 */
	private static String comentarios;

	/**
	 * Incialización de cada uno de los atributos.
	 * @throws Exception
	 */
	@BeforeClass
	public static void setUp() throws Exception {
		utilities = new Utilities();
		driver=utilities.chooseBrowser(Browsers.CHROME);
		driver.get(dirWeb);
		driver.manage().window().maximize();
		chooseRol = new String[]{"Cliente","Técnico"};
		userAdmin = "admin";
		passAdmin = "admin";
		userSuper = "david";
		passSuper = "supervisor";
		userTestClient = "TestCliente";
		passTestClient = "passTestClient";
		userTestTecnico="TestTecnico";
		passTestTecnico="passTestTecnico";

		today = new Date();
		comentarios ="Incidencia revisada y solucionada por"
				+ " el Técnico de Pruebas";
	}

	/**
	 * Clase Test que ejecuta cada una de las acciones de los roles de la aplicación
	 * @throws InterruptedException
	 */
	@Test
	public void testWebPage() throws InterruptedException {

		testAdministrador();

		testCliente();

		testSupervisor();

		testTecnico();
```

```
            testCloseInicidence();


        }

        /**
         * Método del rol Adminsitrador, crea los dos usuarios de pruebas
         * @throws InterruptedException
         */
        public void testAdministrador() throws InterruptedException{

            System.out.println("--------------------------------");
            System.out.println("--------ADMINISTRADOR----------");
            System.out.println("--------------------------------");

            createUser(userTestClient, passTestClient, chooseRol[0]);

            System.out.println("Crear Cliente de Pruebas ---->  OK");

            createUser(userTestTecnico, passTestTecnico, chooseRol[1]);

            System.out.println("Crear Técnico de Pruebas ---->  OK");

        }

        /**
         * Método del rol Cliente, genera una incidencia.
         * @throws InterruptedException
         */
        public void testCliente() throws InterruptedException{

            System.out.println("--------------------------------");
            System.out.println("-----------CLIENTE-------------");
            System.out.println("--------------------------------");

            utilities.logon(userTestClient,passTestClient,driver);

            assertEquals(utilities.getHomePage(), driver.getTitle());

            System.out.println("Inicio de Sesión Cliente ---->  OK");

            utilities.pressBoton(driver, "misIncidencias");

            utilities.pressBoton(driver, "agregarIncidencia");

            utilities.insert(driver, "descripcion", "Incidencia Prueba");

            utilities.insert(driver, "fecha",new
SimpleDateFormat("dd/MM/yyyy").format(today));

            utilities.selectValue(driver, "prioridad", "Alta");

            utilities.selectValue(driver, "categoria", "HARDWARE");

            utilities.pressBoton(driver, "submit");

            assert checkIncidence(): "Creación Incidencia ----> KO";

            System.out.println("Creación Incidencia ---->  OK");

            utilities.logoff(driver);
```

```
            assertEquals(utilities.getInitialPage(), driver.getTitle());

            System.out.println("Cierre de Sesión Cliente Pruebas ---->  OK");

      }

      /**
       * Método del rol Supervisor, asigna una incidencia creada a un ténico, añadiendo
los elementos
       * afectados en la misma.
       * @throws InterruptedException
       */
      public void testSupervisor() throws InterruptedException{

            System.out.println("-------------------------------");
            System.out.println("---------SUPERVISOR------------");
            System.out.println("-------------------------------");

            utilities.logon(userSuper,passSuper, driver);

            assertEquals(utilities.getHomePage(), driver.getTitle());

            System.out.println("Inicio de Sesión Supervisor ---->  OK");

            utilities.pressBoton(driver, "gestionIncidencias");

            utilities.selectValue(driver, "filtroIncidencias", "Por Cliente");

            utilities.selectValue(driver, "filtroRoles", userTestClient);

            utilities.chooseOption(driver, "Ver");

            utilities.selectValue(driver,"accion","Aceptar");

            Select select = new Select(driver.findElement(By.id("elementos")));
            select.selectByVisibleText("TECLADO");

            utilities.selectValue(driver, "tecnico", userTestTecnico);

            utilities.pressBoton(driver, "submit");

            assert checkEstateIncidence("Asignada","Por Cliente",userTestClient): "Cambio
estado Incidencia ----> KO";

            System.out.println("Cambio de Estado Incidencia Asignada ---->  OK");

            System.out.println("Asignar Técnico ---->  OK");

            utilities.logoff(driver);

            assertEquals(utilities.getInitialPage(), driver.getTitle());

            System.out.println("Cierre de Sesión Supervisor ---->  OK");

      }
      /**
       * Método del rol Técnico, selecciona la incidencia asignada por el supervisor
       * y la resuelve con solución.
       * @throws InterruptedException
       */
      public void testTecnico() throws InterruptedException{

            System.out.println("-------------------------------");
```

```
            System.out.println("-----------TÉCNICO-------------");
            System.out.println("-------------------------------");

            utilities.logon(userTestTecnico, passTestTecnico, driver);

            assertEquals(utilities.getHomePage(), driver.getTitle());

            System.out.println("Inicio de Sesión Técnico ---->  OK");

            utilities.pressBoton(driver, "incidencias");

            utilities.chooseOption(driver, "Ver");

            utilities.insert(driver, "comentario", comentarios);

            utilities.pressBoton(driver, "submit");

            assert checkIncidenceClosed(): "Estado Incidencia Cerrada Con solución ---
KO";

            System.out.println("Estado Incidencia Cerrrada Con solución --- KO");

            utilities.logoff(driver);

            assertEquals(utilities.getInitialPage(), driver.getTitle());

            System.out.println("Cierre de Sesión Técnico Pruebas ---->  OK");

      }
      /**
       * Método del rol Supervisor, cierra la incidencia creada definitivamente.
       * @throws InterruptedException
       */
      public void testCloseInicidence() throws InterruptedException{

            utilities.logon(userSuper,passSuper, driver);

            assertEquals(utilities.getHomePage(), driver.getTitle());

            System.out.println("Inicio de Sesión Supervisor ---->  OK");

            utilities.pressBoton(driver, "gestionIncidencias");

            utilities.selectValue(driver, "filtroIncidencias", "Por Técnico");

            utilities.selectValue(driver, "filtroRoles", userTestTecnico);

            utilities.chooseOption(driver, "Ver");

            utilities.pressBoton(driver, "submit");

            System.out.println("Cierre de Incidencia por Supervisor ---->  OK");

            utilities.logoff(driver);

            assertEquals(utilities.getInitialPage(), driver.getTitle());

            System.out.println("Cierre de Sesión Supervisor ---->  OK");
      }
      /**
       * Método que libera recursos.
       */
      @AfterClass
```

```
	public static void absolve(){

	utilities.closeDriver(driver);

	}
private void createUser(String user, String pass, String rolUser) throws
InterruptedException{

	utilities.logon(userAdmin, passAdmin, driver);

	assertEquals(utilities.getHomePage(), driver.getTitle());

	System.out.println("Inicio de Sesión Administrador ---->  OK");

	utilities.pressBoton(driver, "gestionUsuarios");

	utilities.pressBoton(driver, "agregarUsuario");

	utilities.insert(driver, "name", user);

	utilities.insert(driver, "PrimerApellido", EMPTY_STRING);

	utilities.insert(driver, "SegundoApellido", EMPTY_STRING);

	utilities.insert(driver, "login", user);

	utilities.insert(driver, "password", pass);

	utilities.insert(driver, "Rpassword", pass);

	utilities.selectValue(driver, "tipo", rolUser);

	utilities.pressBoton(driver, "submit");

	//Thread.sleep(500);
	driver.navigate().refresh();

	assert checkUserCreate(user): "Crear el usuario"+ user+"----> KO";

	System.out.println("Crear el usuario"+ user+"----> OK" );

	utilities.logoff(driver);

	assertEquals(utilities.getInitialPage(), driver.getTitle());

	System.out.println("Cierre de Sesión Administrador ---->  OK");
	}

	private boolean checkUserCreate(String loginUser){
		boolean exists = false;
		List<WebElement> users = driver.findElements(By.id("nombre"));
		for(int k=0; k<users.size(); k++)
		{
			if(users.get(k).getText().equals(loginUser)){
				return true;
			}
		}

		return exists;
	}
	/**
	 * Método privado que comprueba si se creó una incidencia.
	 * @return incidence
```

```
	 */
	private boolean checkIncidence(){
		boolean incidence = false;
		List<WebElement> incidences  = driver.findElements(By.id("Array"));
		if(!incidences.isEmpty()){
			incidence = true;
			return incidence;
		}
		return incidence;
	}

	/**
	 * Método privado que comprueba el estado de la incidencia
	 * @param expectedEstate
	 * @param filter
	 * @param user
	 * @return estateOk
	 * @throws InterruptedException
	 */
	private boolean checkEstateIncidence(String expectedEstate, String filter, String
user) throws InterruptedException{
		boolean estateOk = false;
		utilities.selectValue(driver, "filtroIncidencias", filter);

		utilities.selectValue(driver, "filtroRoles", user);

		WebElement incidence  = driver.findElement(By.id("estado"));
		if(incidence.getText().equals(expectedEstate)){
			estateOk = true;
			return estateOk;

		}
		return estateOk;
	}
	/**
	 * Método privado que comprueba que la incidencia tiene estado cerrada
	 * @return
	 * @throws InterruptedException
	 */
	private boolean checkIncidenceClosed() throws InterruptedException{
		boolean close = false;
		utilities.selectValue(driver, "filtroIncidencias", "Cerradas");
		List<WebElement> incidences = driver.findElements(By.id("estado"));
		for( WebElement we : incidences){
			if(we.getText().equals("Cerrada(con solución)")){
				close = true;
				return close;
			}
		}
		return close;
	}
}
```

A continuación, vamos a añadir la clase con la que restauramos el sistema, esto es, la clase con la que eliminamos físicamente todas las modificaciones del entorno, para dejarlo tal y como lo encontramos.

```
package Selenium;

import static org.junit.Assert.*;
import java.util.List;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

/**
 * Clase que elimina los usuarios creados para la ejecución del plan de pruebas
 * @author Jorge
 *
 */
public class TestRestore {

	private static WebDriver driver;
	private static final String dirWeb = "http://virtual.lab.inf.uva.es:28032/";
	private static Utilities restore;
	private static final String userAdmin = "admin";
	private static final String passAdmin = "admin";
	private static final String userTestClien = "TestCliente";
	private static final String userTestTecni = "TestTecnico";
	private static boolean delete = false;
	private static List <WebElement> ListId ;
	private static List <WebElement> ListEliminar;


	/**
	 * Método que incializa los parámetros necesarios para  la ejecución
	 */
	@Before
	public  void setUp(){
		restore = new Utilities();
		driver=restore.chooseBrowser(Browsers.CHROME);
		driver.get(dirWeb);
		driver.manage().window().maximize();
		}

	/**
	 * Método que ejecuta la eliminación del cliente y técnico de pruebas creados.
	 * @throws InterruptedException
	 */
	@Test
	public void testRestor() throws InterruptedException {

		deleteElements(userTestClien);
		//Thread.sleep(1000);
		deleteElements(userTestTecni);
		assertEquals(driver.getTitle(), restore.getInitialPage());
		System.out.println("Test Restore ------> OK");
	}

	/**
	 * Metodo que libera los recursos utilizados, WebDriver
```

```
	 */
	@After
	public void absolve(){
		restore.closeDriver(driver);
	}


	/**
	 * Metodo privado que elimina los usuarios indicados
	 * @param driver
	 * @param idElement
	 * @throws InterruptedException
	 */
	private static void deleteElements(String idUsuario) throws InterruptedException
	{
		restore.logon(userAdmin, passAdmin, driver);

		restore.pressBoton(driver, "gestionUsuarios");

		ListId = driver.findElements(By.id("nombre"));

		ListEliminar = driver.findElements(By.id("eliminar"));

		int position = findElements(idUsuario);

		if(delete){
			ListEliminar.get(position).click();
			driver.switchTo().alert().accept();
		}

			restore.logoff(driver);

	}
	/**
	 * Método privado que devuelve la posición de los elementos a borrar, si estos
existen.
	 * @param element
	 * @return
	 */
	private static int findElements(String element){
		int eliminar = 0;

			for (int i=0; i< ListId.size(); i++)
			{
				if(ListId.get(i).getText().equalsIgnoreCase(element)){
					eliminar = i;
					delete = true;

				}
			}

		return eliminar;
	}

}
```

Y por último, para zanjar el desarrollo de pruebas realizado se va a mostrar la suite de JUnit que se utilizará para ejecutar automáticamente todas las pruebas anteriormente dichas y una vez ejecutado veremos los resultados en la sección 5.1.3.

```
package Selenium;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

/**
 * Suite que ejecuta cada uno de los test de cada uno de los usuarios de forma conjunta
 * @author Jorge
 *
 */
@RunWith(Suite.class)
@SuiteClasses({ TestWebPage.class,
                TestRestore.class
                })
public class AllTests {

}
```

5.1.3. Resultados

Como resultados, tendremos dos resultados que nos serán utilies por la información que nos aportan, el primero lo veremos en la Figura 21, que lo hemos generado al escribir por consola cuando se ejecutaban cada uno de los casos de uso pertenencientes al plan de pruebas marcado.

El otro resultado que obtendremos, nos lo proporciona el propio framework JUnit, que como podemos ver en la Figura 22, nos indica las pruebas que se han ejecutado y si se han ejecutado correctamente o no, o el tiempo de ejecución de las pruebas.

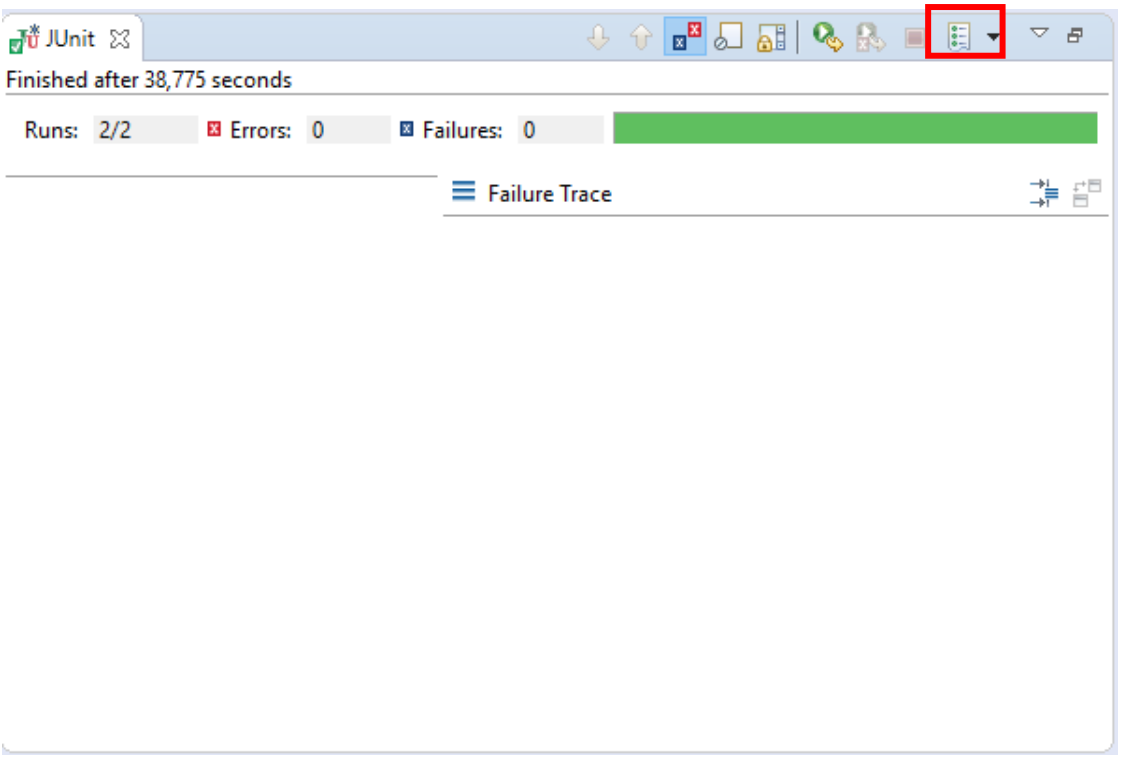

Figura 21. Resultados JUnit

Si pulsamos en el cuadro rojo, que aparece en la parte superior derecha de la Figura 21 podremos descargar un fichero XML con el que tendremos más detalle, esto es, nos indicará el número de pruebas que se han realizado y por cada una de ellas el resultado y tiempo de ejecución en realizarlo, este fichero tiene el siguiente formato:

```
<?xml version="1.0" encoding="UTF-8"?><testrun name="AllTests (1)"
project="TestSelenium" tests="2" started="2" failures="0" errors="0"
ignored="0">
  <testsuite name="Selenium.AllTests" time="32.63">
    <testsuite name="Selenium.TestWebPage" time="22.385">
      <testcase name="testWebPage" classname="Selenium.TestWebPage"
time="22.385"/>
    </testsuite>
    <testsuite name="Selenium.TestRestore" time="10.245">
      <testcase name="testRestor" classname="Selenium.TestRestore"
time="10.245"/>
    </testsuite>
  </testsuite>
</testrun>
```

```
<terminated> AllTests (1) [JUnit] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (3 jul. 2017 11:53:55)
Starting ChromeDriver 2.29.461591 (62ebf098771772160f391d75e589dc567915b233) on port 47018
Only local connections are allowed.
jul 03, 2017 11:53:59 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFORMACIÓN: Detected dialect: OSS
-------------------------------
--------ADMINISTRADOR----------
-------------------------------
Inicio de Sesión Administrador ---->  OK
Crear el usuarioTestCliente----> OK
Cierre de Sesión Administrador ---->  OK
Crear Cliente de Pruebas ---->  OK
Inicio de Sesión Administrador ---->  OK
Crear el usuarioTestTecnico----> OK
Cierre de Sesión Administrador ---->  OK
Crear Técnico de Pruebas ---->  OK
-------------------------------
-----------CLIENTE-------------
-------------------------------
Inicio de Sesión Cliente ---->  OK
Creación Incidencia ---->  OK
Cierre de Sesión Cliente Pruebas ---->  OK
-------------------------------
---------SUPERVISOR------------
-------------------------------
Inicio de Sesión Supervisor ---->  OK
Cambio de Estado Incidencia Asignada ---->  OK
Asignar Técnico ---->  OK
Cierre de Sesión Supervisor ---->  OK
-------------------------------
-----------TÉCNICO-------------
-------------------------------
Inicio de Sesión Técnico ---->  OK
Estado Incidencia Cerrrada Con solución --- KO
Cierre de Sesión Técnico Pruebas ---->  OK
Inicio de Sesión Supervisor ---->  OK
Cierre de Incidencia por Supervisor ---->  OK
Cierre de Sesión Supervisor ---->  OK
```

Figura 22. Resultado Terminal Eclipse

5.2. Watir

Watir (Web Aplication Testing In Ruby) es una herramienta de Testing de aplicaciones web mediante el lenguaje de programación Ruby que automatiza las operaciones de los navegadores web, actualmente trabaja en los siguientes navegadores:

- Internet Explorer.
- Firefox
- Microsoft Edge.
- Chrome
- Safari.

Watir fue desarrollado inicialmente por BretPettichord y Paul Rogers. Es un proyecto de código abierto, que tiene una lista creciente de participantes.

La funcionalidad que simula esta herramienta es como si un usuario la realizase cualquier acción con la plataforma web, esto es, escribir un correo electrónico, consultar cierta información, rellenar un formulario, etcétera.

También muestra la posibilidad de identificar cada uno de los elementos HTML que hay en la página, pudiendo manipularlos y realizar acciones con ellos, de esta forma es con la que se puede simular el funcionamiento al igual que si fuese un usuario.

Encontramos tres tipos de herramientas Watir:

- WatirClassic: hace uso de la funcionalidad del enlazado de objetos que Ruby tiene integrada.
  Opera diferente de otras herramientas de pruebas basadas en HTTP que emulan la existencia de un navegador, en vez de ello, inicia instancia del navegador a través del protocolo OLE que está implementado sobre la arquitectura COM que es la que permite la comunicación entre procesos (por ejemplo, entre Ruby e Internet Explorer) y la creación y manipulación dinámica de objetos, que es lo que hace Ruby con el Internet Explorer. Microsoft llama a esto "automatización OLE" y al programa que ejerce la manipulación lo llama controlador de automatización. Técnicamente el proceso de Internet Explorer es un servidor y sirve objetos de automatización al exponer sus métodos mientras que el programa escrito en Ruby se convierte en el cliente que manipula los objetos de automatización.

- Watir-Webdriver: es una versión moderna del API de Watir basada en Selenium (es un entorno de pruebas de software para aplicaciones basadas en la web). Selenium 2.0 (Selenium-WebDriver) intenta ser la implementación de referencia para la especificación WebDriver. Dado que Watir-Webdriver es derivado tanto de Selenium 2.0 como de la especificación de HTML Watir-Webdriver debe ser siempre compatible con las especificaciones del W3C (es un consorcio internacional que produce recomendaciones para la World Wide Web)

- Watirspec: hacer pruebas funcionales para aplicaciones web. Sin embargo, es necesario tener en cuenta que Watir sólo funciona en IE (aunque existe FireWatir y SafariWatir, tanto que se ejecuta en otras plataformas para Firefox y Safari, respectivamente). Si bien esto parece terriblemente restrictivo para algunos, me dado cuenta de que la fuerza de RSpec + Watir no está en las pruebas de compatibilidad del navegador sino en ser capaz de correr a través de la aplicación web como se ha visto y utilizado por el usuario real, a través de un navegador real. RSpec + Watir produce una capacidad sinérgica donde cada uno no puede. La herramienta combinada nos permite probar cualquier aplicación web (ya sea Ruby on Rails o no) y de hacerlo de una manera BDD.

Después de la clasificación de los diferentes tipos de herramientas de las que consta Watir, instalaremos Watir-Web driver, ya que es el que mejor se adapta a nuestro entorno y es capaz de ejecutarse en más navegadores que Internet Explorer.

5.2.1. Instalación.

A continuación, vamos a instalar esta herramienta, antes de nada, debemos de instalar Ruby, ya que es necesario para que funcione, puesto que es una gema a instalar. La explicación de la instalación de Ruby la podemos ver detalladamente en el Anexo II.

Una vez que hemos seguido los pasos indicados en el anexo, y, por tanto, instalado correctamente Ruby, procederemos a instalar Watir, para ello tenemos que escribir en la terminal de Ruby el siguiente comando.

```
gem install watir
```

Como vemos en la Figura 23, será la salida que veamos en la terminal de Ruby después de escribir el comando, indicado. Se puede ver que se nos instalan una serie de módulos que utilizaremos posteriormente.

El IDE que vamos a utilizar para desarrollar el plan de pruebas de Watir va a ser Eclipse, en concreto el mismo Eclipse que se utilizó para el desarrollo del mismo plan en Selenium; En la Figura 24, vemos que tan solo tenemos que añadir las bibliotecas correspondientes a nuestra versión de Ruby para que podamos funcionar con Eclipse las pruebas en cuestión.

Llegados a este punto, tenemos Watir correctamente instalado y preparado para desarrollar el plan de pruebas en esta herramienta.

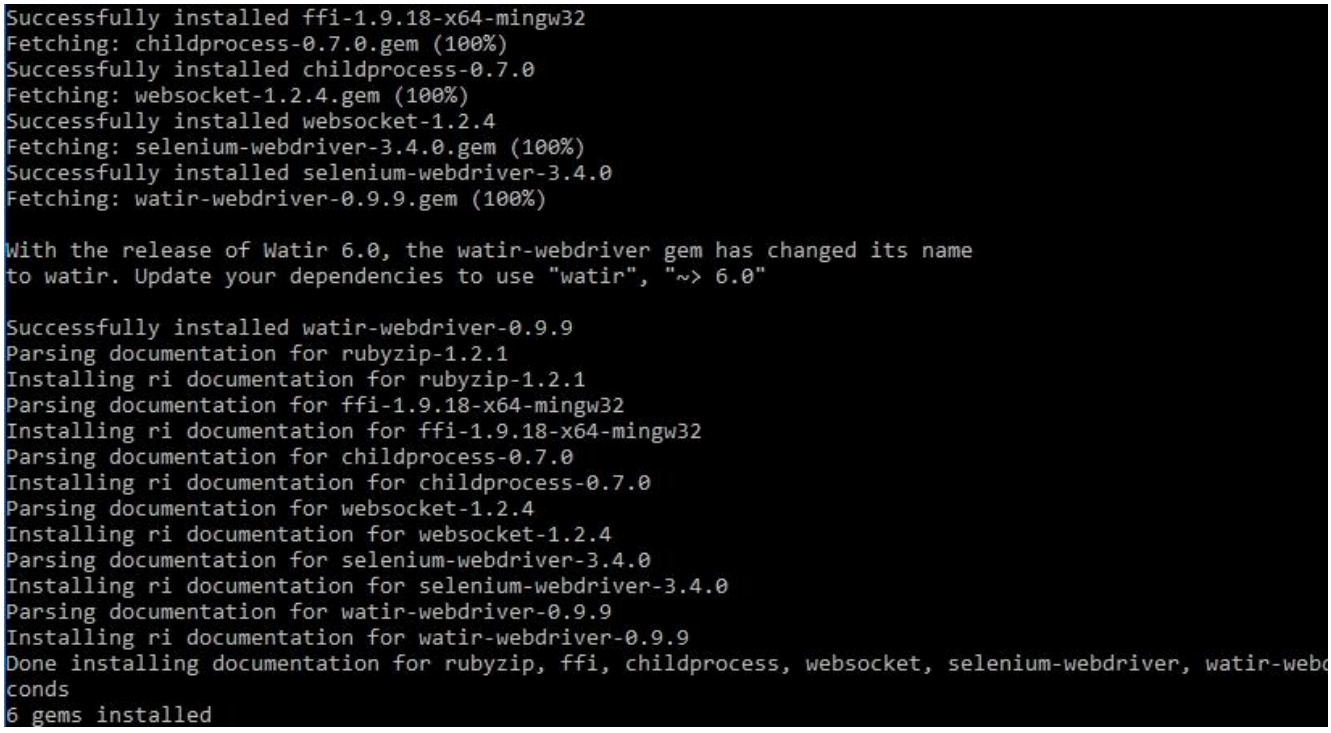

Figura 23. Instalación Watir

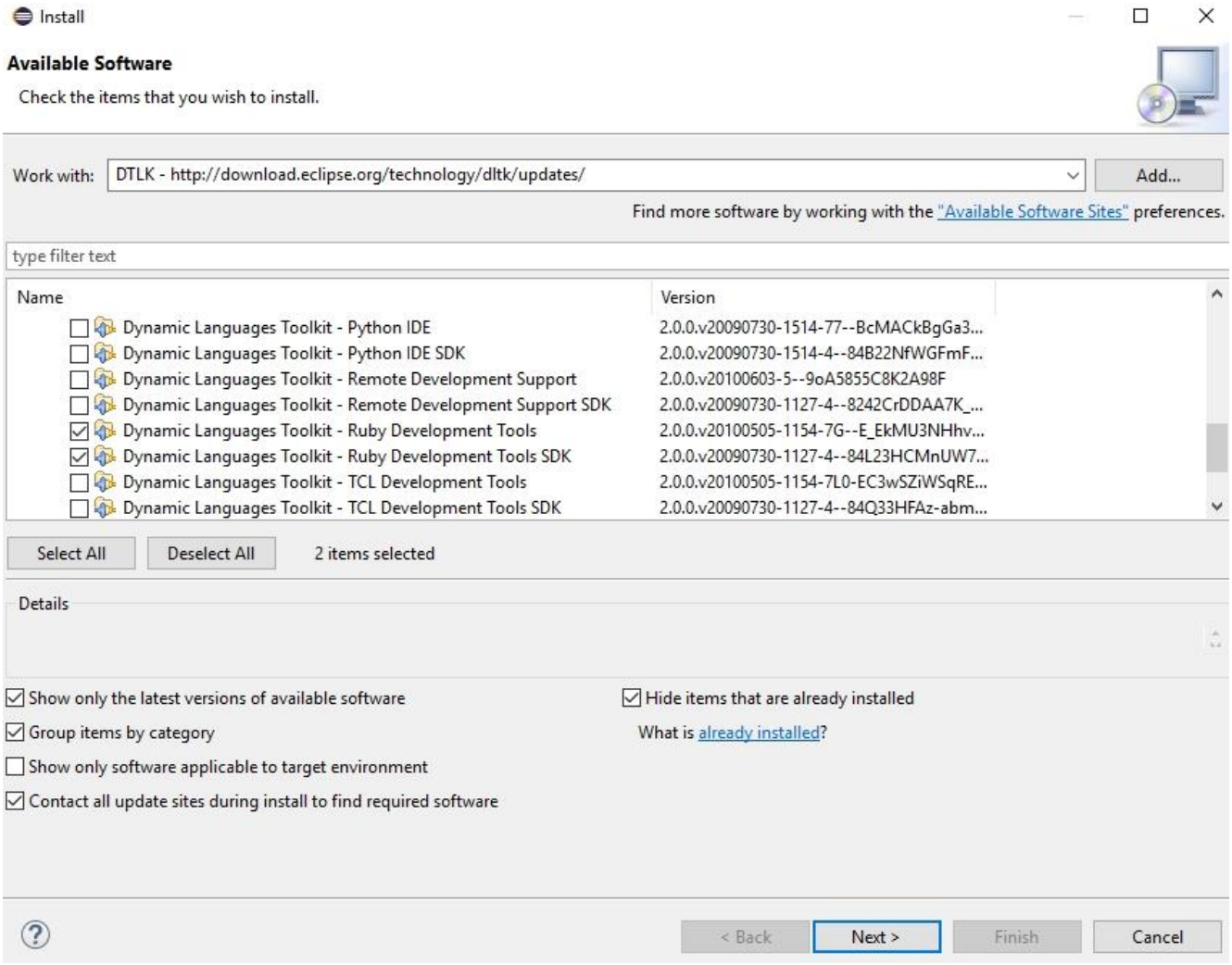

Figura 24. Watir en Eclipse.

5.2.2. Desarrollo del plan de pruebas

A continuación, detallaremos el código fuente de las diferentes clases que hemos creado para desarrollar el plan de pruebas detallado en la sección 4.3 de este documento.

Al igual que con Selenium, hemos creado un enum con los navegadores disponibles:

```
module Browsers

  CHROME = :chrome
  FIREFOX = :firefox
  EDGE = :edge
  IE = :ie
end
```

Ahora, mostramos el código fuente desarrollado para el plan de pruebas en Ruby para esta herramienta:

```
require 'rubygems'
require 'test/unit'
require 'watir'
require 'BrowsersWatir.rb'


class TestWebPage
  extend Test::Unit::Assertions
  timeInicial = Time.now
  initialPage = "Inicio"
  principalPage = "Gestión de Incidencias"

  b = Watir::Browser.start 'http:virtual.lab.inf.uva.es:28032', Browsers::CHROME
  b.window.maximize

  #Inicio Sesión Administrador
  puts "--------------------------"
  puts "-----ADMINISTRADOR--------"
  puts "--------------------------"
  b.text_field(:name,"user").set "admin"
  b.text_field(:id,"password").set "admin"
  b.button(:id,"btnAceptar").click
  b.link(:id, "agregarUser").click
  assert (initialPage==(b.title))
    puts "Inicio Sesión Administrador ----> OK"

  #Se crea al cliente de Pruebas

  b.button(:id,"agregarUsuario").click
  b.text_field(:id,"name").set "ClientePrueba"
  b.text_field(:id,"PrimerApellido").set " "
  b.text_field(:name,"SegundoApellido").set " "
  b.text_field(:id,"login").set "ClientePrueba"
  b.text_field(:name,"password").set "PruebaCliente"
  b.text_field(:id,"Rpassword").set "PruebaCliente"
  b.select_list(:name, "tipo").select("Cliente")
```

```
b.button(:id,"submit").click
puts "Crear usuario Cliente de Prueba ----> OK"


#Se crea al técnico de Pruebas
b.button(:id,"agregarUsuario").click
b.text_field(:id,"name").set "TecnicoPrueba"
b.text_field(:id,"PrimerApellido").set " "
b.text_field(:name,"SegundoApellido").set " "
b.text_field(:id,"login").set "TecnicoPrueba"
b.text_field(:name,"password").set "PruebaTecnico"
b.text_field(:id,"Rpassword").set "PruebaTecnico"
b.select_list(:name, "tipo").select("Técnico")
b.button(:id,"submit").click
puts "Crear usuario Técnico de Prueba ----> OK"
#Cierre Sesión Administrador
b.link(:id, "logout").click
b.alert.ok
assert (principalPage==(b.title))
  puts "Cierre Sesión Administrador ----> OK"

#Inicio sesión Cliente Pruebas.
puts "--------------------------"
puts "--------CLIENTE-----------"
puts "--------------------------"

b.text_field(:name,"user").set "ClientePrueba"
b.text_field(:id,"password").set "PruebaCliente"
b.button(:id,"btnAceptar").click
assert (initialPage==(b.title))
  puts "Inicio Sesión Cliente de Pruebas ----> OK"

b.link(:id, "misIncidencias").click
b.button(:id,"agregarIncidencia").click

b.textarea(:name,"descripcion").set "Incidencia Generada por el Cliente de Pruebas."
time = Time.now
today = time.strftime("%d/%m/%Y")
b.input(:id,"fecha").send_keys today
b.select_list(:name, "prioridad").select("Alta")
b.select_list(:name, "categoria").select("HARDWARE")
b.button(:id,"submit").click
puts "Crear Incidencia -----> OK"
#Cierre Sesión
b.link(:id, "logout").click
b.alert.ok
assert (principalPage==(b.title))
  puts "Cierre Sesión Cliente ----> OK"

#Inicio Sesión Supervisor
puts "--------------------------"
puts "------SUPERVISOR----------"
puts "--------------------------"

b.text_field(:name,"user").set "david"
b.text_field(:id,"password").set "supervisor"
b.button(:id,"btnAceptar").click

assert (initialPage==(session.title))
  puts "Inicio Sesión Supervisor ----> OK"

b.link(:id, "gestionIncidencias").click
b.select_list(:name, "filtroIncidencias").select("Por Cliente")
```

```
  b.select_list(:name, "filtroRoles").select("ClientePrueba")
  b.link(:text, "Ver").click
  b.select_list(:name, "accion").select("Aceptar")
  b.select_list(:id, "elementos").select("TECLADO")
  b.send_keys(:return)
  b.select_list(:name, "tecnico").select("TecnicoPrueba")
  b.button(:id,"submit").click
  puts "Asignar Incidencia ---> OK"
  #Cierre Sesión
  b.link(:id, "logout").click
  b.alert.ok
  assert (principalPage==(b.title))
    puts "Cierre Sesión Supervisor ----> OK"

  #Inicio sesión Técnico Pruebas.
  puts "--------------------------"
  puts "--------TÉCNICO-----------"
  puts "--------------------------"

  b.text_field(:name,"user").set "TecnicoPrueba"
  b.text_field(:id,"password").set "PruebaTecnico"
  b.button(:id,"btnAceptar").click
  assert (initialPage==(session.title))
    puts "Inicio Sesión Técnico de Pruebas ----> OK"

  b.link(:id, "incidencias").click
  b.link(:text, "Ver").click
  b.textarea(:name,"comentarioTecnico").set "Incidencia revisada y solucionada por el
Técnico de Pruebas."
  b.button(:id,"cerrarSolucion").click
  puts "Resolución de Incidencia ----> OK"
  b.link(:id, "logout").click
  b.alert.ok
  assert (principalPage==(session.title))
    puts "Cierre Sesión Técnico de Pruebas ----> OK"

  #Inicio Sesión Supervisor
  puts "--------------------------"
  puts "------SUPERVISOR----------"
  puts "--------------------------"
  b.text_field(:name,"user").set "david"
  b.text_field(:id,"password").set "supervisor"
  b.button(:id,"btnAceptar").click

  assert (initialPage==(session.title))
  puts "Inicio Sesión Supervisor ----> OK"

  b.link(:id, "gestionIncidencias").click
  b.select_list(:name, "filtroIncidencias").select("Por Técnico")
  b.select_list(:name, "filtroRoles").select("TecnicoPrueba")
  b.link(:text, "Ver").click
  b.button(:name,"cerrar").click
  puts "Cierre de Incidencia ---> OK"
  b.link(:id, "logout").click
  b.alert.ok
  assert (principalPage==(b.title))
    puts "Cierre Sesión Supervisor ----> OK"

  b.close
  timeFinal = Time.now
  timeEjecucion = timeFinal - timeInicial
  puts "Tiempo ejecución: "+timeEjecucion.to_s+" Segundos"
  end
```

5.2.3. Resultados

A continuación, veremos los resultados que obtenemos por la terminal de eclipse después de la ejecución del plan de pruebas con Watir:

```
        -------------------------
        -----ADMINISTRADOR--------
        -------------------------
  Inicio Sesión Administrador ----> OK
Crear usuario Cliente de Prueba ----> OK
Crear usuario Técnico de Prueba ----> OK
  Cierre Sesión Administrador ----> OK
        -------------------------
        --------CLIENTE-----------
        -------------------------
Inicio Sesión Cliente de Pruebas ----> OK
       Crear Incidencia -----> OK
     Cierre Sesión Cliente ----> OK
        -------------------------
        ------SUPERVISOR----------
        -------------------------
    Inicio Sesión Supervisor ----> OK
       Asignar Incidencia ---> OK
    Cierre Sesión Supervisor ----> OK
        -------------------------
        --------TÉCNICO-----------
        -------------------------
Inicio Sesión Técnico de Pruebas ----> OK
    Resolución de Incidencia ----> OK
Cierre Sesión Técnico de Pruebas ----> OK
        -------------------------
        ------SUPERVISOR----------
        -------------------------
    Inicio Sesión Supervisor ----> OK
      Cierre de Incidencia ---> OK
    Cierre Sesión Supervisor ----> OK

   Tiempo ejecución: 39.756909 Segundos
```

5.3.Capibara

Capibara es un marco de automatización basada en web que se utiliza para la creación de pruebas funcionales que simulan cómo los usuarios podrían interactuar con la aplicación.

Al igual que Watir, Capibara es una biblioteca / gema Ruby construida con el objetivo de ser utilizado en la parte de un controlador basado en la web subyacente.

Ofrece una conexión DSL fácil de usar (Domain Specific Language) que se utiliza para describir acciones que son ejecutadas por el controlador Web subyacente, como Chrome, Firefox, Edge, Internet Explorer.

Cuando la página se carga mediante el DSL (y el conductor Web subyacente), Capibara intentará localizar el elemento relevante en el DOM (Document Object Model) y ejecutar la acción, como botón de click, enlace, etc.

Los drivers web que soporta Capibara son los siguientes:

- Rack: test: Por defecto, este será el driver web que utilicemos con Capibara. Este driver es considerado uno de los más rápidos, pero no tiene soporte JavaScript ni se puede acceder a por recursos HTTP fuera de la aplicación.

- Selenium-WebDriver: Es compatible con Capibara. Este driver sí tiene soporte JavaScript y puede acceder a los recursos HTTP de fuera de la aplicación. Para utilizar este driver es necesario agregar la siguiente línea de código:

```
Capybara.register_driver :selenium
```

- Capibara-webkit: Para pruebas en las que no necesitemos el soporte de JavaScript, utilizaremos este driver, que usa QtWebKit, un motor renderizado de contenido web y Open source. Además, este driver es significativamente más rápido que Selenium, ya que este no carga el navegador en su totalidad. [18] Para utilizar este driver es necesario agregar la siguiente línea de código:

```
Capybara.register_driver :wekit
```

5.3.1. Instalación

A continuación, vamos a instalar esta herramienta, antes de nada, debemos de instalar Ruby, ya que es necesario para que funcione, puesto que es una gema a instalar. La explicación de la instalación de Ruby la podemos ver detalladamente en el Anexo II.

Una vez que hemos seguido los pasos indicados en el anexo, y, por tanto, instalado correctamente Ruby, procederemos a instalar Watir, para ello tenemos que escribir en la terminal de Ruby el siguiente comando:

```
gem install Capibara
```

5.3.2. Desarrollo del plan de pruebas

A continuación, detallaremos el código fuente de las diferentes clases que hemos creado para desarrollar el plan de pruebas detallado en la sección 4.3 de este documento.

Al igual que con Selenium, hemos creado un enum con los navegadores disponibles:

```
module Browsers
CHROME = :chrome
FIREFOX = :firefox
EDGE = :edge
IE = :ie
End
```

A continuación, se muestra la clase de pruebas del núcleo de la aplicación.

```
require 'capybara'
require 'test/unit'
require 'BrowsersCapybara.rb'


class TestClass < Test::Unit::TestCase
extend Test::Unit::Assertions
  timeInicial = Time.now
  Capybara.register_driver :selenium do |app|
  Capybara::Selenium::Driver.new(app, :browser => Browsers::CHROME)
  end

  session = Capybara::Session.new(:selenium)
  session.visit("http://virtual.lab.inf.uva.es:28032")
  session.current_window.maximize

  initialPage = "Inicio"
  principalPage = "Gestión de Incidencias"
  time = Time.now
  today = time.strftime("%d/%m/%Y")
  #Administrador
  puts "--------------------------"
  puts "-----ADMINISTRADOR--------"
  puts "--------------------------"

  session.fill_in('user', :with => 'admin')
  session.fill_in('password', :with => 'admin')
  session.click_button('btnAceptar')
  sleep(1)

  assert (initialPage==(session.title))
  puts "Inicio Sesión Administrador ----> OK"

  session.click_link('agregarUser')
  sleep(1)
  #Creamos Cliente  Pruebas
  session.click_button('agregarUsuario')
  session.fill_in('name', :with => 'ClientePrueba')
  session.fill_in('PrimerApellido', :with => '  ')
  session.fill_in('SegundoApellido', :with => '  ')
```

```
session.fill_in('login', :with => 'ClientePrueba')
session.fill_in('password', :with => 'PruebaCliente')
session.fill_in('Rpassword', :with => 'PruebaCliente')
session.select('Cliente',:from=>'tipo')
session.click_button('submit')
puts "Crear usuario Cliente de Prueba ----> OK"

#Creamos Tecnico  Pruebas
session.click_button('agregarUsuario')
session.fill_in('name', :with => 'TecnicoPrueba')
session.fill_in('PrimerApellido', :with => '  ')
session.fill_in('SegundoApellido', :with => '  ')
session.fill_in('login', :with => 'TecnicoPrueba')
session.fill_in('password', :with => 'PruebaTecnico')
session.fill_in('Rpassword', :with => 'PruebaTecnico')
session.select('Técnico',:from=>'tipo')
session.click_button('submit')
puts "Crear usuario Técnico de Prueba ----> OK"
session.click_link('logout')
session.accept_alert
assert (principalPage==(session.title))
puts "Cierre Sesión Administrador ----> OK"

puts "Crear usuario Técnico de Prueba ----> OK"

# Cliente PRUEBAS
puts "--------------------------"
puts "--------CLIENTE-----------"
puts "--------------------------"

session.fill_in('user', :with => 'ClientePrueba')
session.fill_in('password', :with => 'PruebaCliente')
session.click_button('btnAceptar')
assert (initialPage==(session.title))
puts "Inicio Sesión Cliente de Pruebas ----> OK"

session.click_link('misIncidencias')
session.click_button('agregarIncidencia')
session.fill_in('descripcion', :with => 'Incidencia creada por el Cliente de Test')
session.fill_in('fecha', :with => today)
session.select('Alta',:from=>'prioridad')
session.select('HARDWARE',:from=>'categoria')
session.click_button('submit')
puts "Crear Incidencia -----> OK"
session.find_by_id('Array')
session.click_link('logout')
session.accept_alert
assert (principalPage==(session.title))
puts "Cierre Sesión Cliente ----> OK"
sleep(1)

#Supervisor
puts "--------------------------"
puts "------SUPERVISOR----------"
puts "--------------------------"
session.fill_in('user', :with => 'david')
session.fill_in('password', :with => 'supervisor')
session.click_button('btnAceptar')

assert (initialPage==(session.title))
puts "Inicio Sesión Supervisor ----> OK"

session.click_link('gestionIncidencias')
```

```
  session.select('Por Cliente',:from=>'filtroIncidencias')
  session.select('ClientePrueba',:from=>'filtroRoles')
  session.click_link('Ver')
  session.select('Aceptar',:from=>'accion')
  session.select('TECLADO',:from=>'elementos')
  session.select('TecnicoPrueba',:from=>'tecnico')
  session.click_button('submit')
  puts "Asignar Incidencia ---> OK"
  session.click_link('logout')
  session.accept_alert
  assert (principalPage==(session.title))
  puts "Cierre Sesión Supervisor ----> OK"

  #Tecnico
  puts "--------------------------"
  puts "--------TÉCNICO-----------"
  puts "--------------------------"
  session.fill_in('user', :with => 'TecnicoPrueba')
  session.fill_in('password', :with => 'PruebaTecnico')
  session.click_button('btnAceptar')
  assert (initialPage==(session.title))
  puts "Inicio Sesión Técnico de Pruebas ----> OK"
  session.click_link('incidencias')
  session.click_link('Ver')
  session.fill_in('comentarioTecnico', :with => 'Incidencia revisada y solucionada por el
Técnico de Pruebas.')
  session.click_button('cerrarSolucion')
  puts "Resolución de Incidencia ----> OK"
  session.click_link('logout')
  session.accept_alert
  assert (principalPage==(session.title))
  puts "Cierre Sesión Técnico de Pruebas ----> OK"
  sleep(1)

  #Supervisor
  puts "--------------------------"
  puts "------SUPERVISOR----------"
  puts "--------------------------"
  session.fill_in('user', :with => 'david')
  session.fill_in('password', :with => 'supervisor')
  session.click_button('btnAceptar')
  #assert_equal("Inicio",session.title)
  puts "Inicio Sesión Supervisor ----> OK"
  session.click_link('gestionIncidencias')
  session.select('Por Técnico',:from=>'filtroIncidencias')
  session.select('TecnicoPrueba',:from=>'filtroRoles')
  session.click_link('Ver')
  session.click_button('submit')
  puts "Cierre de Incidencia ---> OK"
  session.click_link('logout')
  session.accept_alert
  assert (principalPage==(session.title))
  puts "Cierre Sesión Supervisor ----> OK"

  timeFinal = Time.now
  timeEjecucion = timeFinal - timeInicial
  puts "Tiempo ejecución: "+timeEjecucion.to_s+" Segundos"
end
```

5.3.3. Resultados

A continuación, veremos los resultados que obtenemos por la terminal de eclipse después de la ejecución del plan de pruebas con Capibara:

```
        --------------------------
        -----ADMINISTRADOR--------
        --------------------------
  Inicio Sesión Administrador ----> OK
Crear usuario Cliente de Prueba ----> OK
Crear usuario Técnico de Prueba ----> OK
  Cierre Sesión Administrador ----> OK
Crear usuario Técnico de Prueba ----> OK
        --------------------------
        --------CLIENTE-----------
        --------------------------
Inicio Sesión Cliente de Pruebas ----> OK
        Crear Incidencia -----> OK
     Cierre Sesión Cliente ----> OK
        --------------------------
        ------SUPERVISOR----------
        --------------------------
    Inicio Sesión Supervisor ----> OK
        Asignar Incidencia ---> OK
    Cierre Sesión Supervisor ----> OK
        --------------------------
        --------TÉCNICO-----------
        --------------------------
Inicio Sesión Técnico de Pruebas ----> OK
    Resolución de Incidencia ----> OK
Cierre Sesión Técnico de Pruebas ----> OK
        --------------------------
        ------SUPERVISOR----------
        --------------------------
    Inicio Sesión Supervisor ----> OK
       Cierre de Incidencia ---> OK
    Cierre Sesión Supervisor ----> OK
   Tiempo ejecución: 29.926923 Segundos´
```

5.4. Cucumber

Cucumber es una herramienta software utilizada para ejecutar de forma automática pruebas de aceptación en entornos web. A diferencia de las otras herramientas vistas en cada una de las secciones de este capítulo se basa en la utilización de un nuevo estilo BDD (Behavior-Driver Developement), de esta forma podremos realizar descripciones funcionales en texto plano como pruebas software automatizadas.

En las siguientes secciones, vamos a profundizar más sobre este proceso de desarrollo software.

5.4.1. BDD, Gherkin

BDD es un proceso de desarrollo software que más allá del desarrollo de software, más allá del Testing, realmente, nació con el objetivo de evitar todas las malas repercusiones que conllevaban los malos entendimientos en un proyecto.

A continuación, podemos ver un tweet que publicó Dan Norton en Twitter sobre el BDD, publicado el 26 de mayo del 2013:

"#BDD in a tweet: Using Examples at multiple levels to create a shared understanding and surface uncertainty to deliver software that maters"

Dan North

La idea que nos transmite a través de este tweet, es que cada uno de los roles del proyecto y de los clientes aporte su conocimiento para que entre todos se pueda crear el software que importa, es decir, el que se necesita, satisfaciendo las necesidades marcadas por el cliente.

En la Figura 25, vemos de forma gráfica que, para poder diseñar, realizar y probar el software que se va a realizar de una forma colaborativa es necesario que, en esta colaboración, intervengan al menos tres roles dentro de un proyecto: negocio, desarrollo y Testing.

Figura 25. Behavior-Driver Development [14]

Todo este procedimiento, para aplicarlo de una forma más cómoda se ha definido un DSL (Domain Specific Language) a este lenguaje se le ha bautizado como Gherkin.

Como veníamos comentando la característica más destacada de este lenguaje es muy similar al habitual, de hecho, soporta más de 60 idiomas [13], ya que su principal objetivo es, que más allá de la implementación, negocio, desarrolladores y tester sepan de forma conjunta cómo va a ser el software a crear, que características va a tener y cómo va a ser su funcionalidad.

Para poder poner en práctica este lenguaje es necesario tener claro los siguientes conceptos:

- Feature: Indica el nombre de la funcionalidad que vamos a probar. Debe ser un título claro y explícito. En base a esto realizaremos los escenarios con sus pruebas concretas.

- Scenario: Es una descripción del escenario que vamos a probar.

- Given: Aporta el contexto para el escenario en el que se va a ejecutar el test, tales como: dónde se ejecuta el test, prerrequisitos en los datos. Además, incluye los pasos necesarios para poner al sistema en el estado que se desea probar.

- When: Especifica el conjunto de acciones que lanzan el test, detallando la interacción del usuario que acciona la funcionalidad que deseamos testear.

- Then: Indica el resultado esperado en el test. Observaremos los cambios en el sistema y vemos si son los deseados o no en base a los especificados.

Lo ideal, es probar en diferentes escenarios la misma funcionalidad, de esta forma realizaremos las pruebas que posteriormente automatizaremos. A continuación, vamos a ver como instalar Cucumber y por tanto a poner en práctica este lenguaje.

5.4.2. Instalación

A continuación, vamos a detallar cada uno de los pasos necesarios para instalar Cucumber.

Para programar, cada uno de los casos de prueba hemos decidido utilizar Ruby, por esta razón, lo tendremos que instalar como una nueva gema de la herramienta, así que tenemos que ejecutar el siguiente comando en la terminal de Ruby:

```
gem install cucumber
```

Para terminar, tendremos que instalar ansicon, es una biblioteca que permitirá a la terminal mostrar más colores, por lo que deberemos actualizarla por encima de la versión 1.3.1 para evitar un warning y tener una mejor calidad y color en nuestras pruebas.

Para ello tenemos que acceder a: https://github.com/adoxa/ansicon/downloads , nos descargamos la última versión, en zip, y lo descomprimimos. Seguidamente tendremos que copiar la carpeta x86 o x64, dependiendo de la arquitectura del procesador de nuestra computadora a la carpeta bin de nuestra PATH de Ruby.

En la Figura 26. Podemos ver la salida en la terminal después de que se ejecute el comando anteriormente mostrado, y podemos comprobar que todo se descargó correctamente.

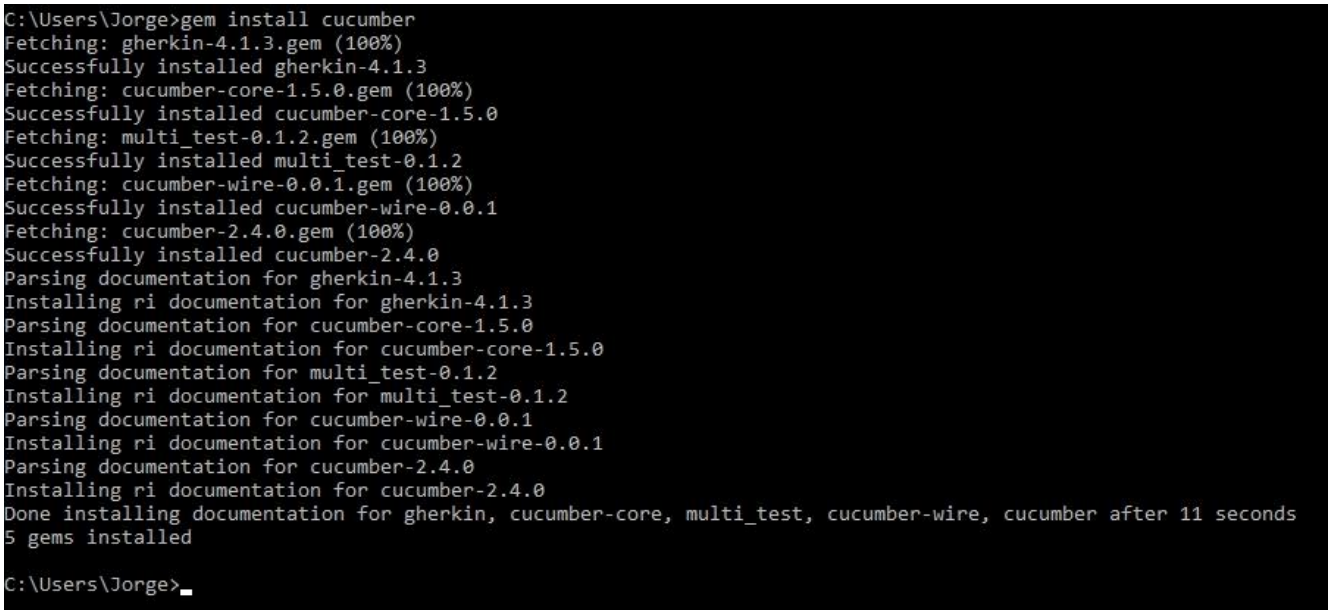

Figura 26. Instalación de Cucumber.

5.4.3. Desarrollo del plan de pruebas

En el desarrollo de pruebas de esta herramienta tendremos que generar un proyecto Cucumber, debe tener la estructura de carpetas que aparece en la Figura 27, donde veremos que tenemos que tener dentro del proyecto una carpeta con el nombre de features, a este nivel tendremos los features correspondientes y después tendremos dos carpetas cuyos nombres serán: step_definitions y support y se usarán para la implementación de cada uno de los pasos y para el arranque del navegador.

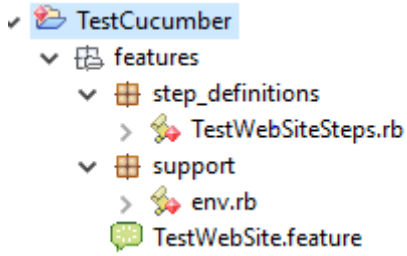

Figura 27. Estructura de Carpetas Proyecto Cucumber

A continuación, vemos el código fuente del TestWebSite.feature:

```
Feature: Smoke Test Web
Scenario: Case Test Administrator

	Given The Administrator is on Event manager
	When  The Administrator enter "admin"
	And     The Administrator create user client with login "ClienteTest"
	And     The Administrator create user technical with login "TecnicoTest"
	And  The Administrator logout

Scenario: Case Test Client
	Given The Client is on Event manager
	When The Client enter "ClienteTest"
	And The Client records incidence of "HARDWARE"
	And The Client logout

Scenario: Case Test Checker
	Given The Checker is on Event manager
	When The Checker enter "david"
	And The Checker accepts the incidence
	And The Checker logout

Scenario: Case Test Technical
	Given The Technical is on Event manager
	When The Technical enter "TecnicoTest"
	And The Technical resolves the incidence of "ClienteTest"
	And The Technical logout

Scenario: Case Test Checker Close Incidence
	Given The Checker is on Event manager
	When The Checker enter "david"
	And The Checker looking for incidence and close
	And The Checker logout
```

A continuación, el código Fuente de la clase TestWebSiteStpes.java, esta contendrá la lógica de cada una de las entradas del fichero . feature

---

```
####################
# ADMINISTRATOR
####################
Given /^The Administrator is on Event manager$/ do
  visit('http://www.virtual.lab.inf.uva.es:28032/')
end

When /^The Administrator enter "([^"]*)"$/ do |user|
  fill_in('user',:with => user)
  fill_in('password',:with => "admin")
  click_button("btnAceptar")

end
And /^The Administrator create user client with login "([^"]*)"$/ do |userClient|
  click_link('agregarUser')
  click_button('agregarUsuario')
  fill_in('name', :with => userClient)
  fill_in('PrimerApellido', :with => '  ')
  fill_in('SegundoApellido', :with => '  ')
  fill_in('login', :with => userClient)
  fill_in('password', :with => userClient)
  fill_in('Rpassword', :with => userClient)
  select('Cliente',:from=>'tipo')
  click_button('submit')
 end
And /^The Administrator create user technical with login "([^"]*)"$/ do |userTechnical|
  click_link('agregarUser')
  click_button('agregarUsuario')
  fill_in('name', :with => userTechnical)
  fill_in('PrimerApellido', :with => '  ')
  fill_in('SegundoApellido', :with => '  ')
  fill_in('login', :with => userTechnical)
  fill_in('password', :with => userTechnical)
  fill_in('Rpassword', :with => userTechnical)
  select('Técnico',:from=>'tipo')
  click_button('submit')
  sleep(1)
 end
 And /^The Administrator logout$/ do
      click_link('logout')
      accept_alert
 end
####################
# CLIENT
####################
Given /^The Client is on Event manager$/ do
  visit('http://www.virtual.lab.inf.uva.es:28032/')
end
 When /^The Client enter "([^"]*)"$/ do |userClient|
  fill_in('user',:with => userClient)
    fill_in('password',:with => userClient)
    click_button("btnAceptar")
end

And /^The Client records incidence of "([^"]*)"$/ do |typeIncidence|
    time = Time.now
    today = time.strftime("%d/%m/%Y")
    click_link('misIncidencias')
```

```
    click_button('agregarIncidencia')
    fill_in('descripcion', :with => 'Incidencia creada por el Cliente de Test')
    fill_in('fecha', :with => today)
    select('Alta',:from=>'prioridad')
    select(typeIncidence,:from=>'categoria')
    click_button('submit')
    sleep(1)
end

And /^The Client logout$/ do
        click_link('logout')
        accept_alert
end
####################
# CHECKER
####################
Given /^The Checker is on Event manager$/ do
  visit('http://www.virtual.lab.inf.uva.es:28032/')
end

When /^The Checker enter "([^"]*)"$/ do |userChecker|
    fill_in('user',:with => userChecker)
    fill_in('password',:with => "supervisor")
    click_button("btnAceptar")
end

And /^The Checker accepts the incidence"$/ do

    click_link('gestionIncidencias')
    click_link('Ver')
    select('Aceptar',:from=>'accion')
    sleep(1)
    select('TECLADO',:from=>'elementos')
    select("TecnicoTest",:from=>'tecnico')
    click_button('submit')
end

And /^The Checker looking for incidence and close$/ do
    click_link('gestionIncidencias')
    select('Por Técnico',:from=>'filtroIncidencias')
    select("TecnicoTest",:from=>'filtroRoles')
    click_link('Ver')
    click_button('submit')
end

And /^The Checker logout$/ do
  click_link('logout')
  accept_alert
end
####################
# TECHNICAL
####################
Given /^The Technical is on Event manager$/ do
  visit('http://www.virtual.lab.inf.uva.es:28032/')
end

When /^The Technical enter "([^"]*)"$/ do |userTechnical|
      fill_in('user',:with => userTechnical)
      fill_in('password',:with => "TecnicoTest")
      click_button("btnAceptar")
end

And /^The Technical resolves the incidence of "([^"]*)"$/ do |userClient|
```

```
    click_link('incidencias')
    click_link('Ver')
    fill_in('comentarioTecnico', :with => 'Incidencia revisada y solucionada por el
Técnico de Pruebas.')
    click_button('cerrarSolucion')
end
And /^The Technical logout$/ do
  click_link('logout')
  accept_alert
end
```

Ahora, el código Fuente de la clase env.java, esta contendrá el arranque del navegador.

```
require 'capybara/cucumber'
require 'selenium-webdriver'
require 'uri'

# Tell FireFox to use proxy settings
if ENV['http_proxy']
  Capybara.register_driver :selenium do |app|
    profile = Selenium::WebDriver::Firefox::Profile.new

    uri = URI(ENV['http_proxy'])

    profile["network.proxy.type"] = 1 # manual proxy config
    profile["network.proxy.http"] = uri.host
    profile["network.proxy.http_port"] = uri.port

    if ENV['https_proxy']
      uri = URI(ENV['https_proxy'])
      profile["network.proxy.https"] = uri.host
      profile["network.proxy.https_port"] = uri.port
    end

    Capybara::Selenium::Driver.new(app, :profile => profile)
  end
end

Capybara.default_driver = :selenium
```

5.4.4. Resultados

Por último, vamos a ver los resultados de la ejecución de este plan de pruebas, lo podemos ver en la Figura28.

Veremos que tenemos que: abrir una terminal y, acceder a nuestra ruta local, donde alojemos el proyecto en eclipse, en mi caso la siguiente ruta:

```
C :\Users\Jorge\workspace
```

Una vez realizado tendremos que ejecutar:

```
cucumber
```

Y podremos ver el resultado en la Figura28. Como podemos ver son los resultados más detallados, ya que además de tener un OK en cada una de las pruebas, tenemos una descripción funcional de lo que se hace en cada caso.

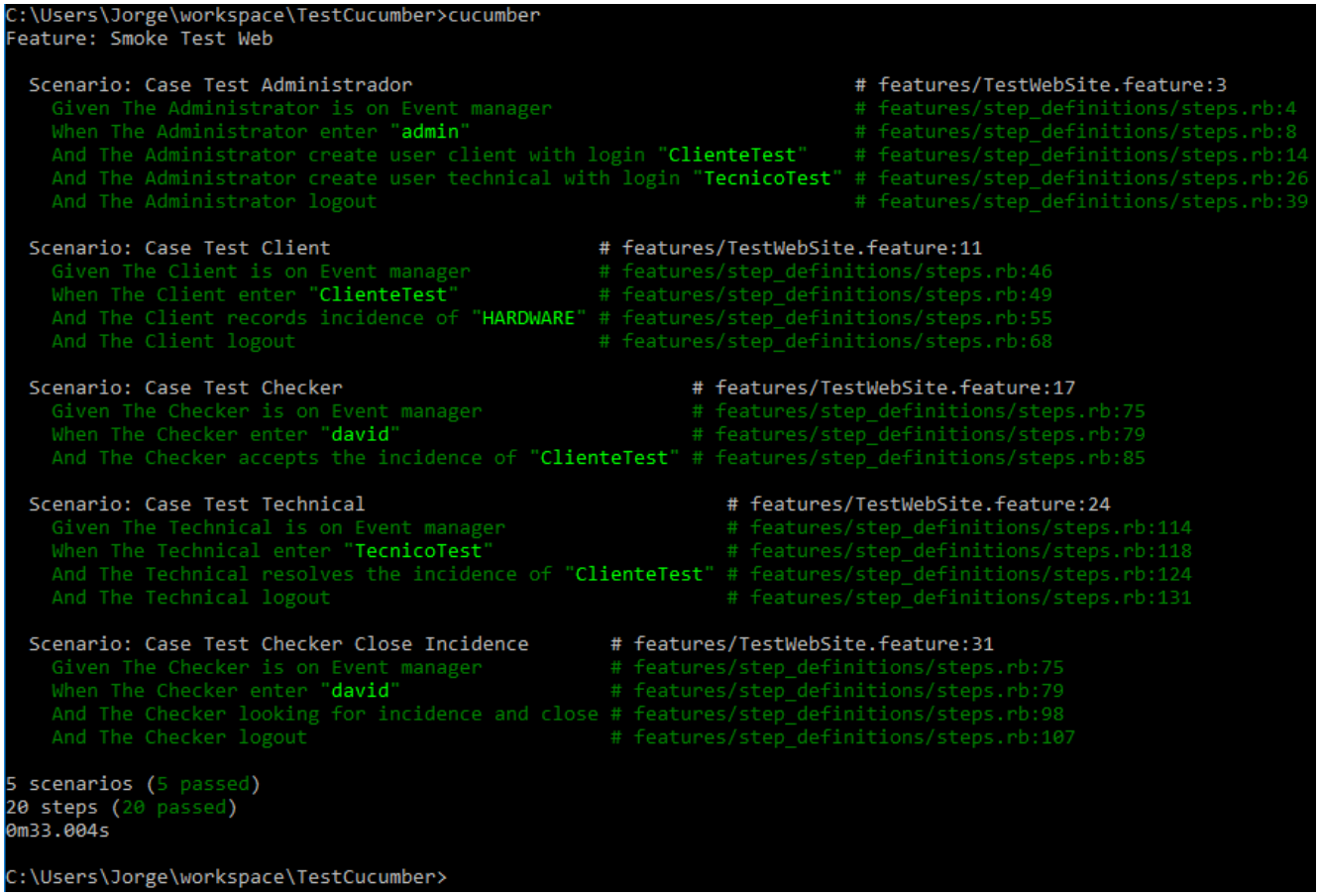

Figura 28. Resultados Cucumber

Capítulo VI: Resultados y Comparativa de las Herramientas

6.1. Resultados

A continuación, hemos llegado al punto en el que se comentará las ventajas obtenidas y por tanto los beneficios que hemos tenido al utilizar este tipo de pruebas, empezaremos diciendo que gracias a cada una de ellas hemos realizado el objetivo principal de este proyecto que era realizar la automatización de pruebas de aceptación funcionales o pruebas de regresión en esta página web en la que se ha partido de la hipótesis que ira sufriendo varios cambios a lo largo de la existencia de la misma.

Como hemos podido ver en el Capítulo V, en cada una de las herramientas había un subapartado destinado a los resultados. Por tanto, podemos comprobar que, se ha obtenido un resultado homogéneo en todas ellas, ya que vemos el resultado de cada una de las acciones que forman parte del núcleo de la aplicación realizadas correctamente, esto es, que recibieron un OK.

Este es el principal objetivo de todas las herramientas y del uso de cada una de ellas a la hora de realizar la ejecución de los planes de pruebas, ya que cada una de ellas ha sido seleccionada para satisfacer este requisito. Por lo que, partimos de una base esencial para poder realizar la comparativa de cada una de ellas.

En estos resultados cabe destacar el uso de la automatización siempre y cuando sea beneficioso desde el punto de vista de coste y tiempo de empleo en el desarrollo de las pruebas y de los diferentes tipos de pruebas, esto queda reflejado en el cono de Crohn en el que se veía de forma gráfica este caso.

Para entender mejor esta explicación, vamos a realizar un ejemplo para que se asiente mejor la idea que queremos transmitir, ya que partimos de un punto en común de todas las herramientas, y es que con todas ellas obtenemos un resultado similar, ya que en mayor o menor detalle nos indican un OK en la ejecución de todas las pruebas.

Por tanto, como decíamos a lo largo de este documento, realizar la automatización de pruebas en un entorno web es interesante a largo plazo, ya que ganaremos tanto tiempo como dinero.

Una vez indicado todo esto, comenzaremos indicando que todas las herramientas tardan alrededor de 30 segundos en realizar el plan de pruebas definido (se puede ver en los resultados obtenidos en el Capítulo anterior) y en entregar todos los resultados, mientras que la realización manual de este mismo plan de pruebas es aproximadamente de 2,5 minutos (teniendo en cuenta que el usuario conoce la aplicación, es decir experimentado), sin realizar un informe de lo que ha podido pasar en cada una de las pruebas.

Teniendo en cuenta este detalle, la realización de un informe similar al que obtenemos con la ejecución de la pruebas, para un usuario experimentado, será de unos 10 minutos. Por tanto, la realización del plan de pruebas de forma manual será de unos 12,5 minutos para un usuario experimentado.

De aquí obtenemos, que el proceso automático es significativamente más rápido que el manual, por tanto, esta la gran diferencia, respecto al tiempo y dinero se encuentra en el desarrollo de estas pruebas en la herramienta que deseemos y en la licencia, si tuviese, de la misma.

Una vez explicado todo este proceso, podremos concluir que empezará a sernos rentable la automatización cuando tiempo que dediquemos a repetir estas pruebas por desarrollos, actualizaciones, regresiones, etc.… supere el tiempo invertido en desarrollar las pruebas de la misma.

En la Figura 29, podemos ver la gráfica correspondiente a un caso que pudiera ser perfectamente real, en el que se tuviese que repetir las pruebas en la aplicación software tres veces por semana por desarrollos y pruebas por regresión y una más al mes por actualizaciones.

También tendremos que tener en cuenta que el tiempo en desarrollar las pruebas es de 30h por un programador experimentado. Por tanto, vemos que a partir de los 11.5 meses estaríamos amortizando el tiempo y dinero empleado en la automatización. Además, hay que tener en cuenta que esta es una aplicación pequeña, donde el número de pruebas no es muy elevado, pero en aplicaciones más grandes y con más número de pruebas, las cantidades de tiempo y dinero crecerían proporcionalmente.

Como conclusión de la gráfica, es rentable automatizar desde el punto de vista económico si vamos a utilizar durante bastante tiempo la aplicación. Esto desde el punto de vista económico, pero desde el de calidad, quizá nos compense más por este caso que por el de tiempo, ya que en la calidad y en otras muchas características que nos proporciona la automatización encontremos la gran diferencia.

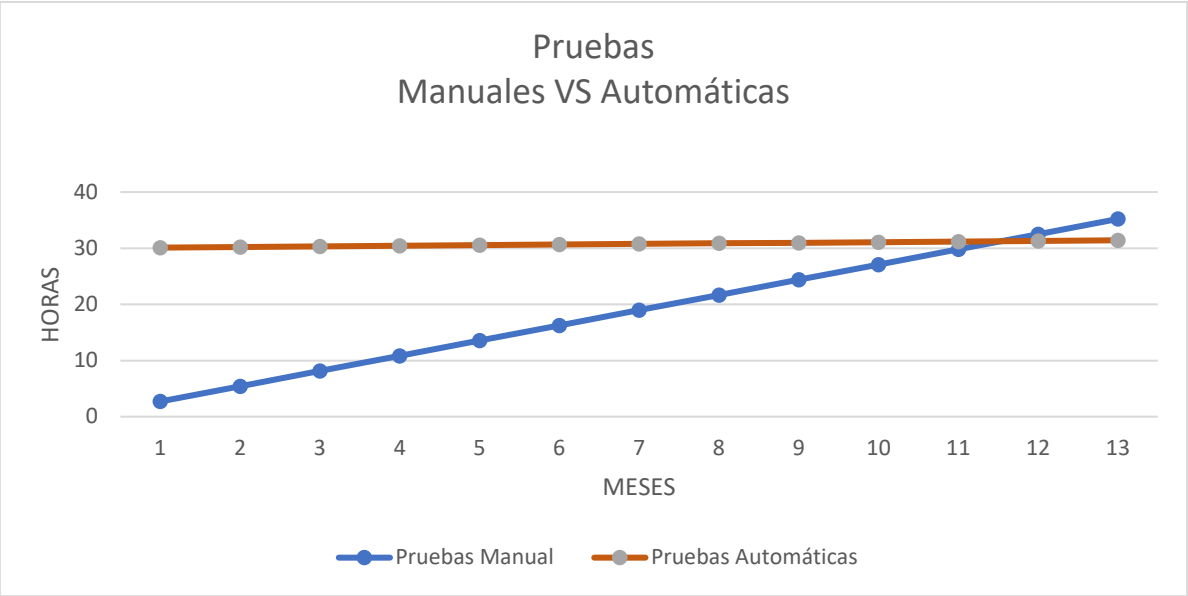

Figura 29. Pruebas Manuales VS Automáticas.

6.2. Comparativa de las herramientas.

Ahora nos sumergiremos en las diferencias que hay entre las diferentes herramientas, cabe destacar que todas partían de un fin común, la ejecución del plan de pruebas realizado, para la comprobación de la funcionalidad del núcleo de la aplicación. Como hemos visto antes, hemos obtenido unos resultados homogéneos.

Para establecer una buena comparativa evidentemente depende del nivel de experiencia que tienen los diferentes usuarios para empezar a utilizar este tipo de herramientas.

Por tanto, partiremos de mi caso personal, en el que fui un programador sin experiencia en el desarrollo de pruebas con estas herramientas y poco a poco iba cogiendo experiencia y, por tanto, realizando los planes de pruebas con mayor dominio y eficazmente.

Como podemos ver en la tabla 45, es una comparativa de cada una de las herramientas que se han utilizado en este proyecto. Podemos ver que hay muchas características que son iguales en todas las herramientas, en otros casos vemos como hay herramientas que destacan o que realmente tienen características que son superiores a la demás.

Cabe destacar, las diferencias desde el punto de vista de desarrollo, esto es, la facilidad o precisión que tienen las diferentes herramientas para acceder o manipular los diferentes elementos del DOM.

Podemos encontrar dos casos, cuando tenemos la posibilidad de hacer distinción entre el atributo del elemento que deseemos o el caso contrario, que indiquemos uno de ellos y sea la propia herramienta que lo diferencie, este caso se da en capibara, característica que crea distinción respecto de las otras herramientas.

Otro detalle que hay que comentar, es el uso de los diferentes navegadores, en este caso todas las herramientas se posicionan bastante bien, esto es, tienen la posibilidad de usar un gran número de navegadores, pero esto es posible a que usan Selenium-WebDriver, si no fuese así solo usurarían navegadores como Internet Explorer o Mozilla Firefox. Aun así, Selenium sigue superando a estas herramientas, ya que puede usar el navegador Opera, ya que dispone drivers para el mismo, mientras que las gemas de Ruby aun no esta versión incluida.

	SELENIUM	CAPIBARA	WATIR	CUCUMBER
OPENSOURCE	SI	SI	SI	SI
MULTIPLATAFORMA	SI	SI	SI	SI
POSIBILIDAD DE ESCRIBIR LAS PRUEBAS EN VARIOS IDIOMAS	NO	NO	NO	SI
AUTOMATIZACIÓN	SI	SI	SI	SI
TIPOS PRUEBAS	Aceptación Regresión Funcionales Unitarias Integración Sistema	Aceptación Regresión Funcionales Unitarias	Aceptación Regresión Funcionales Unitarias Integración	Aceptación Funcionales Unitarias
NAVEGADORES	INTERNET EXPLORER MOZILA FIREFOX GOOGLE CHROME SAFARI OPERA	INTERNET EXPLORER MOZILA FIREFOX GOOGLE CHROME SAFARI	INTERNET EXPLORER MOZILA FIREFOX GOOGLE CHROME SAFARI	INTERNET EXPLORER MOZILA FIREFOX GOOGLE CHROME SAFARI
FACIL LEER	MEDIO-BAJO	MEDIO-BAJO	MEDIO-BAJO	ALTA
USABILIDAD	MEDIA	MEDIA	MEDIA	ALTA
ENTORNO	WEB	WEB	WEB	WEB
LENGUAJES PROGRAMACIÓN	Java, C#, Ruby, Groovy, Perl, Php, Python…)	Ruby	Ruby	Java, Ruby

	SELENIUM	CAPIBARA	WATIR	CUCUMBER
ESCRITURA CASOS DE PRUEBA	DESARROLLO TESTERS	DESARROLLO TESTERS	DESARROLLO TESTERS	DESARROLLO TESTERS NEGOCIO
FACILIDAD DE APRENDIZAJE	MEDIA	ALTA	ALTA	ALTA – MUY ALTA

Tabla 45. Comparativa Herramientas de Automatización

Otra de las cosas que tenemos que destacar, es el tiempo, en horas, de realización de cada uno de los planes de pruebas en las difertentes herramientas. En la tabla 46, podemos comprobar que cada vez es menor, ya que, la experiencia que iba adquiriendo en el desarrollo de las pruebas es fundamental para realizarlas de forma más ágil.

SELENIUM	CAPIBARA	WATIR	CUCUMBER
33 h	31 h	30 h	25 h

Tabla 46. Tiempo dedicado al desarrollo de los casos de Prueba

Capítulo VII: Plan de Gestión de Proyecto

7.1. Seguimiento

En cuanto al seguimiento del proyecto, se cumplieron los dos primeros sprints, a partir de ahí, en el mes de marzo, por situación del proyecto en el mundo laboral, tuve que viajar a Madrid, en varias ocasiones, lo que me imposibilitó continuar con las tareas que se habían estimado realizar este mes.

Por esta razón, se ha pospuesto la fecha de finalización del proyecto, inicialmente se estimó que se iba a terminar dicho proyecto el 2 de junio, debido a este problema que surgió se pospone a fecha de finalización a 3 de julio.

Como bien preveíamos en la gestión de riesgos, capítulo dos de este documento, el riesgo número 4 es por el que nos hemos visto afectados. Estimamos en considerar con holgura los tiempos de cada una de las tareas, pero no consideramos este caso tan extremo de falta de disponibilidad, por tanto, el plan de contingencia que se ha seguido es mantener la planificación inicial, con la diferencia que hubo un mes con muy poco avance y por tanto repercutió en la entrega final del proyecto.

A continuación, en las siguientes tablas 47, 48, 49, 50, 51, 52, 53, 54, 55: podremos ver las tareas que se han ido realizando por cada iteración.

Iteración 1 (30/01/17 – 12/02/17)
Planificación
Estructura del Documento
Descripción Web

Tabla 47. Iteración 1

Iteración 2 (13/02/17 – 28/02/2017)
Análisis de la aplicación y definición del plan de Pruebas
Smoke Test (Plan de Pruebas Acotado)
Elaboración de los Casos de Prueba

Tabla 48. Iteración 2

Iteración 3 (01/03/17 – 16/03/17)
Análisis de cada herramienta de automatización

Tabla 49. Iteración 3

Iteración 4 (13/04/17 – 28/04/82017)
Elaboración de los Casos de Prueba Selenium
Elaboración de los Casos de Prueba Watir

Tabla 50. Iteración 4

Iteración 5 (29/04/17 – 14/05/2017)
Elaboración de los Casos de Prueba Watir
Elaboración de los Casos de Prueba Capibara

Tabla 51. Iteración 5

Iteración 6 (15/05/17 – 31/05/17)
Elaboración de los Casos de Prueba Capibara
Elaboración de los Casos de Prueba Cucumber
Ejecución del plan de pruebas en cada una de las herramientas

Tabla 52. Iteración 6

Iteración 7 (01/06/17 – 14/06/17)
Obtención de los resultados
Comparativa de Herramientas
Conclusiones

Tabla 53. Iteración 7

Iteración 8 (17/06/17 – 28/06/17)
Trabajo Futuro
Terminar de elaborar la memoria, maquetación

Tabla 54. Iteración 8

Iteración 9 (28/06/17 – 03/07/17)
Revisión de Documentación
Preparación de la Presentación de Defensa

Tabla 55. Iteración 9

7.2. Recursos del proyecto

En esta sección detallaremos los recursos que necesitaremos para realizar a lo largo de todo el proyecto, tanto los recursos hardware como los recursos software, los recursos hardware se definieron en el Capítulo II, en concreto, en la Tabla I cuando se hace mención de los roles y responsabilidades del proyecto.

En las Tablas 56, 57, 58, se muestran los recursos hardware y software utilizados en el proyecto.

Ordenador de Desarrollo (HP Pavilion Sleekbook)	
CPU	Intel®Core™ I3-3227U CPU @1.90 GHz
GPU	Intel HD Graphics 4000
RAM	6.00 GB
HDD	500 GB
OS	Windows 10 Home (x64)
DISPLAY	15.6” (1366 x 768 píxeles)

Tabla 56. Recursos Hardware (I)

Máquina virtual UVA	
CPU	Common KVM processor 2394.268 MHz
RAM	2.00 GB
HDD	6 GB
OS	Ubuntu 16.04.2 LTS (x64)

Tabla 57. Recursos Hardware (II)

Recursos Software.
Microsoft Office 2016 (Office 365)
Astah Professional
Selenium
Watir
Capibara
Cucumber
Drivers Navegadores

Tabla 58. Recursos Software

7.3. Coste del Proyecto

A continuación, se muestran las Tablas 59, 60, 61, que muestran el coste estimado que va a tener el desarrollo de este proyecto, se incluye el coste tanto hardware como software y el coste de horas de trabajo en los diferentes roles, al ser realizado por la misma persona, se ha tenido en cuenta el número de horas empleadas por el coste establecido por hora de cada rol.

Coste Hardware/Software	
Recursos Hardware (Ordenador Desarrollo)	400 €
Recursos Hardware (Máquina Virtual UVA)	0€ (*)
Recursos Software	0€ (*)
Total	**400 €**

Tabla 59. Estimación Coste Hardware/Software.

Coste Recursos Humanos		
Rol	**Horas**	**Total**
Scrum Master	10	350 €
Product Owner	35	700 €
Equipo desarrollo	320	4800 €
	Total	**5850 €**

Tabla 60. Estimación Coste Recursos Humanos.

Coste Recursos Humanos	
Coste Hardware	400 €
Coste Software	0 €
Coste Recursos Humanos	5850 €
Total	**6250 €**

Tabla 61. Estimación Coste Total

En el coste de software figura a 0€ y con *, ya que, gracias a un convenio con la Universidad de Valladolid, por el hecho de ser estudiante la licencia de ese software es gratuita.

Los costes de cada hora de cada uno de los roles indicados se han obtenido de calculando el coste por cada hora a partir del sueldo anual [8] [15] [19].

Capítulo VIII: Conclusiones y Trabajo Futuro

8.1. Conclusiones

La realización de este trabajo fin de grado aporta una ampliación de conocimiento en el entorno de las pruebas, además ha resultado muy instructivo y será útil para los desarrollos en entornos web, realizado en Everis, en mi día a día.

En la vida laboral la prueba quizá no se invierta el tiempo que se necesita, bien sea por plazos ajustados de entregas, funcionalidad que creemos que es sencilla y está funcionando…

Tenemos que tener en cuenta que las realizaciones de las pruebas es un paso esencial e importante en los desarrollos, al fin y al cabo, es el producto final que vamos a entregar a cliente y tiene que funcionar tal y como se pactó y éste especificó.

Por otro lado, como hemos ido viendo a lo largo de este documento, el mundo de las pruebas es muy grande, empezando por los difrerentes tipos de pruebas que hay, la realización de los planes de prueba en el lenguaje en el que vayamos a utilizar para desarrollarlo y la automatización de las mismas.

La automatización de pruebas, hemos hecho hincapié a lo largo del documento, con la idea de intentar transmitir sus ventajas [5]:

- Reduce el error humano: la automatización elimina el error humano, partiendo de la premisa de que está correctamente desarrollada y configurada, una prueba automatizada llevará a cabo la prueba de la misma manera cada vez que se ejecute. Por tanto, no habrá pasos que sean pasados por alto o mal interpretados.

- Reduce la cantidad de tiempo que tarda en completar una serie de pruebas. Con la automatización a través de herramientas, las pruebas se pueden ejecutar mucho más rápido que un probador manual.

- Flexibilidad de Horario: las pruebas automatizadas se pueden ejecutar en cualquier momento una gran cantidad de pruebas puede ejecutarse durante la noche o los fines de semana o en cualquier momento cuando un probador manual no está disponible.

- Mejor calidad del producto final: la automatización permite un uso más eficaz de los medios de ensayo El tiempo necesario para ejecutar las pruebas se reduce. La calidad del proceso de pruebas de software y es más fiable y se puede repetir fácilmente. Esto libera a los probadores para concentrarse en encontrar los errores más complejos a través de pruebas exploratorias ad hoc.

- Reutilización: las pruebas automatizadas pueden volver a ser utilizadas, los guiones de prueba pueden escribirse para ser agnóstico del dispositivo o plataforma en la que se están ejecutando, por lo que dichos scripts se pueden reutilizar.

  También la interfaz de usuario puede cambiar, pero la secuencia de comandos debe soportar si el comportamiento subyacente sigue siendo el mismo.

- Facilita la Integración Continua: Una suite de pruebas automatizadas se puede incluir en integración continua del proyecto.

  Por ejemplo, un conjunto de pruebas de regresión puede ser realizado todas las noches, lo que permite al equipo a comprobar fácilmente la estabilidad de la construcción al día siguiente.

Por otro lado, la utilización de diferentes herramientas, garantiza un aprendizaje muy útil, ya que como se explicó en el capítulo VI, se puede realizar una buena comparativa entre todas las herramientas y decidir así, cual es la mejor herramienta en base a las posibilidades o especificaciones iniciales.

Otros de los puntos a destacar es la importancia del estudio a la hora de comprobar si la automatización nos es útil, es importe valorar, que el aprendizaje de una herramienta y el desarrollo de un plan de pruebas que pruebe el núcleo de funcionalidad de nuestra aplicación es una tarea que no es sencilla, por lo que en muchos casos habrá que valorar el tiempo y dinero que tenemos que invertir para automatizar estos casos.

Es bastante interesante la comparativa que hemos realizado sobre cada una de las herramientas de automatización, ya que hemos visto que a pasar de que todas están destinadas a automatizar pruebas, todas tienen una o varias características en las que destacan, este es punto a tener en cuenta para diferenciarlas y, por tanto, para elegirlas.

También cabe destacar el estudio con la gráfica comparativa de tiempos, en la que se puede apreciar que la automatización de pruebas es mucho más rápida que la manual ya que en el ejemplo, en menos de un año se conseguía amortizar el tiempo invertido en el desarrollo de las pruebas.

Visto todo esto, un punto muy importante que podemos tratar, son los usos que se puede dar a estas herramientas y, en definitiva, a todo el trabajo que hay por detrás para poder automatizar las pruebas. A parte de todos los que se han indicado, como resumen, podría ser la elaboración de informes en los que se recojan los resultados de las pruebas, ya que en definitiva son lo que nos interesa, ya que es un indicador de que a pesar de las modificaciones que se han hecho, el núcleo de la aplicación funciona sin ningún tipo de problema.

Un uso importante, sería la monitorización de la ejecución de este plan de pruebas cada determinado tiempo, de esta forma se puede extraer mucha información de cómo está funcionando la aplicación. Este tema es muy importante, sobre todo, al inicio o en la puesta en marcha de una aplicación, ya que podemos detectar caídas, timeouts, etc y, en definitiva, anomalías que impidan a nuestra aplicación a funcionar correctamente.

También se ha podido comprobar de primera mano, que el tiempo de dedicación que se invertía en las primeras herramientas, se reducía considerablemente en las siguientes, a pesar de que se tratase de otra herramienta y otro lenguaje, se podía apreciar la experiencia adquirida en las herramientas anteriores, lo cual, es un gran punto a favor, a la hora de tomar la decisión de automatizar las pruebas.

Por último, comentar que lo más complicado ha sido sacar el tiempo de dedicación después de la jornada laboral, y que es un trabajo que me ha resultado muy gratificante e interesante. Después de todo, sé que me resultará muy útil como formación personal y profesional, pudiendo proponerlo en el actual proyecto como posible mejora.

8.2. Trabajo futuro

Este trabajo fin de grado deja la puerta abierta a un gran número de posibilidades para poder trabajar en el futuro ampliando los objetivos marcados en este proyecto.

En el entorno que hemos montado, tenemos una página web alojada en un servidor, de forma que desde el punto de vista de la automatización de pruebas se pueden realizar varios proyectos de estudio, también está alojado en dos servidores proporcionados por la Universidad de Valladolid. Uno para el contenido de la página en cuestión y el otro para alojar toda la base de datos de la aplicación.

En ambos casos, sería interesante construir nuevamente esta aplicación, para tenerla en unos servidores que no se borren, ya que como pertenecen a la asignatura de Planificación y Gestión de Plataformas Informáticas se procederá en julio al borrado de las mismas. Por tanto, para que no muera el proyecto en ese borrado sería interesante realizar un migrado de ambas máquinas virtuales a otras en las que podamos gestionar nosotros mismos la aplicación.

El entorno que tiene preparado está página web es muy interesante para el ámbito educativo, ya que es un entorno en el que están perfectamente establecidos unos límites y que por tanto puede ser el objetivo, de muchas asignaturas para la realización de prácticas a modo de pruebas concepto.

En el mundo del diseño y realización de pruebas, sabemos que se pueden realizar tantas pruebas como se desee ya que el número de estas tiende a infinito. Seguido a esta idea, podemos pensar en realizar más pruebas de funcionalidad en esta aplicación web y otros tipos de pruebas, ya que a pesar de ser un entorno sencillo se asemeja muy bien a un entorno real, ya que presenta una arquitectura cliente servidor de tres niveles perfectamente definida.

Desde otro punto de vista, las herramientas que se pueden utilizar para automatizar cada uno de los tipos de prueba son un gran número de ellas, por lo que podemos probar nuevas herramientas para cualquier tipo de pruebas y con las diferentes características que elijamos.

Por ejemplo, en este proyecto se ha realizado una comparativa de herramientas opensource, podía haberse realizado con alguna herramienta cuya licencia tiene un coste, pero si que tiene licencia gratuita (con funcionalidad reducida) durante un período de tiempo determinado. En este caso, por ejemplo, puede ser muy interesante ver las diferencias que hay entre una herramienta opensource a una de pago o viceversa, ya que como hemos visto, aunque haya varias herramientas que estén destinadas para un mismo fin va a haber diferencias, que será importante tenerlas en cuenta para cuando necesitemos, realmente, usarlas.

Otro detalle que se podría mejorar, es a la hora de realizar los informes con los resultados de la ejecución del plan de pruebas, se podría generar un Excel con los resultados, indicando el resultado de cada una de las pruebas parciales, incluso realizar el desarrollo de enviar una notificación de correo electrónico con los resultados.

En definitiva, completar más los planes de pruebas realizados para obtener informes que se generen automáticamente para poder obtener los resultados de una manera formal y rápida, ya que esta parte además complementaría y formalizaría todo este proyecto.

Bibliografía

[1] Blanco Bueno, Carlos. Ingeniería del Software II. Universidad de Cantabria[Online].
http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1-pruebasSistemasSoftware.pdf
Fecha de última consulta: 14/05/2017

[2] Cohn, Mike. Succeeding with Agile. Software Development Using Scrum.
Addison-Wesley Signature Series. ISBN-13: 978-0137043293.
Fecha de última consulta: 18/06/2017.

[3] Fuente, Redondo, Pablo de la. Gestión de Riesgos. Asignatura Planificación y Gestión de Plataformas Informáticas. [Online].
https://aulas.inf.uva.es/pluginfile.php/26796/mod_resource/content/0/materiales_planif/resumen_riesgos.pdf
Fecha de última consulta: 14/05/2017

[4] Fuente, Redondo, Pablo de la. Presentación de SCRUM. Asignatura Planificación y Gestión de Plataformas Informáticas. [Online].
https://aulas.inf.uva.es/pluginfile.php/26842/mod_resource/content/6/PGPI_SCRUM_2_2016_2017.pdf
Fecha de última consulta: 27/04/2017

[5] GameSparks. Automated Testing with Cucumber and Capybara[Online]
https://www.gamesparks.com/blog/automated-testing-with-cucumber-and-capybara/
Fecha de última consulta: 30/06/2017

[6] Ian Sommerville. "Ingeniería del software".
7ª edición, publicado por PEARSON EDUCACION, 2005. ISBN: 84-7829-074-5.
Fecha de última consulta: 14/05/2017

[7]. IEEE. Guide to the Software Engineering Body of Knowledge SWEBOK, 2004 version Capítulo 5 [Online]
http://www.cc.uah.es/drg/b/HispaSWEBOK.Borrador.pdf
Fecha de la última consulta: 13/05/2017.

[8] Infojobs. Product Owner [Online]
https://www.infojobs.net/ofertas-trabajo/product-owner
Fecha de última consulta: 15/06/2017

[9] ISO/IEC/IEEE 29119 Software Testing [Online]
http://www.softwaretestingstandard.org/part4.php
Fecha de última consulta: 13/05/2017

[10] JUnit [Online]
http://junit.org/junit4/
Fecha de última consulta: 22/06/2017

[11] Martin, Fowler Test Pyramid. [Online]
https://martinfowler.com/bliki/TestPyramid.html
Fecha de última consulta: 28/05/2017

[12] Microsoft. Modo protegido mejorado en IE para el escritorio [Online]
https://msdn.microsoft.com/es-es/library/dn265025(v=vs.85).aspx
Fecha de última consulta: 01/07/2017

[13] Morales, María. Gherkin y 10 buenas prácticas a tener en cuenta a la hora de escribir Features (Parte 1) [Online]
http://www.javiergarzas.com/2016/05/gherkin-10-buenas-practicas-cuenta-la-hora-escribir-features-parte-1.html
Fecha de última consulta: 23/06/2017

[14] Neverov, Iván. Sphere Software. Achieve Quality Code and ROI throught Test[Online]
https://sphereinc.com/achieve-quality-code-and-roi-through-test-automation/
Fecha de última consulta: 28/05/2017

[15] North, Dan & Associates. organizaciones más rápidas, el software más rápido [Online]
https://dannorth.net/
Fecha de última consulta: 23/06/2017

[16] Palacio, Juan. Gestión de proyectos Scrum Manager v. 2.5 [Online].
http://www.scrummanager.net/files/sm_proyecto.pdf
Fecha de última consulta: 27/04/2017.

[17] Palacios, Jerónimo. Scrum Master es uno de los trabajos más prometedores del 2017 según LinkedIn. [Online]
https://jeronimopalacios.com/2017/02/scrum-master-uno-los-trabajos-mas-prometedores-del-2017-segun-linkedin/
Fecha de última consulta: 15/06/2017

[18] Pete Deemer, Gabrielle Benefield, Craig Larman y Bas Vodde. Una introducción básica a la teoría y práctica de Scrum. [Online].
http://scrumprimer.org/primers/es_scrumprimer20.pdf
Fecha de última consulta: 27/04/2017.

[19] PMOinformatica.com. La oficina de proyectos en informática. Cucumber [Online]
http://www.pmoinformatica.com/2014/09/5-preguntas-y-respuestas-sobre-cucumber.html
Fecha de última consulta: 23/06/2017

[20] Project Management Institute, PMBOK, Project Management Body of Knowledge, 2000. Publicado por Project Management Institute, Four Campus Boulevard, Newtown Square, Inc. ISBN: 1-88410-22-2
Fecha de última consulta: 14/05/2017.

[21] Qt. Qt Documentación [Online]
http://doc.qt.io/qt-4.8/qtwebkit-guide.html
Fecha de última consulta: 24/06/2017

[22] Scott, Alister. Introducing the software testing ice-cream cone (anti-pattern) [Online]
https://watirmelon.blog/2012/01/31/introducing-the-software-testing-ice-cream-cone/
Fecha de última consulta: 03/06/2017.

[23] Selenium Grid Configuration [Hub and Node Communication] [Online]
http://www.serveridol.com/2013/05/04/selenium-grid-hub-and-node-communication/
Fecha de última consulta: 05/06/2017.

[24] SeleniumHQ. Browser Automation. Selenium IDE[Online]
http://www.seleniumhq.org/projects/ide/
Fecha de última consulta: 02/06/2017.

[25] Sitepoint. The Basics of Capybara and Improving Your Tests [Online].
https://www.sitepoint.com/basics-capybara-improving-tests/
Fecha de última consulta: 02/07/2017.

[26] VASS digital. SCRUM, desarrollo ágil por excelencia[Online]
http://www.vassdigital.com/scrum-la-metodologia-de-desarrollo-agil-por-excelencia/
Fecha de última consulta: 26/04/2017.

[27] w3ii.com. Los últimos tutoriales de desarrollo web. Selenium WebDriver [Online]
http://www.w3ii.com/es/selenium/selenium_webdriver.html
Fecha de última consulta: 25/06/2017.

[28] w3schools.com. JavaScript HTML DOM [Online]
https://www.w3schools.com/js/js_htmldom.asp
Fecha de última consulta: 21/06/2017

[29] Wikidot. Ruby Tutorial ...o como pasar un buen rato programando. Unit Testing
http://rubytutorial.wikidot.com/unit-testing
Fecha de última consulta: 16/05/2017

[30] Wowro, Michael. It Kosmopolit – Testautomatisierung [Online]
http://it-kosmopolit.de/blog/2012/10/10/selenium-webdriver-mindmap/
Fecha de última consulta: 17/06/2017

ANEXO I

I. Manual de Usuario

A pesar de todo el análisis que hemos realizado, será necesario el manual de usuario de la aplicación, puesto que en las pruebas de aceptación es la funcionalidad que seguirán cada uno de los diferentes tipos de roles de usuarios que usarán la misma.

I.I. Acceso General a la aplicación.

En este apartado, detallaremos las interfaces con las que cualquier tipo de usuario va a interactuar independientemente del rol que tenga.

Una de ellas es la pantalla de acceso al sistema, donde el usuario tendrá que introducir su username y su contraseña para poder acceder al sistema, tal y como podemos ver en la Figura 30:

Figura 30. Página inicio aplicación

Una vez que se han introducido los datos y se pulse el botón iniciar sesión y el sistema validará si los datos introducidos son correctos permitiendo el acceso al sistema, en caso afirmativo visualizaremos una pantalla como la que nos muestra la Figura 31, más adelante veremos que cada uno de los diferentes roles de la aplicación, tiene una página inicial diferente.

En caso de que no sea correcta la contraseña, si hacemos click en ¿Olvidaste la contraseña? Nos mostrará la pantalla que se aprecia en la Figura 32. En esta pantalla, hay que introducir los datos oportunos para que podamos restablecer nuestra cuenta, ya que se informará al administrador para que pueda restablecernos la contraseña y poder logarnos correctamente.

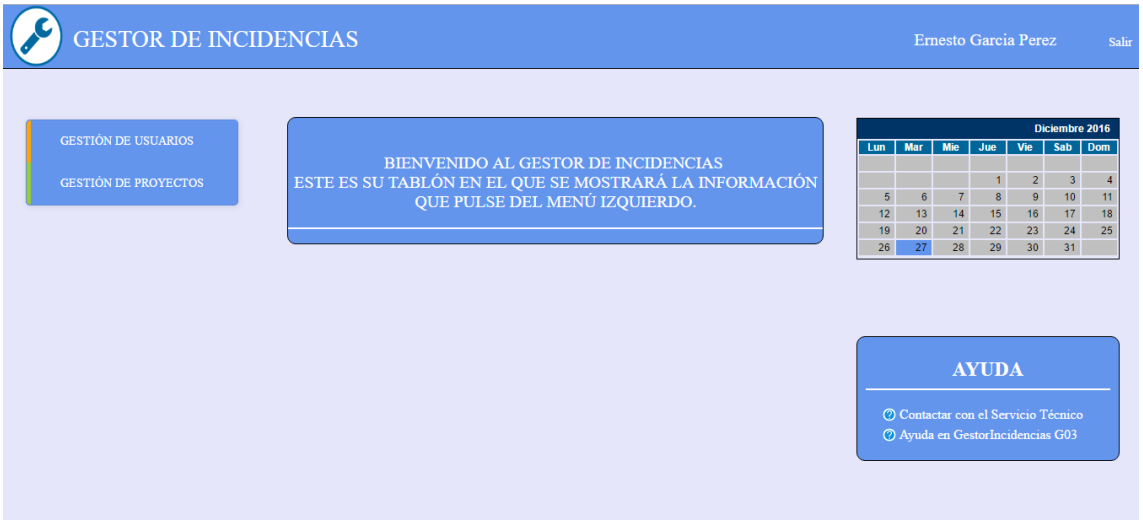

Figura 31. Página Principal.

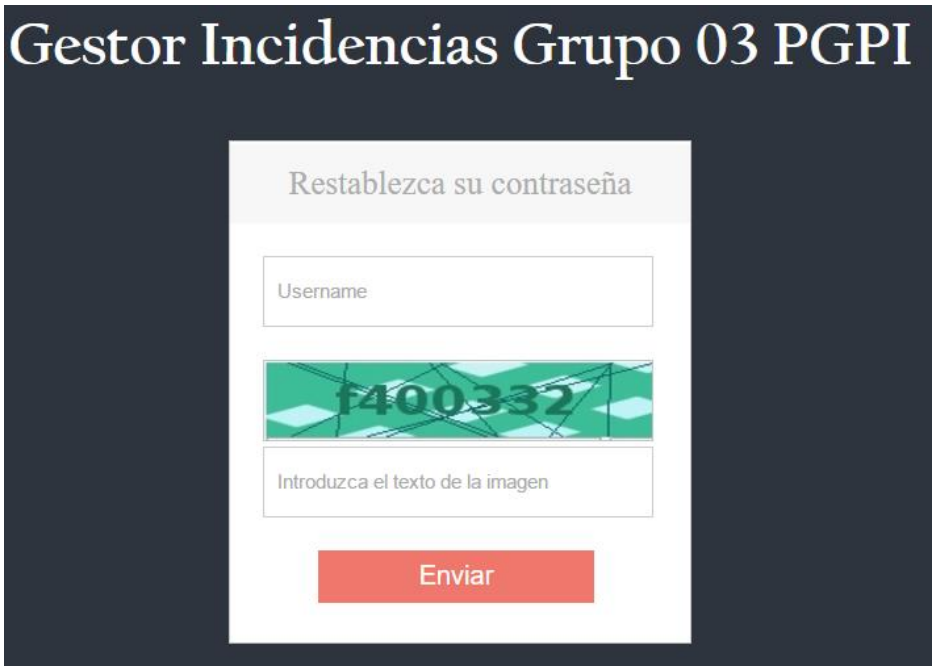

Figura 32. Página Restablecer Contraseña.

A continuación, en el siguiente punto, detallaremos el manual de usuario relacionado con cada uno de los diferentes roles.

I.II. Manual de Usuario Administrador.

Este usuario, se encargará de dar de alta proyectos. Cargará los datos iniciales, como tablas de categorías, etc. En realidad, este usuario pertenece al sistema general, no tanto que, a la aplicación, aunque es el encargado de realizar las funciones indicadas para la aplicación. También se encargará de dar de alta las cuentas de los usuarios, distinguiendo entre clientes, técnicos y supervisor.

Una vez que se ha detallado toda la funcionalidad que tiene este rol en concreto, vamos a mostrar cómo llevar a cabo toda esta funcionalidad.

Como hemos indicado anteriormente, partimos de que se registró un usuario correctamente, en este caso, con el login y la password del administrador correctos.

Si seleccionamos en la pestaña del menú lateral “GESTION DE USUARIOS”. Podremos ver la pantalla que muestra la Figura 33. Como se puede apreciar, tenemos una tabla con todos los usuarios del sistema donde podremos modificarlos o eliminarlos. También disponemos de la funcionalidad de registrar un nuevo usuario si seleccionamos en el botón “Registrar usuario”, justo encima de la tabla que contiene a los usuarios.

Figura 33. Página Home Administrador

A continuación, iremos detallando cada una de estas tres funcionalidades de las que dispone la pantalla que aparece en la Figura 33. Vamos a comenzar por la opción de registrar usuario, si seleccionamos este botón podremos ver la pantalla de la Figura 34.

Como vemos es un formulario donde tiene todos los datos a introducir que son necesarios en la aplicación, todos ellos obligatorios, estos son: el nombre, apellidos, login, password, y el tipo de rol que va a desempeñar dicho usuario en la aplicación. Una vez que hayamos completado todos y cada uno de los campos daremos al botón “Guardar” para dar de alta a nuestro nuevo usuario.

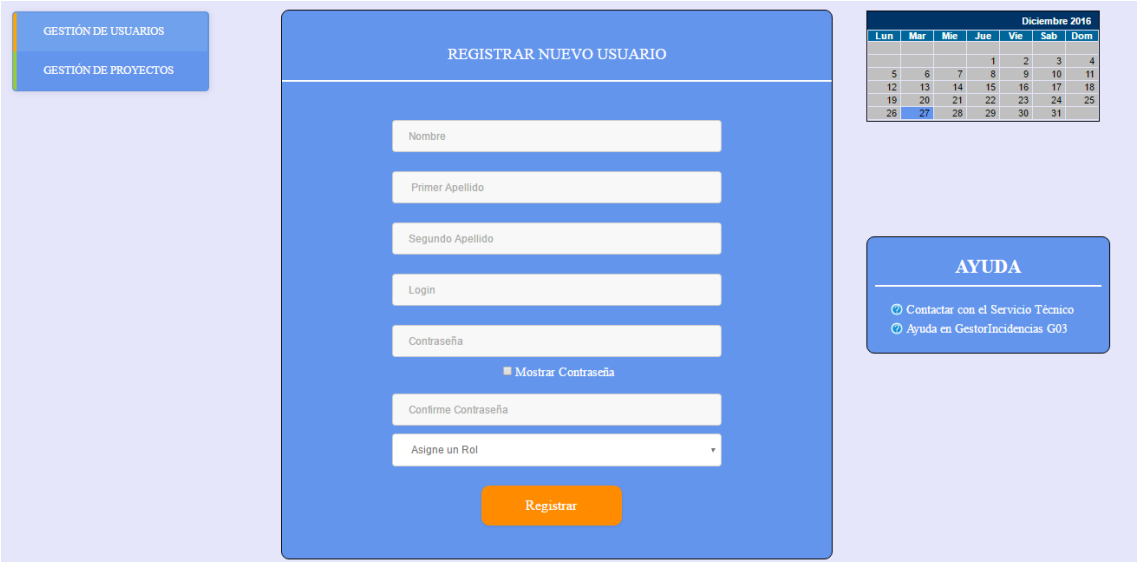

Figura 34. Página Registrar Usuario.

Una vez que hayamos guardado a nuestro nuevo usuario volveremos a aparecer en la página de la Figura 33, donde podremos observar el nuevo usuario que se ha dado de alta.

Tal y como mencionamos antes, a cada uno de los usuarios podemos modificarlos o eliminarlos, si queremos modificarlos haremos click en la opción modificar, en este caso veremos la pantalla de la Figura 35.

En esta pantalla podemos ver que tenemos un formulario, tal y como rellenamos en "Registrar Usuario" pero en el que los campos del formulario aparecen con los datos que introdujimos en el momento de darlo de alta, para modificar cualquiera de los campos y por tanto la información de este usuario haremos click en dicho campo e introduciremos la información que queramos actualizar. Cuando lo hayamos realizado, podremos dar al botón "Guardar" para actualizar dicha información

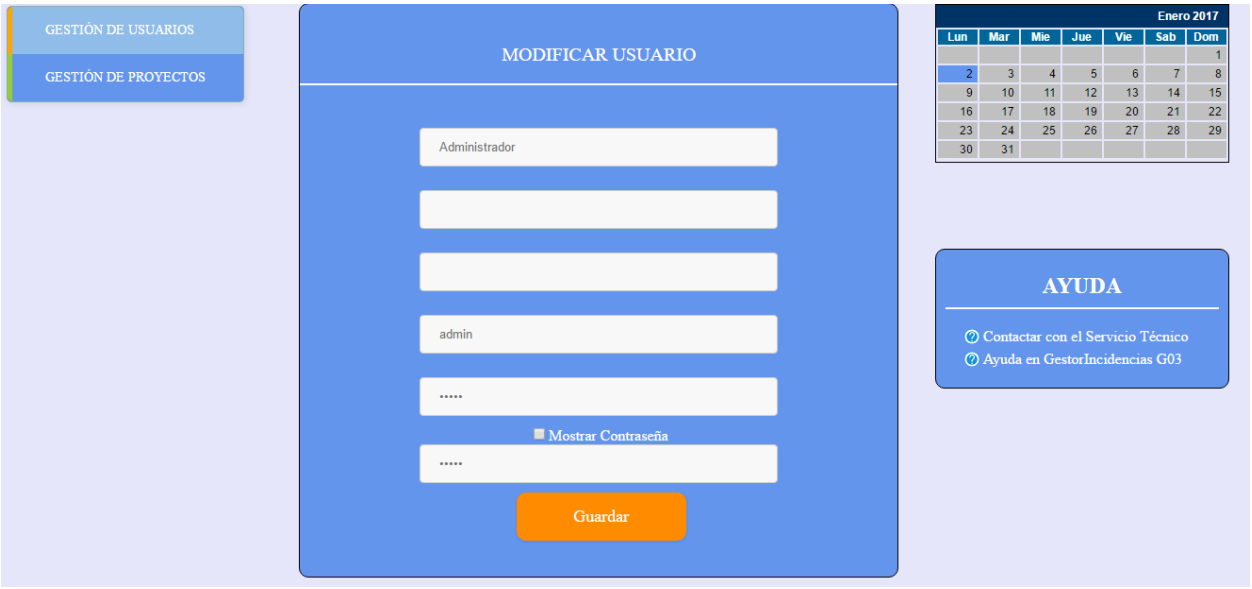

Figura 35. Página Modificar Usuario.

Cuando hayamos modificado nuestro usuario, volveremos a la página de la Figura 33, donde podremos ver los nuevos cambios que hayamos hecho a un usuario en concreto en cada una de las diferentes columnas de la tabla.

Por último, como comentamos, también podremos eliminar un usuario, para ello haremos click sobre la opción eliminar y tendremos que aceptar la siguiente ventana de confirmación que aparece en la Figura 36, para poder eliminarlo.

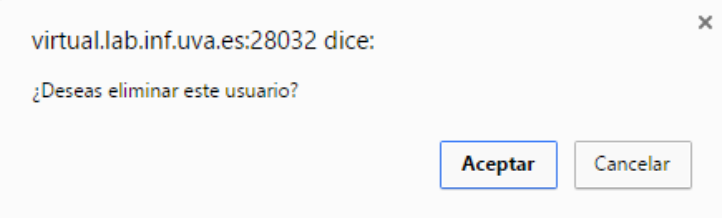

Figura 36. Confirmación Eliminación Usuario.

Una vez realizada esta acción, también volveremos a la página de la Figura 33, y una vez más podremos comprobar, que hemos eliminado dicho usuario con éxito.

Como se detalló al principio del punto, podremos gestionar proyectos. Para ello, seleccionaremos en el menú lateral la opción de "GESTION PROYECTOS", apreciando la pantalla de la Figura 37.

Como vemos en la Figura 37, presenta una funcionalidad de gestión de usuarios, también tenemos una tabla que nos muestra cada uno de los proyectos que están registrados en el sistema. Podremos realizar dos funcionalidades desde esta pantalla: dar de alta nuevos proyectos, y eliminarlos.

Figura 37. Página Gestión Proyectos.

Par dar de alta un nuevo proyecto, tendremos que seleccionar en el botón "Añadir Proyecto" viendo la página de la Figura 38.

Figura 38. Página Nuevo Proyecto.

Como vemos, tenemos un formulario en el que solo hay que añadir la descripción de nuestro nuevo proyecto, una vez que la hayamos detallado daremos al botón “Guardar” y nos redirigirá a la página de gestión de proyectos, donde podremos ver nuestro nuevo proyecto.

Si por el contrario queremos eliminar, tendremos que hacer click sobre eliminar, aceptar la confirmación de la eliminación de éste, similar a la Figura 36 y nuevamente nos redirigirá a la página de la Figura 35, en la que podremos comprobar que este se eliminó con éxito, el proyecto deseado.

I.III. Manual de Usuario Cliente.

Este usuario, podrá introducir los datos correspondientes a las incidencias que quiere comunicar y podrá consultar el estado de todas las incidencias que ha introducido mientras éstas estén abiertas. También podrá consultar la información correspondiente a todas las incidencias que él ha generado y que estén cerradas, identificando la causa del cierre.

Si nos logueamos correctamente con este rol, nos aparecerá la siguiente pantalla:

Como podemos observar en la Figura 39, tendremos que seleccionar en el menú izquierdo "MIS INCIDENCIAS" para poder realizar todas las acciones que se detallaron al inicio de este punto.

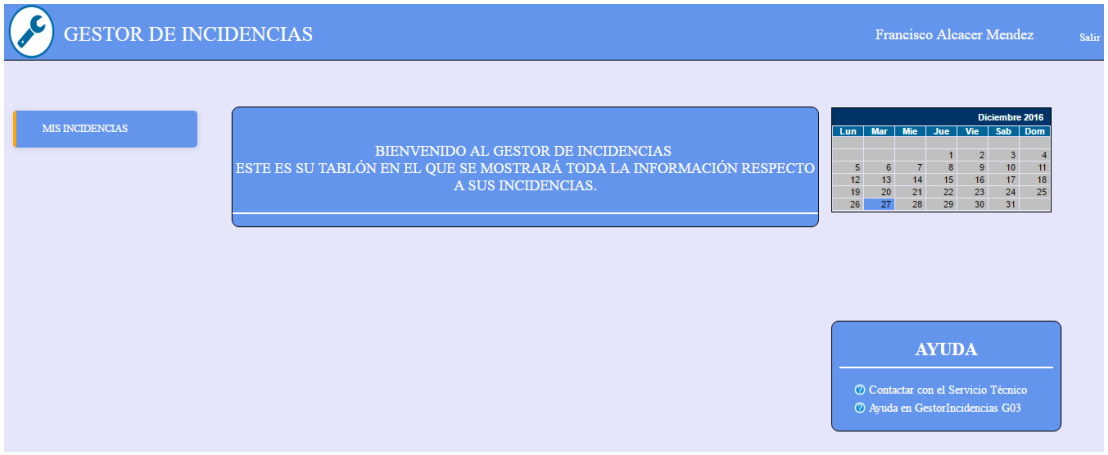

Figura 39. Página Home Cliente

Como podemos apreciar en la Figura 40, tenemos la posibilidad de generar incidencias a través del botón "Generar Incidencia", filtrarlas a través del filtro y verlas o eliminarlas desde la tabla que en la veremos todas las incidencias que generó el cliente

Figura 40. Página "Mis Incidencias"

Comenzaremos por el botón “Generar Incidencia”, dicho botón lo utilizaremos para dar de alta una nueva incidencia, la pantalla que visualizaremos es la que aparece en la Figura 41.

Como se puede observar, consta de un formulario en el que hay que rellenar los campos correspondientes a la información de la incidencia, estos son una descripción, la fecha que solo se podrá dar de alta incidencias menores o iguales a la fecha actual, si no mostrar un aviso y no nos dejara dar de alta la incidencia. También tendremos que indicar la prioridad que tiene dicha incidencia y la categoría a la que pertenece la misma: Hardware, software básico, software de aplicaciones o problemas con las comunicaciones.

Una vez contemplemos todos los campos del formulario daremos a “Registrar” y nos redirigirá a la pantalla “mis incidencias” donde podremos observar en la tabla la propia incidencia creada.

Figura 41. Pantalla Registrar Incidencia

La siguiente funcionalidad a explicar, es el filtro, que se puede apreciar en la Figura 42, podremos generar los listados las incidencias que nos interesen en cada momento, si hacemos uso del mismo, podremos obtener las abiertas o las cerradas:

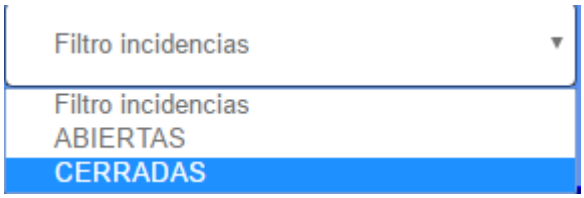

Figura 42. Filtro Incidencias

Seleccionando una de las opciones que nos proporciona el filtro, nos aparecerán en la tabla las incidencias abiertas por el cliente que tienen el estado de abiertas o cerradas, dependiendo la opción marcada.

Por último, podremos ver una incidencia o eliminarla, dependiendo donde seleccionemos en cada una de las filas que nos muestra la Figura 43.

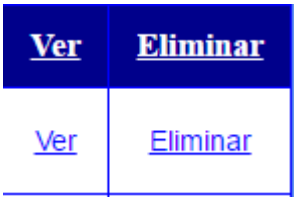

Figura 43. Opciones Incidencias.

Podremos seleccionar cualquiera de las dos opciones:

Si seleccionamos ver, aparecerá la pantalla de la Figura 44, en la que se nos muestra el título de la incidencia, el estado, la fecha de apertura, la descripción, la categoría, los elementos asignados a dicha incidencia, y el comentario realizado por el técnico que la atendió.

Como hemos comentado, todos los campos tienen información relacionada con la incidencia, pero como podemos ver, en la parte inferior tenemos dos botones, cerrar incidencia resuelta o cerrar incidencia sin solución, en este caso aparecen deshabilitados (en gris) ya que la incidencia está cerrada, si estuviese en estado abierta aparecerán de color naranja indicando que están habilitados para poder seleccionarlo, ambos botones sirven para que el propio cliente cierre la incidencia, bien sea porque se le dio solución a la misma o porque no la tuvo

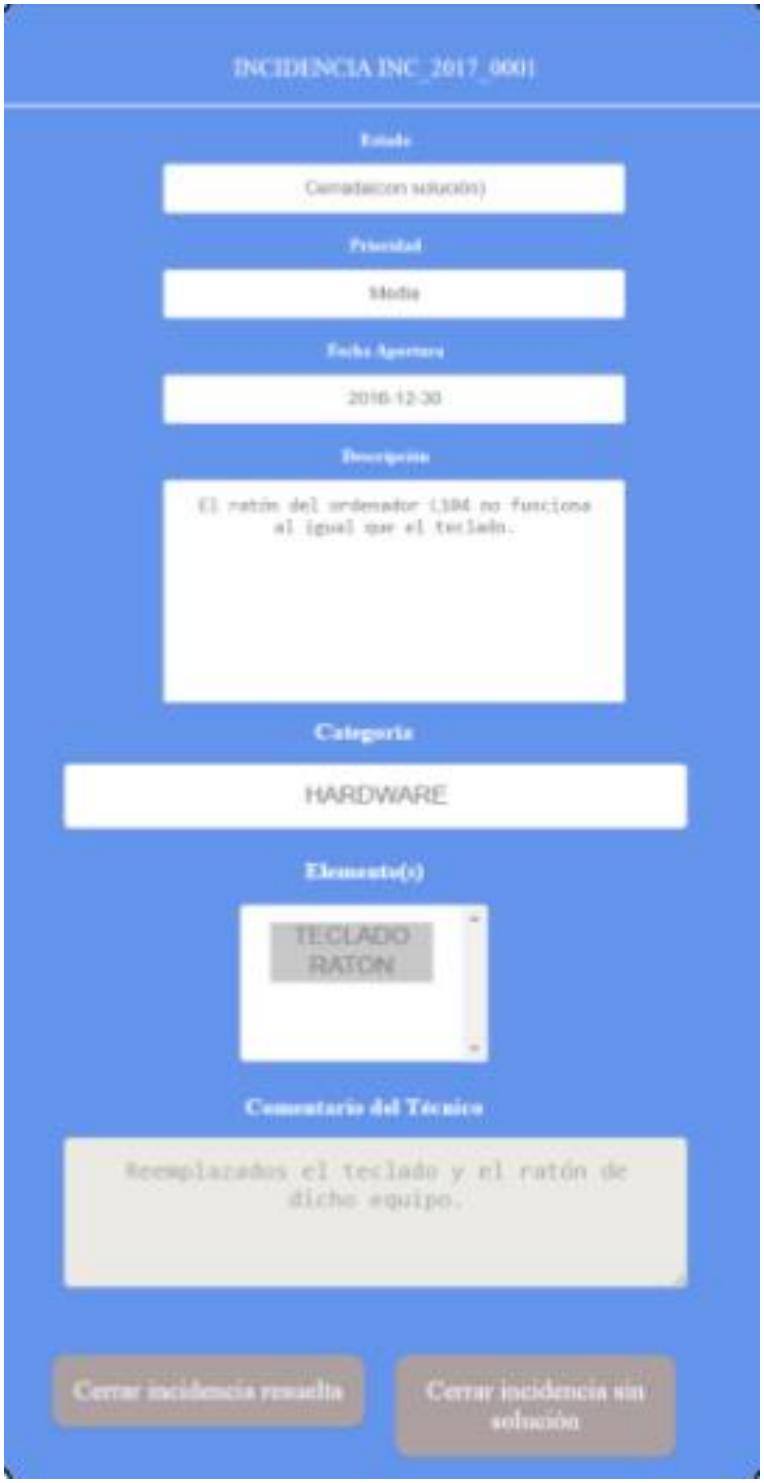

Figura 44. Pantalla Información Incidencia

Y, por último, podremos seleccionar eliminar, donde borraremos del sistema la incidencia seleccionada, deberemos aceptar una ventana de confirmación que nos aparecerá cuando pulsemos como “eliminar”

Y si seleccionamos en aceptar se eliminará o no si damos a cancelar, en cualquiera de las dos opciones volveremos a la página de “Mis incidencias”, donde podremos ver en la tabla si realmente se eliminó o no. Pantalla que podemos ver en la Figura 40.

I.IV. Manual de Usuario Supervisor.

El supervisor se encargará de analizar las solicitudes de incidencias y asignar al técnico que deberá resolverla, el elemento de inventario afectado (o los elementos afectados), junto con una indicación de prioridad a la vista de la demanda del cliente y las características del problema a tratar.

También podrá decidir el instante de cierre de una incidencia y mientras la incidencia está abierta, el supervisor puede consultar su estado y datos de los tiempos requeridos desde el momento de su solicitud por parte del cliente.
También podrá acceder a la información sobre los datos correspondientes a todas las incidencias cerradas que se han registrado en el sistema. Y podrá cerrar definitivamente una incidencia.

En la Figura 45, podemos ver la página home del Supervisor

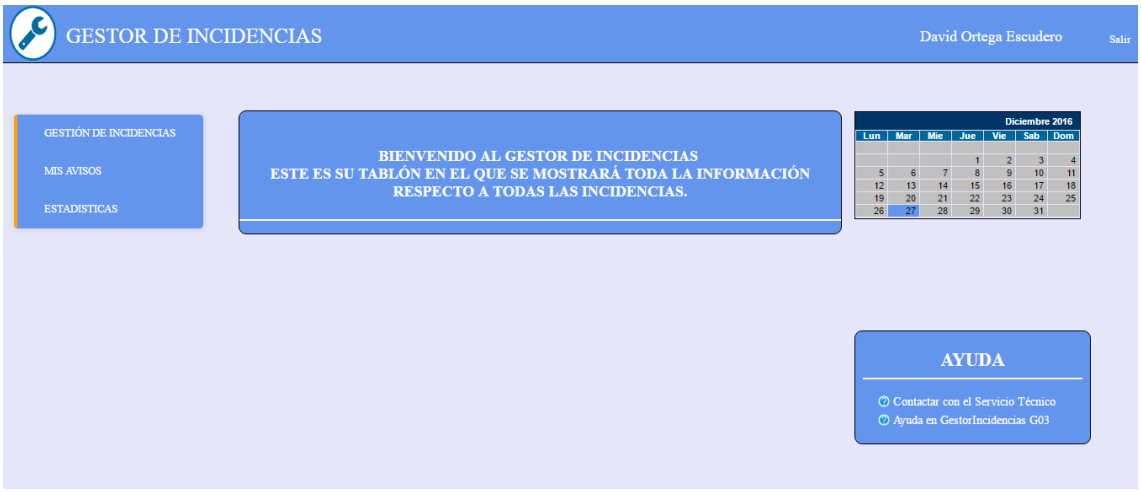

Figura 45. Página Home Supervisor.

Para poder realizar toda la funcionalidad indicada, se deberá de acceder al menú izquierdo a cada una de las tres opciones disponibles:

- Gestión de Incidencias.
- Mis avisos.
- Estadísticas.

Comenzaremos por la pantalla de nos aparecerá cuando seleccionamos a “Gestión de Incidencias”, se mostrará en la Figura 46.

Tal y como podemos ver, es un formulario similar al que nos encontramos cuando damos de alta una incidencia con el usuario cliente, solo que aquí añadiremos en el campo acción si aceptamos o rechazamos una incidencia y en el campo asignar técnico.

El técnico que deseemos que resuelva dicha incidencia, también podremos seleccionar los elementos asociadas a dicha incidencia que se nos mostrarán dependiendo del tipo de categoría a la que se haya asignado dicha incidencia, también podremos cerrar una incidencia definitivamente, este botón se nos habilitara cuando un técnico ha cerrado una incidencia, bien sea con o sin solución, si se da dicho caso, también figurara en la pantalla de “Mis Avisos”, este punto se explicará más adelante.

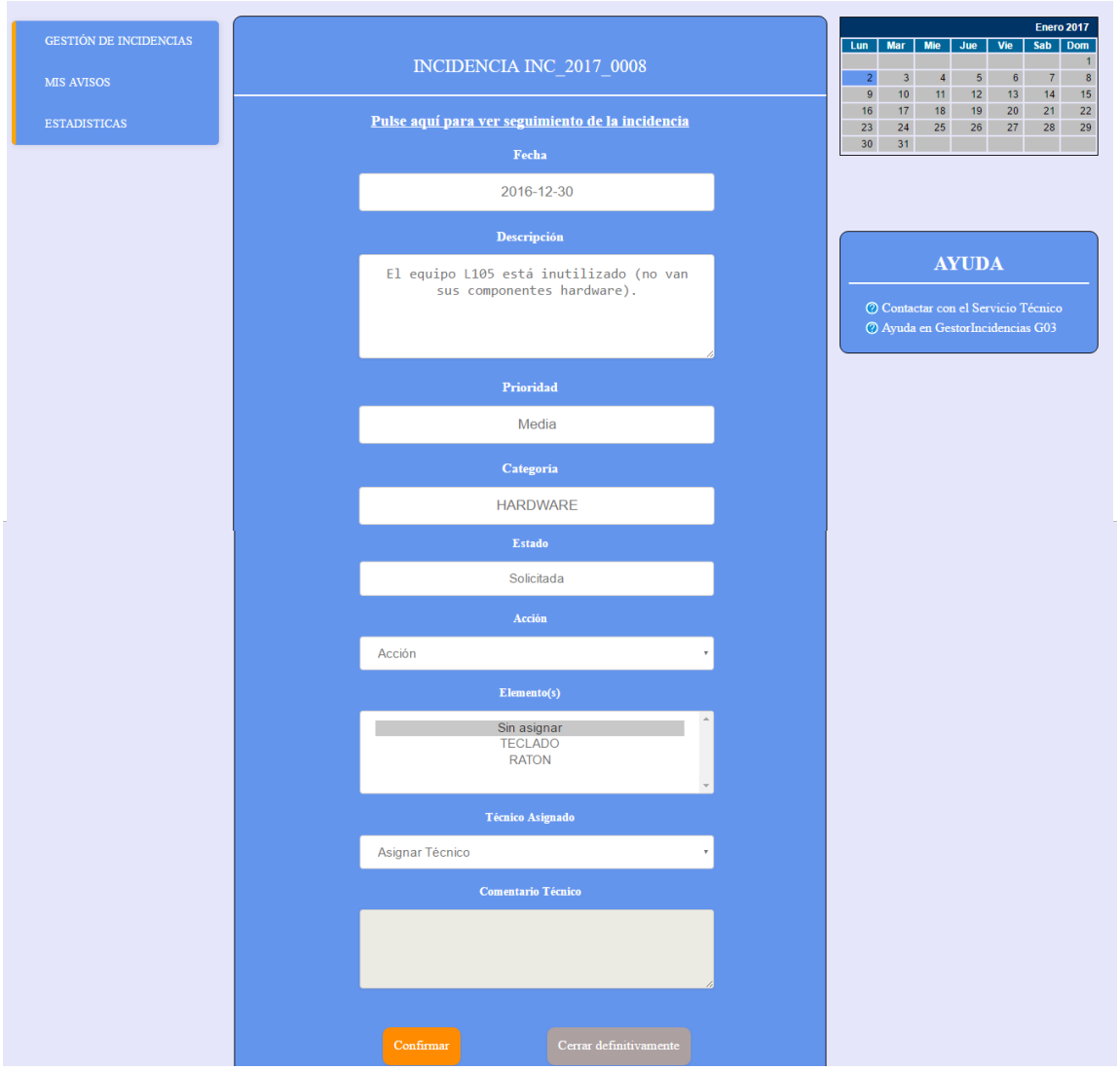

Figura 46. Página Análisis Incidencia

En la Figura 47, podremos ver el movimiento y cambio de estados de la incidencia, para ello tendremos que dar en el link que aparece en la parte superior de la pantalla de la Figura 35, con el literal de "pulse aquí para ver el movimiento de la incidencia".

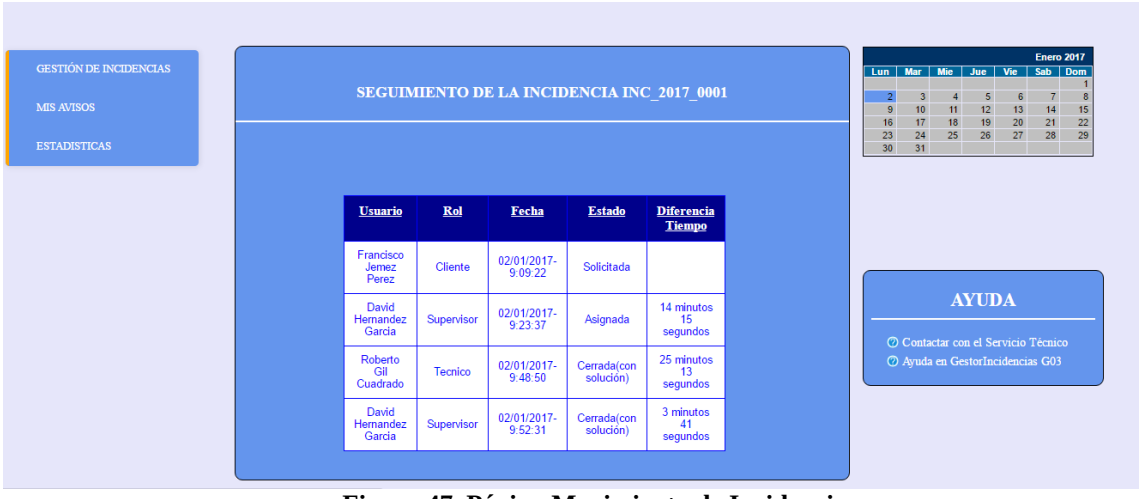

Figura 47. Página Movimiento de Incidencia

Se podrán utilizar los diferentes filtros para poder generar listados de incidencias dependiendo del estado, rol de usuario o situación en el sistema.

En la Figura 48, podemos ver el primer filtro que nos servirá para seleccionar aquellas incidencias que han interactuado con un cliente. De igual forma, si seleccionamos por cliente o por técnico, se nos habilitará el segundo filtro para ver las incidencias de un determinado cliente o técnico.

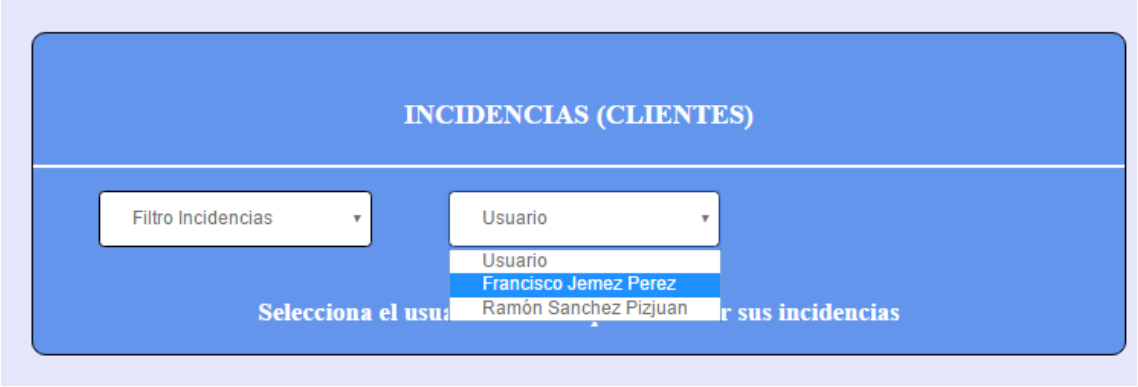

Figura 48. Filtro Incidencias Clientes/Técnicos

A continuación, seguiremos en el punto de “Mis avisos”, en dicha opción aparecerán todas las incidencias que han cerrado los técnicos, donde podremos seleccionar a “ver” y de forma análoga a dicha pantalla cuando lo realizábamos en “Gestión de Incidencias”, obtendremos un formulario en el que veremos todo detalle sobre dicha incidencia, con los comentarios oportunos realizados por el técnico, se puede apreciar en la Figura 49.

Figura 49. Pantalla Avisos de Técnicos

Si seleccionamos en ver, veremos el mismo formulario que hemos detallado antes, solo que se nos habilitará el botó de cerrar definitivamente, lo podemos ver en la pantalla que nos muestra la Figura 50.

Figura 50. Pantalla Cierre Definitivo de Incidencia

Por último, si seleccionamos en el menú principal a "Estadísticas" como se ve en la Figura 51, podremos utilizar el filtro para poder ver las estadísticas generadas por estado, en la que se nos indicará el tiempo promedio que ha estado cada incidencia en un estado determinado.

Figura 51. Pantalla Estadísticas de Incidencias

I.V. Manual de Usuario Técnico.

El técnico podrá introducir los datos sobre comentarios oportunos sobre la evolución de las incidencias que le han sido asignadas. Decidirá el momento de cierre de las mismas, ya sea un cierre con el problema resuelto o sin resolver. En cualquier momento podrá acceder a la información sobre las incidencias abiertas que él tiene asignadas y sobre todas las incidencias cerradas que estén presentes en la plataforma.

Una vez logados en el sistema con el rol de técnico veremos una página home como la de la Figura 52.

Para poder realizar toda la funcionalidad anteriormente explicada tendremos que acceder al menú izquierdo a una de las dos opciones: “Incidencias” o “Mis avisos”.

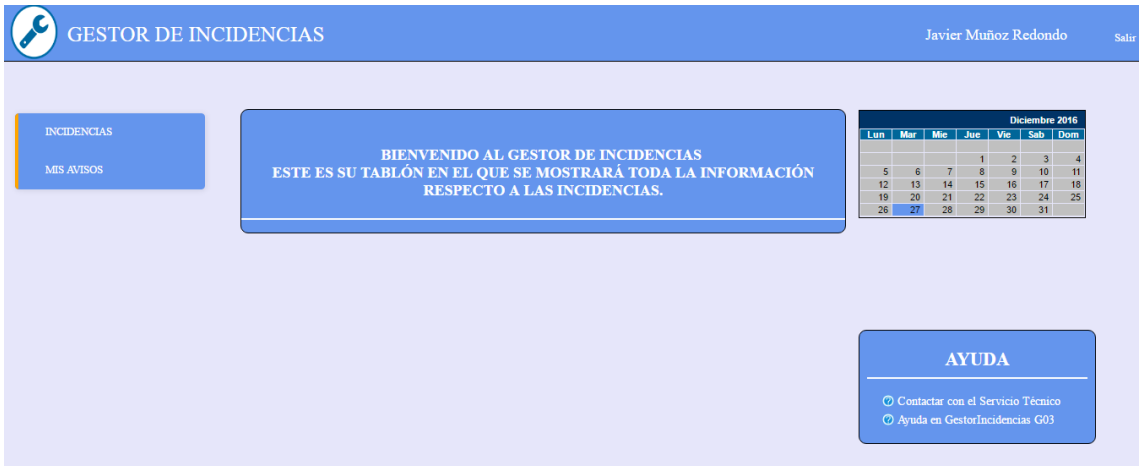

Figura 52. Página Home Técnico

Comenzaremos por “Incidencias”. Si seleccionamos esta opción nos redirigirá a la pantalla de la Figura 53, en esta pantalla veremos todas las incidencias que tiene asignadas el técnico.

Figura 53. Pantalla Incidencias Asignadas

Con el filtro que nos aparece podremos generar listados de todas aquellas incidencias que figuran con el estado abiertas, cerradas o todas las incidencias que hay registradas en el sistema, en cualquiera de las opciones del filtro.

En la Figura 54, veremos la pantalla que obtenemos si seleccionamos en ver, donde podremos añadir el comentario oportuno o solución dada por el técnico.

Una incidencia que tiene el estado “solicitada”, obtendremos el mismo formulario que teníamos en cliente y supervisor, con la diferencia de que se nos habilita, la posibilidad de escribir los comentarios que crea oportunos el técnico en cuestión y la posibilidad de cerrar o no una incidencia.

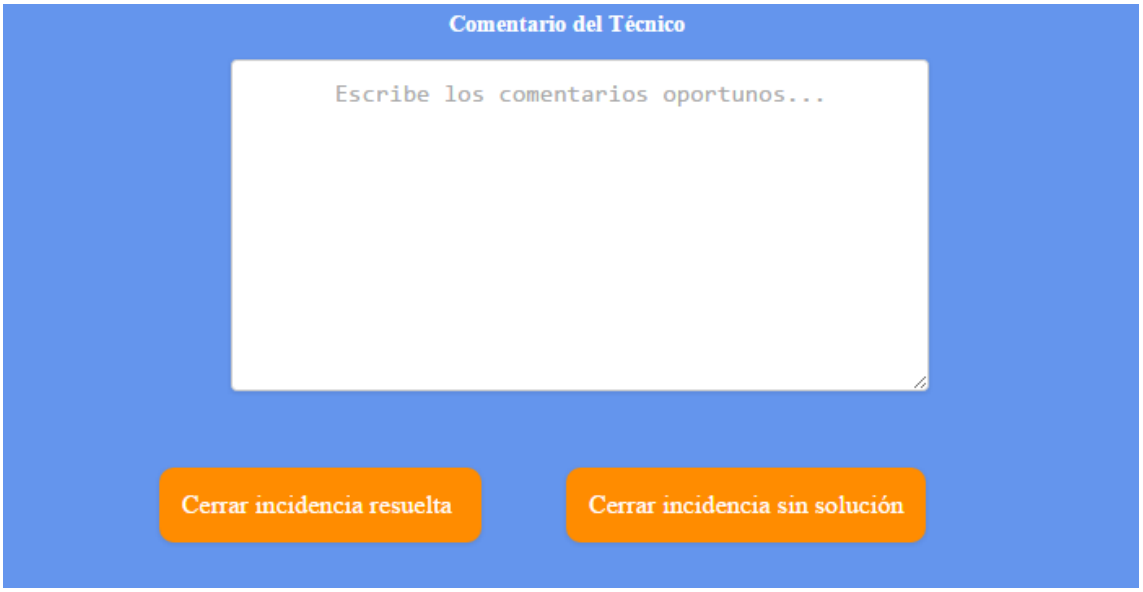

Figura 54. Pantalla Comentario Técnico

I.VI. Ayuda sobre la aplicación.

Una vez hayamos accedido a la aplicación de gestión de incidencias, con cualquiera de los usuarios, tendremos en la parte media y derecha de la pantalla la página que de la Figura 55. En dicho menú podremos acceder a dos opciones: Contactar con el servicio técnico o ayuda en el gestor.

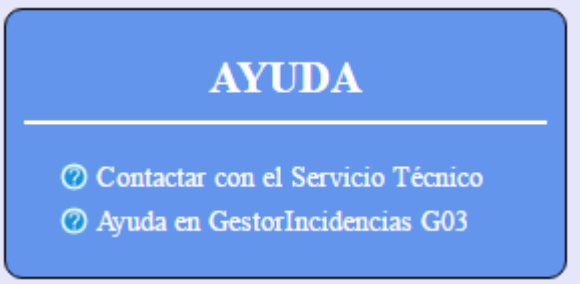

Figura 55. Menú de Ayuda

Si seleccionamos en “Contactar con el Servicio Técnico”, en la que nos proporciona toda la información necesaria para contactar con el servicio técnico de la aplicación:

Si, por el contrario, seleccionamos en “Ayuda en Gestor Incidencias G03” se mostrará la pantalla de la Figura 56, se podrá apreciar un formulario en el que tenemos que añadir un resumen y una descripción del problema con el que nos encontramos y daremos al botón enviar, posteriormente tendremos que confirmar el envío al soporte técnico para que se ponga en contacto con nosotros para resolver dicho problema.

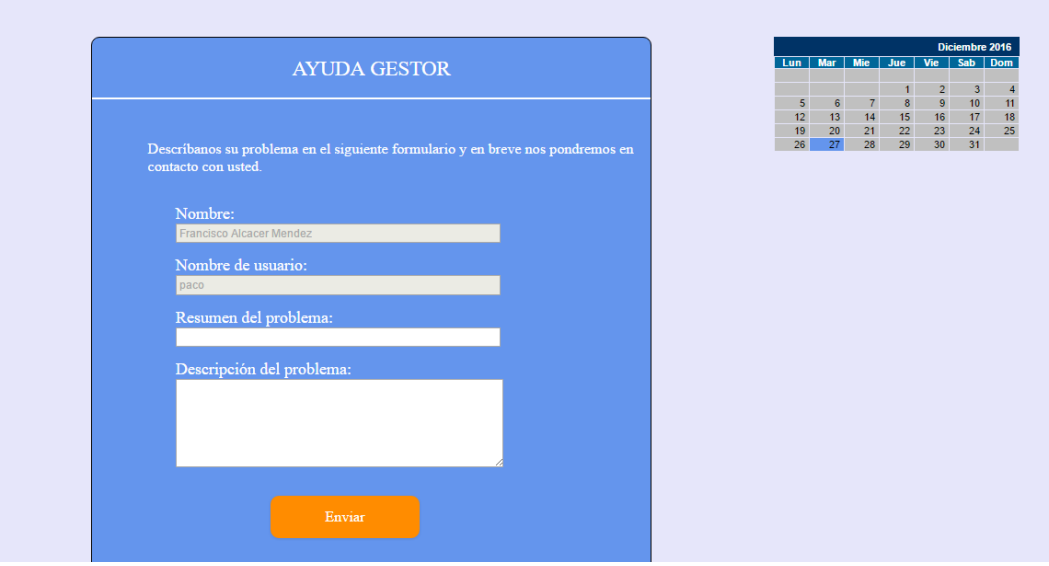

Figura 56 Pantalla Ayuda de la página Web.

ANEXO II

II. Instalación Ruby

En este apartado vamos a explicar cómo se instala Ruby en Windows, ya que es el sistema operativo que tenemos para realizar el proyecto y porque nos es necesario para poder instalar, desarrollar y ejecutar los planes de pruebas en las diferentes herramientas, como son Capibara o Watir [25].

El primer paso que tenemos que hacer es acceder a la página principal de Ruby y descargarnos el ejecutable que corresponda a nuestra arquitectura y seguidamente lo instalamos.

A continuación, en las Figuras 56, 57, 58, 59 y 60, veremos las diferentes pantallas por las que nos encontraremos cuando descargamos Ruby y lo instalamos en nuestra computadora. Una vez realizado este proceso, estaremos perfectamente preparados para poder instalar las herramientas de automatización que utilizan Ruby.

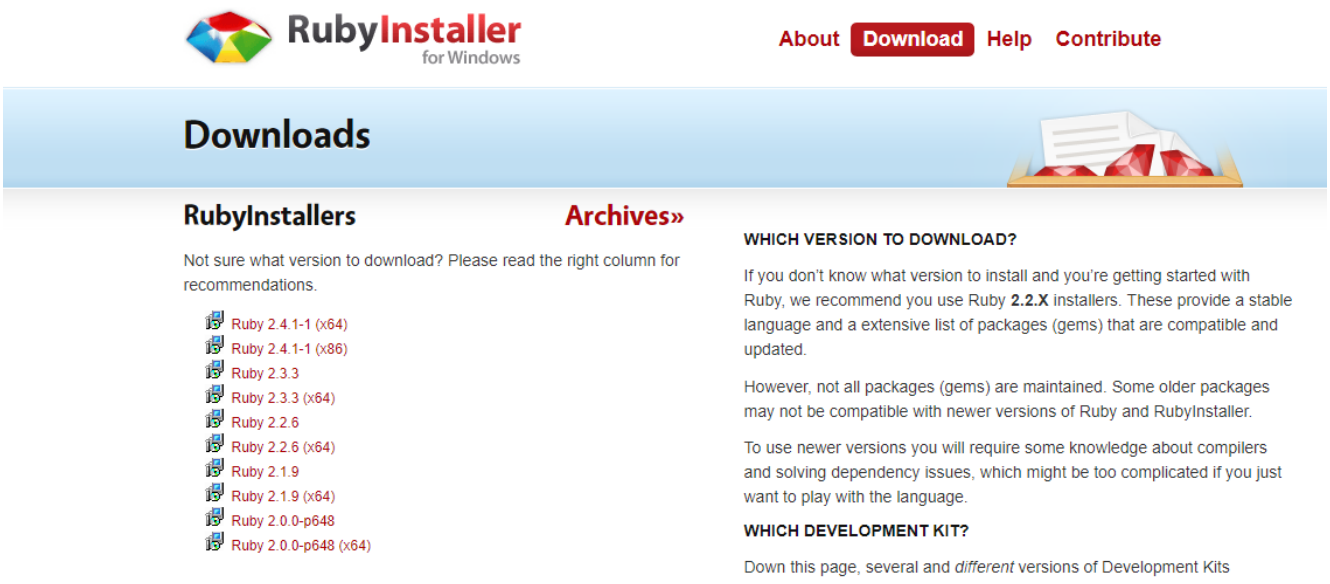

Figura 57. Descarga de Ruby

Continuamos con la instalación, para ello ejecutamos el fichero que acabamos de descargar.

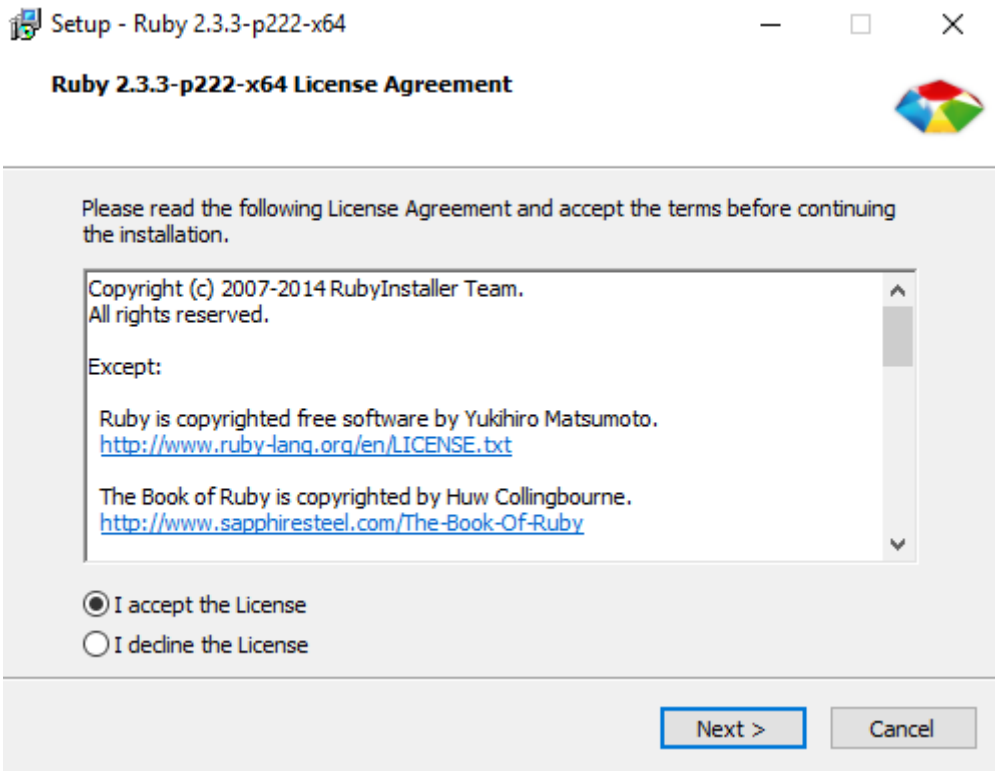

Figura 58. Instalación Ruby I

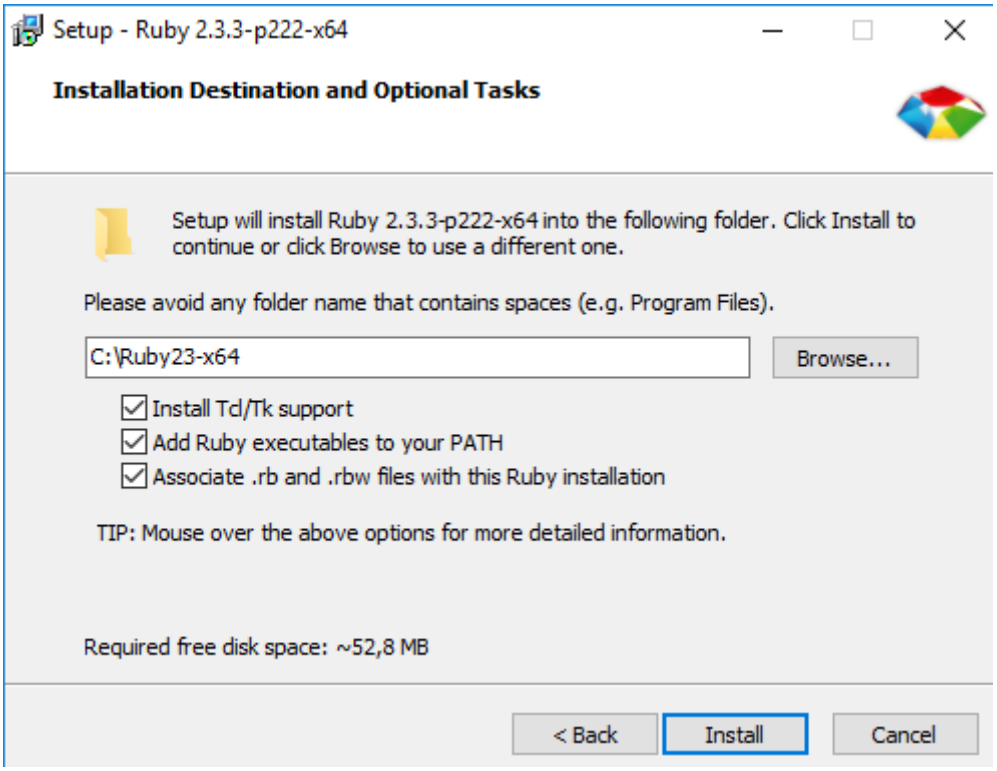

Figura 59. Instalación Ruby II

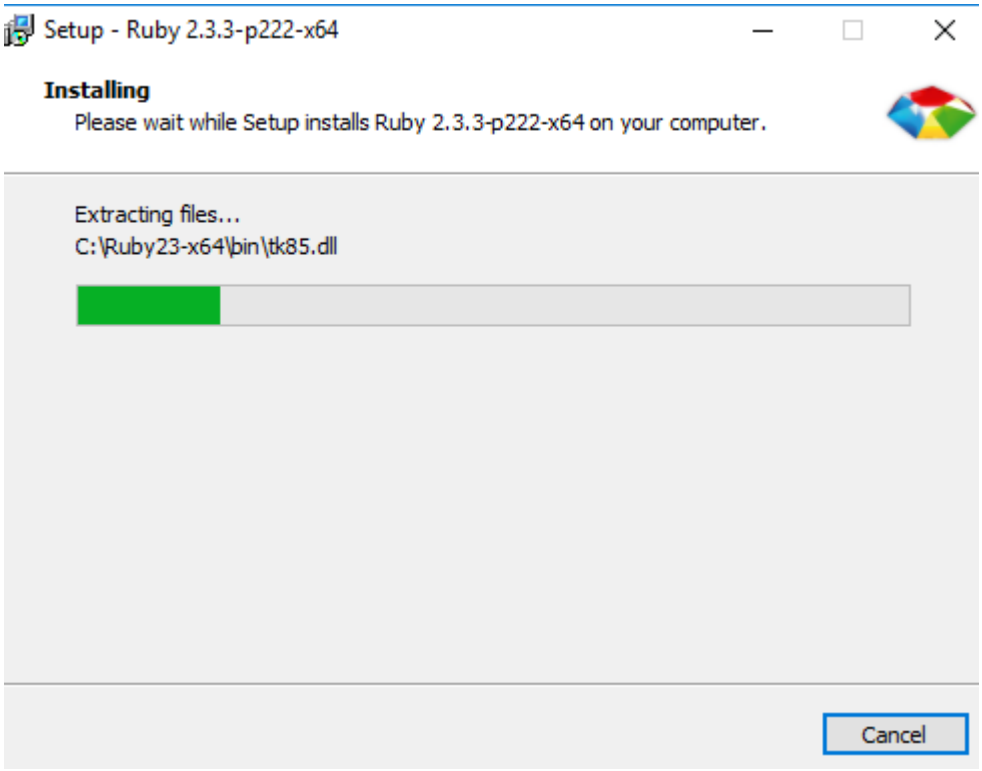

Figura 60. Instalación Ruby III

Figura 61. Fin de Instalación Ruby

ANEXO III

III. Configuraciones Necesarias de los Navegadores

En esta sección se detallará aquellos detalles o peculiaridades a tener en cuenta que hay que realizar en los diferentes navegadores para tenerlos configurados de forma óptima para poder ponernos a desarrollar nuestros planes de prueba.

III.I. Configuración Internet Explorer

En Internet Explorer tendremos que configurar el nivel de modo protegido, es una característica diseñada para dificultar que software malintencionado se instale en nuestro equipo [12]. Para ello habrá que realizar los siguientes pasos.

Primero tendremos que ir a la parte de ajustes de Internet Explorer > Seguridad> Opciones de Internet, a partir de aquí deberemos de seleccionar el checo “Habilitar Modo Protegido (requiere Reiniciar el Internet Explorer)” como se puede ver en la Figuras 61, 62, 63, 64.

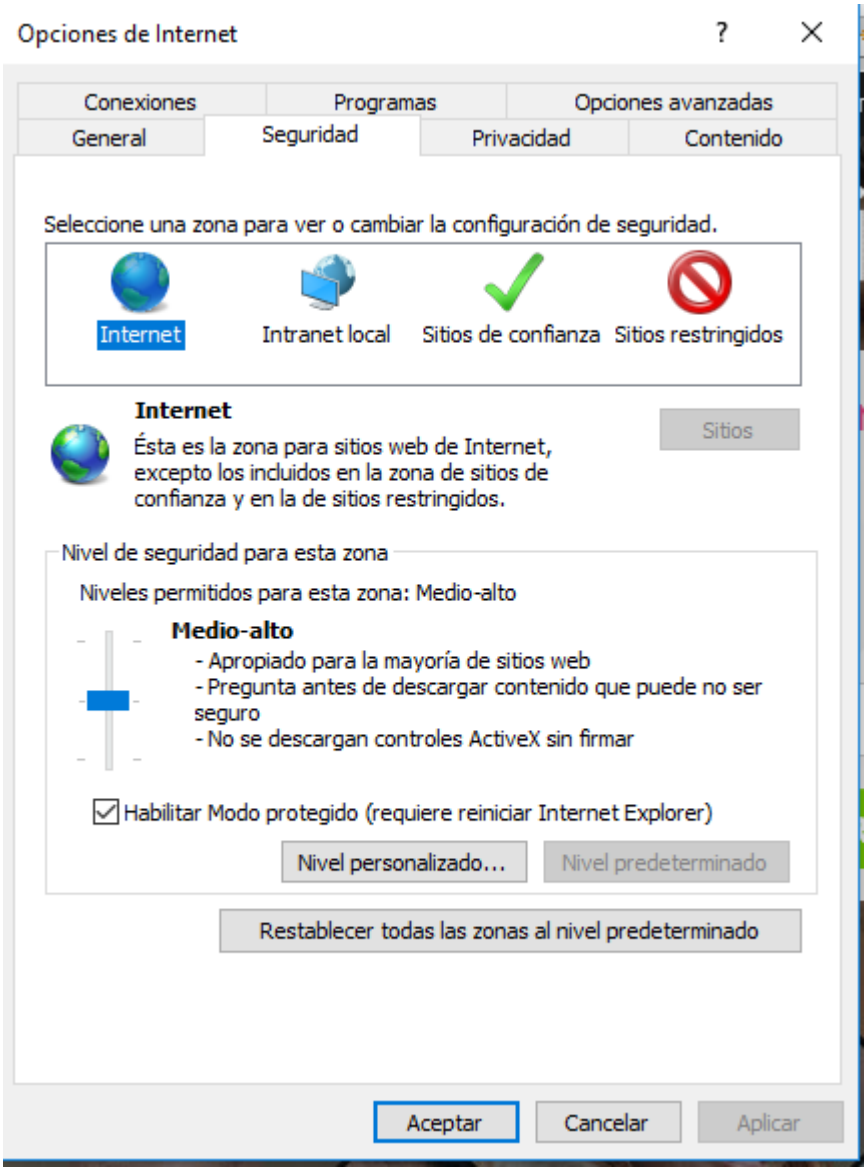

Figura 62.Modo Protegido IE. Internet

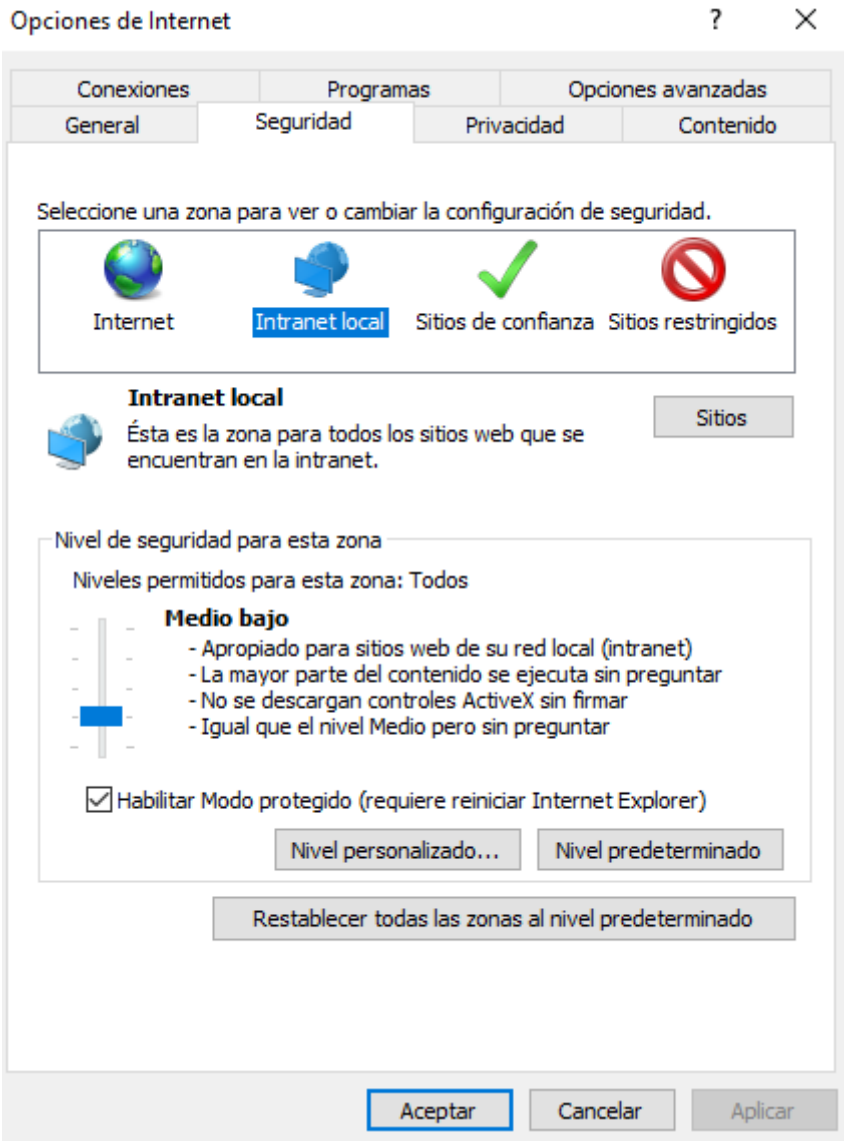

Figura 63. Modo Protegido IE. Intranet Local

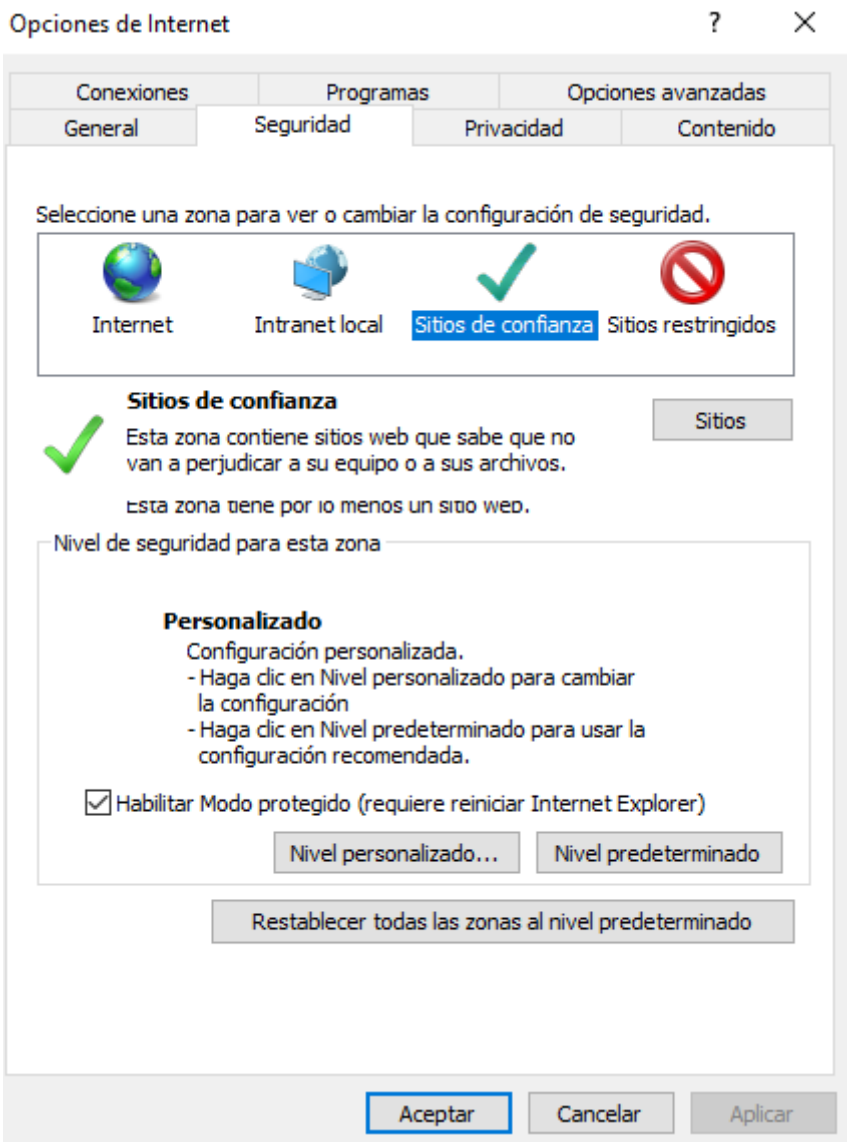

Figura 64.Modo Protegido IE. Sitios de Confianza.

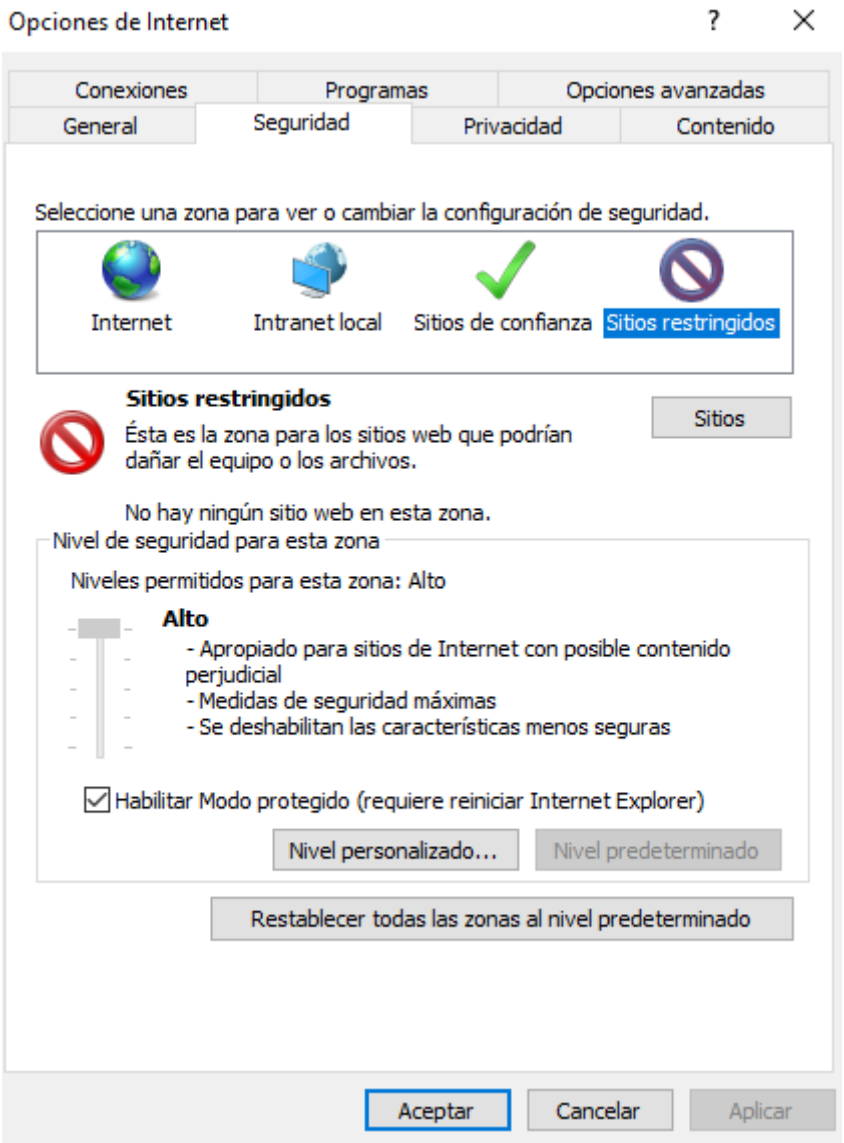

Figura 65. Modo Protegido IE. Sitios Restringidos

Como hemos comentado, para que los cambios tengan efecto habrá que reiniciar el Internet Explorer. Una vez hayamos hecho este paso previo podremos utilizar Internet Explorer sin problemas adicionales para nuestro desarrollo del plan de pruebas.

III.II. Consideraciones en todos los navegadores.

Uno de los puntos a tener en cuenta en todos los navegadores, es que cuando realizamos por ejemplo la línea de código de pulsar un botón, introducir una serie de datos en un campo texto, seleccionar una de las opciones de un campo deplegable, tenemos que pensar que la tarea previa a las anteriores es la búsqueda del campo en cuestión por id, name, link, y demás elementos por los que se puede filtrar la búsqueda. Por tanto, llegados a este punto tenemos que tener en cuenta los siguientes factores:

- El lenguaje de programación que utilicemos para desarrollar los planes de pruebas en cuestión.

- La conexión y calidad de red a la que estamos conectados cuando realizamos la ejecución de las pruebas.

- El navegador y su versión, este punto no es crítico, pero, por ejemplo, en navegadores como Firefox, que actualmente está sufriendo muchas modificaciones y más desde el punto de vista de la seguridad, es algo a tener en cuenta, ya que puede que la versión en cuestión no permita la automatización.

- Arquitectura de nuestro PC. Tendremos que tener en cuenta, a la hora de instalar los drivers la arquitectura de nuestro computador, esto es, si es x86 o x64.

- El número de operaciones o lógica a la hora de seleccionar o buscar una serie de elementos, ya que, si este tiempo creemos que es elevado (depende del navegador), será necesario refrescar la página para que estos elementos sigan activos, de lo contrario nos arrojará una excepción indicándonos que estos objetos han caducado.

Este tiempo previo, hay que contar con él, para configurarlo como margen de timeout a la hora de realizar la búsqueda de dicho elemento en la página web, de lo contrario nuestras pruebas arrojarán la excepción de que no pudieron encontrar dicho elemento y, por tanto, provocando que nuestras pruebas no se satisficieses correctamente.

ANEXO IV

IV. Contenido del CD

En los CDs que se han adjuntado junto con la memoria de este proyecto, seguirá la siguiente estructura:

Tendremos tres carpetas en las que dividiremos en:

- Diagramas, en formato .jpg. y. asta:
    - Casos de Uso
    - Despliegue
    - Actividad

- Memoria en formato pdf.

- Código Fuente de cada uno de los planes de prueba y de cada una de las herramientas:

    - Selenium
    - Watir
    - Capibara
    - Cucumber

- Drivers necesarios para la configuración de los diferentes navegadores.