UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Diseño, desarrollo y evaluación de un sistema automatizado de pruebas de calidad para aplicaciones médicas

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Severí Pinazo, Álvaro

Tutor/a: Martínez Millana, Antonio

CURSO ACADÉMICO: 2022/2023

Agradecimientos

En primer lugar me gustaría dar las gracias a Ricardo y Loli, mis padres, y a mi hermana Laura, ellos han sido el verdadero motivo por el cual he conseguido llegar a día dónde estoy con su apoyo incondicional en todas mis decisiones.

En segundo lugar las gracias a mis abuelos, Paco y Lolin, por el apoyo cada día preocupándose por mí y dándome apoyo en cada examen que tenía.

Por último, Ricardo, Ángeles y Teresa, que aunque físicamente no hayan podido recorrer este camino entero conmigo, sé que lo hacen cada día desde allá dónde se encuentren.

Resumen

Este trabajo aborda la automatización de pruebas para el aseguramiento de la calidad en programas informáticos en el ámbito sanitario. Para ello, la presente memoria describe el marco teórico de la gestión de calidad en entornos software, las funciones que realizan los y las profesionales que se dedican a estas tareas y la importancia que tiene el aseguramiento de la calidad para el producto final. El trabajo incluye un estudio comparativo sobre las ventajas e inconvenientes que tiene realizar las pruebas de funcionalidad del sistema de forma automatizada en lugar de realizarlas de forma manual.

Además, la parte más importante será el estudio sobre las diferentes herramientas que se pueden encontrar en el mercado para conocer cuál es la que mejor se adapta a las necesidades concretas de la aplicación médica sobre la que se enfoca el presente proyecto.

Por último, la memoria describe los resultados del estudio en forma de un caso práctico en el que se pone en funcionamiento el software elegido con la aplicación médica. El caso práctico permite obtener resultados y llegar a conclusiones sobre la utilidad de la automatización de pruebas y el alcance del trabajo desarrollado.

Resum

Aquest treball aborda l'automatització de proves per a l'assegurament de la qualitat en programes informàtics en l'àmbit sanitari. Per a això, la present memòria descriu el marc teòric de la gestió de qualitat en entorns de software, les funcions que realitzen els i les professionals que es dediquen a aquestes tasques i la importància que té l'assegurament de la qualitat per al producte final. El treball inclou un estudi comparatiu sobre els avantatges i inconvenients que té realitzar les proves de funcionalitat del sistema de manera automatitzada en lloc de realitzar-les de manera manual.

A més, la part més important serà l'estudi sobre les diferents eines que es poden trobar en el mercat per a conéixer quina és la que millor s'adapta a les necessitats concretes de l'aplicació mèdica sobre la qual s'enfoca el present projecte.

Finalment, la memòria descriu els resultats de l'estudi en forma d'un cas pràctic en el qual es posa en funcionament el programa triat amb l'aplicació mèdica. El cas pràctic permet obtindre resultats i arribar a conclusions sobre la utilitat de l'automatització de proves i l'abast del treball desenvolupat.

Abstract

This work addresses the automation of tests for quality assurance in computer programs in the health field. To this end, this report describes the theoretical framework of quality management in software environments, the functions performed by the professionals dedicated to these tasks and the importance of quality assurance for the final product. The work includes a comparative study on the advantages and disadvantages of performing system functionality tests in an automated way instead of doing them manually.

In addition, the most important part will be the study on the different tools that can be found in the market to find out which one best suits the specific needs of the medical application on which this project focuses.

Finally, the report describes the results of the study in the form of a practical case in which the chosen software is put into operation with the medical application. The practical case allows obtaining results and reaching conclusions about the usefulness of test automation and the scope of the work developed.

Índice de contenidos

Índice de figuras

Índice de tablas

Abreviaturas

QA: Qualty Assurance, que tiene como significado gestión de calidad.

QC: Quality Control, que tiene como significado control de calidad.

ZK: es un framework de desarrollo. Se usa el ZK5 en el cual los generadores de los componentes HTML hace los identificadores dinámicos.

1. Introducción.

1.1. Justificación.

Uno de los aspectos más importantes en los sistemas de gestión de la información es su buen funcionamiento para los usuarios, y eso conlleva que estén diseñados lo mejor posible y que no tengan errores cuando se ejecutan en el día a día.

Esta importancia se incrementa en el ámbito hospitalario, ya que no solo se trata con números o datos, si no que detrás de cada código o número hay pacientes con diversas necesidades clínicas a las que un fallo en el sistema puede provocar fallos irreversibles. Hoy en día, la gestión de información clínica está prácticamente digitalizada, y los profesionales de la salud emplean múltiples aplicaciones y programas para registrar y consultar información relativa a pruebas, diagnósticos y terapias.

Es por ello por lo que la motivación de este proyecto es crear un entorno de pruebas automatizadas para intentar minimizar al máximo los posibles errores que vayan surgiendo cada vez que se implementen funciones nuevas y actualizaciones de partes ya existentes en el programa para poder corregirlos a tiempo. Esto hará que cada vez el programa sea más seguro y estable.

En paralelo, la automatización de pruebas conlleva ventajas significativas para las empresas que desarrollan aplicaciones médicas. El ahorro de tiempo, y por tanto de costes, por la introducción de automatismos puede ayudar a reducir costes de tiempo y recursos humanos considerables. Esto también afectará al cliente ya que se podrán sacar evolutivos del propio sistema de forma más asidua al reducirse el tiempo en el que el programa se pasa en fase de testeo antes de ponerlo en producción para el cliente.

1.2. Objeto y objetivos.

El objeto del trabajo se ubica en las prácticas que el estudiante ha realizado en la empresa NTT DATA, en concreto en el departamento de ehCOS En el que ha ejercido las labores de desarrollador de aseguramiento de la calidad. El objeto por lo tanto es encontrar una aplicación que permita automatizar las pruebas a las que se someten a los programas informáticos de entorno clínico asistencial. Para ello, se estudiará el marco teórico del aseguramiento de la calidad, las metodologías existentes, las principales herramientas para automatización de pruebas y las ventajas que ello conlleva para el ámbito sanitario.

El objetivo principal de este proyecto consiste en encontrar de manera fundamentada una herramienta que se adapte de forma eficiente a las necesidades planteadas en el departamento de aseguramiento de la calidad del software para la automatización de pruebas.

En nuestro caso concreto deberá tener las características necesarias que cumplan con lo establecido para el programa informático llamado EHCOS, un programa líder en el

sector sanitario y dedicado a soportar los servicios sanitarios de diversos hospitales de todo el mundo.

Este objetivo principal se desarrolla sobre los siguientes objetivos secundarios:

- Estudio bibliográfico sobre metodologías para el aseguramiento de la calidad de software.
- Estudio del contexto regulatorio y legal de las aplicaciones médicas.
- Búsqueda y análisis de herramientas y entornos para la automatización de pruebas de calidad de software.
- Selección de herramienta y caso práctico con una aplicación concreta.
- Análisis de resultados.

Una vez logrado el objetivo principal del trabajo, el departamento tiene previsto adoptar la herramienta seleccionada para analizar las ventajas que puede aportar en cuanto a ahorro de tiempo y mejora de los flujos de trabajo y nuevas necesidades.

1.3. Metodología.

La metodología que se ha llevado a cabo para este análisis se ha dividido en tres fases consecutivas.

La primera de ellas ha consistido en la recopilación de información sobre las pruebas automatizadas, los equipos de QA y las herramientas que tienen disponibles para ello. Esto se ha realizado buscando en diferentes lugares para conseguir el máximo de información posible.

La segunda de ellas ha involucrado la creación de un entorno seguro y la realización de las pruebas mediante una fase práctica con la herramienta seleccionada. Esto se ha realizado en conjunto con la empresa de prácticas.

Por último y con la fase práctica finalizada se procede a extraer todos los resultados obtenidos para la elaboración de unas conclusiones optimas.

1.4. Breve referencia a los ODS.

Las ODS son los objetivos de Desarrollo sostenible adoptados como parte de la Agenda 2030 por los diferentes líderes mundiales en septiembre de 2015.

Son un conjunto de metas globales para afrontar algunos desafíos como los socioeconómicos y ambientales en el mundo, establecidos por las Naciones Unidas. Para alcanzar dichos objetivos es necesario la tecnología y el conocimiento, entre muchas otras áreas [1].

Como estos objetivos deberían estar presentes en nuestro día a día, es posible relacionarlos con cualquier tarea que realicemos. En nuestro caso podemos observar como la mejora de los sistemas informáticos hospitalarios favorecen a la mejora de varios de ellos.

En primer lugar con el ODS 3, relacionado con la salud y el bienestar ya que es un producto diseñado para mejorar la salud del cliente. Esto se debe a que uno de los indicadores de dicho objetivo es "la capacidad del Reglamento Sanitario Internacional (RSI) y la preparación para emergencias de la salud". La meta que se debe conseguir con ello será "Reforzar la capacidad de todos los países, en particular los países en desarrollo, en materia de alerta temprana, reducción de riesgos y gestión de los riesgos para la salud nacional y mundial" [2].

Figura 1. Objetivos de las ODS [3].

1.5. Asignaturas relacionadas.

El proyecto de automatización de pruebas lo podríamos relacionar con dos asignaturas vistas en la carrera de la rama de electrónica, en segundo y tercer curso, ambas relacionadas entre sí.

La primera de ellas es SDP (Sistemas digitales programables) que mediante el lenguaje verilog es dónde se empiezan a ver los primeros bancos de pruebas, en este caso son pruebas por código a diferentes componentes como contadores o FSM.

Isdigi (Integración de sistemas digitales) es la continuación a la anterior asignatura mencionada, dónde siguiendo con la línea de bancos de prueba en verilog se profundiza más en los conceptos, hasta lograr hacer un testbench que compruebe el funcionamiento del microprocesador.

1.6. Estructura del trabajo.

Para llevar a cabo este proyecto vamos a separarlo en seis capítulos.

En el capítulo 2 se describen los conceptos relacionados con la gestión de calidad en los proyectos y sus diferentes ramas. Además también se da una breve introducción sobre las aplicaciones médicas.

En el capítulo 3 se abordan las diferentes herramientas para llevar a cabo la tarea y se elige la que mejor se adapta a nuestras necesidades específicas.

En el capítulo 4 se desarrolla como se ha llevado la implementación de la herramienta de forma práctica en nuestro proyecto.

En el capítulo 5 se describen los resultados que hemos obtenido al llevar a cabo dichas pruebas y los datos que podemos obtener de ellas.

En el capítulo 6 se abordan las conclusiones referentes a todo lo que hemos estudiado y cómo podemos proseguir con el trabajo en un futuro.

2. Marco teórico y conceptual.

En este apartado se aborda el marco conceptual de la gestión de calidad en software, entendiendo como tal el proceso de evaluación de las aplicaciones cuando se ejecutan para cumplir con los fines por los que fueron implementadas. Para ello, en primer lugar se describe cuáles son las diferencias entre los controles de calidad y la gestión de calidad en los proyectos de desarrollo de software y los diferentes métodos que tienen a su alcance para poder realizar sus diferentes funciones para que sean lo más efectivas y eficientes posibles.

Además también se describirán que función tienen las aplicaciones de ámbito sanitario en nuestra sociedad y que requisitos tanto legales como de calidad deben pasar para poder disponer de una garantía en dichas herramientas para poder ofrecérselas a los clientes de los diferentes territorios.

2.1. Gestión de la Calidad. Qué es QA.

Para empezar a conocer que es la gestión de calidad deberemos entender primero que es la calidad. Según el diccionario la palabra calidad significa: conjunto de propiedad inherentes a una cosa que permite caracterizarla y valorarla con respecto a las restantes de su especie [4].

El tipo de propiedades que se evalúen dependerán del tipo de producto, por ejemplo, podemos hablar de productos de mejor calidad que otros debido a que son más resistente, o que tienen un mejor aspecto visual o es más ergonómico que otro, pero en el caso del software no es tan trivial y las propiedades pueden ser difíciles de determinar. Esto se debe a que un usuario puede valorar que un programa es de mejor calidad si realiza determinadas funciones y otra le puede dar más importancia a que sea más eficiente en otro aspecto.

Es por ello por lo que para establecer los criterios que definen la calidad del software, determinar las propiedades y qué grado de cumplimiento deben tener estos requisitos se establecen unos mecanismos llamados Control de Calidad (QC). En las empresas de desarrollo de software existe un equipo de personas llamado QA, que son unas siglas las siglas de Quality Assurance o Garantía de Calidad que se centran exclusivamente en definir y evaluar métricas de calidad [5].

Una vez definido el concepto de calidad en software, se abordan las diferencias que encontramos entre el Control de Calidad (QC) y la Garantía de Calidad (QA), ya que son conceptos que están muy relacionados entre sí pero a menudo se suelen confundir [6].

Quality Control	Quality Assurance
El objetivo principal es asegurar que el software cumple con los estándares y especificaciones de calidad establecidos para ello. Detecta los defectos una vez han ocurrido.	El objetivo principal es asegurar que el software satisface las necesidades y expectativas de los clientes. Intenta prevenir los defectos antes de que ocurran.
Tiene un enfoque centrado en las pruebas y está orientado a verificar los estándares de calidad.	Tiene un enfoque centrado en procesos para mejorar y está orientado a validar si cumple con las normas y procedimientos establecidos.
El control de calidad es verificado una vez se ha desarrollado el software.	La gestión de calidad se lleva a cabo durante todo el ciclo de vida de desarrollo del software, no solo al final.

Tabla 1. Comparación entre control de calidad y gestión de calidad [7].

Aunque vemos que no son el mismo término, en numerosas ocasiones los equipos de QA se encargar también de gestionar el QC.

Figura 2. Esquema QA y QC [8].

Una vez sabemos la diferencia entre ambos términos veremos cuales son los cuatro principios básicos de la Gestión de Calidad (QA) [9].

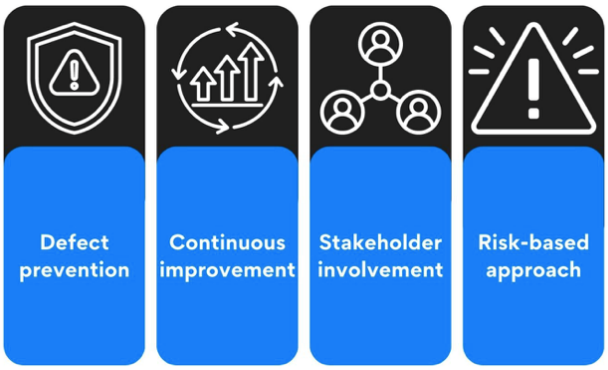


Figura 3. Principios básicos QA [10].

- Prevención de defectos: estipula que es mejor intentar prevenir defectos que tener que corregirlos a posteriori, por lo que se abordarán los principales problemas al inicio del desarrollo y por diseño.

- Mejora continua: consiste en monitorear y mejorar constantemente la calidad del producto y no solo al final de su vida de desarrollo, es algo que debe ir a la par que el desarrollo de este.

- Participación de las partes interesadas: debe involucrar a todas las partes interesadas como clientes, desarrolladores, managers y demás. También hay que tener una comunicación fluida entre todas las partes para un mejor funcionamiento.

- Enfoque de riesgos: debe centrarse en los riesgos más importantes y priorizarlos según su impacto en el producto [11].

Figura 4. Importancia de QA [10].

Por último, se debe analizar la importancia de tener un equipo de QA en nuestro proyecto:

- Garantizar una alta calidad: esto hace que tengamos un producto más eficiente y los clientes confíen más en él.

- Ahorro de tiempo y dinero: como se evalúa durante todo el ciclo de vida cuanto antes detectemos errores menos tiempo invertiremos después en subsanarlos y como consecuencia más rentable será para la empresa. Esto hace que seamos más competitivos a la hora de luchar para conseguir nuevos clientes.

- Software estable: al probar tan a fondo el producto conseguiremos una de las cosas más importantes que valoran los clientes, que sea estable y no les aparezcan a ellos los errores.

- Reputación: tendremos la confianza de nuestros clientes al presentarles un sistema sin errores ya que los habremos detectado y corregido a tiempo sin que ellos lo perciban.

- Seguridad: deberemos cumplir con las normativas y leyes, haciendo hincapié en aquellas que traten sobre la seguridad y sobre la privacidad de los datos.

- Satisfacción del cliente: debe cumplir con todas las necesidades para satisfacer a los clientes.

Por lo tanto, no se puede subestimar la importancia del aseguramiento de la calidad del software. Llevar a cabo un QA exhaustivo es un paso vital para lanzar un producto de software exitoso.

2.2. Metodologías para QA

Para crear el banco de pruebas que desarrollaremos más adelante en la parte práctica debemos tener en cuenta a que tipo de pruebas queremos enfocar nuestra prueba. Para ello vamos a ver de forma breve los diferentes tipos de pruebas que existen [12]:

- Pruebas Unitarias: verifican el comportamiento individual de una parte del sistema muy concreto.
- Pruebas de Integración: verifican el correcto funcionamiento y la interacción de diferentes módulos del sistema.
- Pruebas de Regresión: verifican que después de una actualización de software sigue funcionando todo el sistema y no ha habido ningún error de algo que previamente funcionaba.
- Pruebas de Rendimiento: verifican el rendimiento y las capacidades del sistema ante diferentes cargas de trabajo.
- Pruebas de Seguridad: evalúan la seguridad frente a posibles vulnerabilidades del sistema para hacerlo más robusto.
- Pruebas Funcionales y Usabilidad: verifican que el software cumple con los requisitos establecidos en cuanto a funcionalidades y experiencia de usuario.

Sobre los tipos de pruebas vistos anteriormente nos centraremos en estos últimos, en las pruebas funcionales y de usabilidad. Para ello veremos las diferentes formas de llevarlas a cabo.

2.2.1. Pruebas Manuales

Las pruebas manuales son aquellas en las que se encuentra un técnico de QA probando manualmente una a una, es decir, realizando uno a uno los pasos que indica la prueba en el programa a probar en cuestión.

Consiste en una forma muy rudimentaria de hacer pruebas, ya que en un entorno tan grande como el que estamos tratando en este proyecto hay más de 2.000 casos de prueba a realizar y muchas veces no existe el tiempo óptimo que se le debería dedicar a realizar las pruebas.

Este método es muy costoso en cuanto a tiempo ya que implica una gran cantidad de horas realizarlo y por consiguiente un gran coste económico.

2.2.2. Pruebas automatizadas

Las pruebas automáticas son aquellas que ya están automatizadas y se ejecutan de manera automática y autónoma. Para ello primero se debe conocer el concepto de automatización, que según la RAE es convertir ciertos movimientos en automáticos o indeliberados. Esto quiere decir que las pruebas se realizan solas sin necesidad de que una persona este haciendo todos los pasos de cada uno de los casos de prueba [13].

La principal ventaja que tenemos al utilizar este método es justo la diferencia que hay con el anterior, a las pruebas realizarse de forma sola no es necesario que una persona del equipo de QA este todo el tiempo realizándolas él, si no que con solo con activar la ejecución al principio y al final recoger los resultados y analizarlos para detectar los fallos sería suficiente.

Este método supone un gran ahorro de tiempo ya que la persona puede dedicar la mayor parte de su tiempo a realizar otras labores o centrarse en los casos más complejos que no puedan ser automatizados. Además, supone también un gran ahorro en el coste económico del proyecto que puede ser destinado a otras partes de él, como focalizarse para atraer más ventas o mejorar en desarrollo por ejemplo.

2.3. Automatización de procesos de evaluación

Debido a las grandes ventajas mencionadas anteriormente de la automatización de diversas tareas están presentes en nuestro día a día, ya sea desde una pequeña tarea de los profesores de calcular las notas de sus alumnos con programas específicos para ello como el caso que nos trae a nosotros de los bancos de pruebas. Por eso en la actualidad existen varias formas de automatizar las pruebas que usando distintos métodos se llega a la misma conclusión.

A continuación, se comparan las dos formas más comunes: por código o scraping.

2.3.1. Automatización por código

La automatización por código consiste en el uso de entornos y técnicas para automatizar tareas y procesos previamente definidos. A diferencia de las pruebas manuales, la automatización permite realizar de forma automática y repetitiva pruebas en los programas a evaluar. La automatización por código requiere una comprensión profunda de conceptos relacionados con la programación y el procesado de la información, y especialmente, conocimientos técnicos específicos del campo de aplicación – en el caso de estudio la aplicación sanitaria.

Para tener una automatización por código lo primero que tendremos que crear es un testbench. El testbench en este caso será un programa dónde se diseñará y se especificarán las acciones que se quieren probar del programa original, o programa bajo test, y con qué configuraciones o parámetros de entrada se llamarán a estas funciones. En este caso no tendrá una interfaz gráfica, simplemente veremos un código escrito en el lenguaje de programación dónde lo hayamos realizado, llamando a las funciones específicas

En dicho programa de pruebas tendremos una parte dónde se llamará a las funciones del código original y otras donde se recolecten los resultados de la ejecución de dichas tareas con los datos que se pasaron como argumento.

2.3.2. Scraping

El término scraping es la forma abreviada de llamar a Web Scraping, y aunque tiene muchas funcionalidades y aplicaciones, en el contexto de este proyecto la más útil será la de la automatización de pruebas por navegador, es decir, pruebas a través de la interfaz del usuario.

Para ello la idea principal será grabar una serie de flujos como si los hiciéramos de forma manual dentro del programa que más adelante seleccionaremos para realizar las pruebas.

Para entender que realizará el programa cuando tenga que automatizar las pruebas será repetir los mismos pasos que nosotros le hemos dejado grabados realizándolos anteriormente:

- Primero haciendo una petición para acceder a la URL, es decir, visitando la web del sistema.
- Extrayendo los datos de las partes que queremos probar, aquí será dónde dependerá de la parte del sistema y de las pruebas que estemos realizando.
- Guardando las conclusiones por si ha habido algún error poder detectar dónde se ha producido para subsanarlo lo más pronto posible [14].

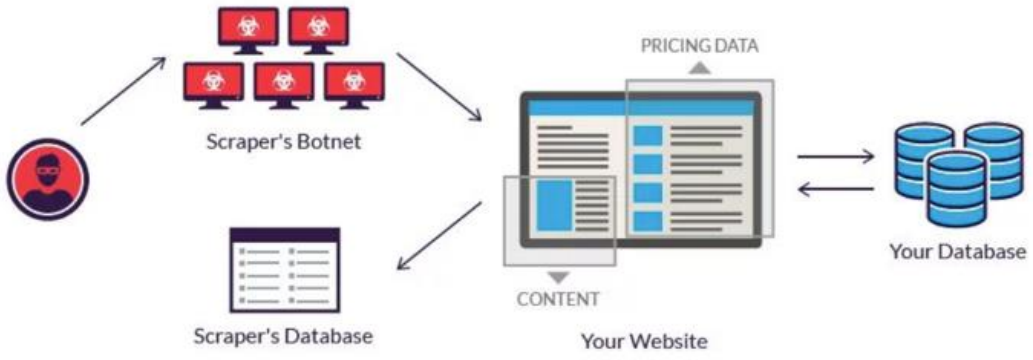


Figura 5. Pasos para seguir en el Scraping [14].

2.4. Aplicaciones médicas

Durante los últimos diez años los sistemas sanitarios se han digitalizado progresivamente hasta alcanzar, prácticamente, una digitalización total de la información sanitaria. Todos los hospitales de países desarrollados o en vías de desarrollo cuentan con sistemas informáticos de gestión pura o sistemas enfocados al manejo de toda la información clínica relacionada con el paciente.

Por la naturaleza del sistema sanitario y la evolución de la atención sanitaria que se presta a los ciudadanos, estos sistemas informáticos están en constante evolución y son múltiples empresas las que compiten para proporcionar estos servicios mediante licitaciones o pliegos de explotación.

En este contexto tan dinámico, las empresas deben disponer de herramientas lo más fiables posibles que permitan verificar y validar los programas que despliegan en infraestructuras tan críticas como en un ambulatorio o en un centro hospitalario, en las cuales el menor fallo puede llegar a provocar una situación irreversible y suponer un problema para el profesional sanitario, un fallo en el propio sistema o en el peor de los casos un problema para el paciente. A parte, el entorno de las tecnologías sanitarias, entre las que figuran las aplicaciones informáticas para la provisión de servicios

sanitarios, dispone de un marco regulatorio muy estricto, definido por directivas europeas con transposición nacional directa, en las cuales se especifica de manera muy clara cómo debe realizarse el aseguramiento de la calidad. El incumplimiento de estas normas puede tener consecuencias muy negativas, abarcando desde la imposibilidad de comercialización hasta penas económicas muy cuantiosas.

2.5. Marco regulatorio de aplicaciones médicas

En el mundo existe leyes cuyo objetivo es asegurar que todo funcione de manera adecuada y justa. Las leyes que podemos encontrar en nuestro día a día se encuentran en todos los ámbitos, que establecen unos límites derechos y responsabilidades para un buen desarrollo de la sociedad.

Por ello no es menos que en ámbito de los ecosistemas sanitarios también existen una serie de leyes que los regulan, con el objetivo de crear un ambiente más seguro y lo más beneficioso posible para el paciente.

Estas leyes no son universales, es decir, varían dependiendo del territorio en el que nos encontremos. Es por ello por lo que las empresas dueñas de las infraestructuras sanitarias no solo deben adaptarlas a las necesidades de los clientes, si no también debe tener en cuenta la legislación vigente en el lugar dónde se encuentran, así pues un mismo ecosistema puede tener unas funciones diferentes dependiendo el lugar dónde se encuentre.

A continuación, vamos a nombrar algunas de las leyes más importantes que regulan estos sistemas en España son:

- Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPDGDD): Esta ley establece el marco normativo para la protección de datos personales en España y garantiza los derechos digitales de los ciudadanos. Es importante tener en cuenta que la LOPDGDD se aplica en conjunto con el Reglamento General de Protección de Datos (RGPD) de la Unión Europea [15].

- Ley 41/2002, de 14 de noviembre, básica reguladora de la Autonomía del Paciente y de Derechos y Obligaciones en Materia de Información y Documentación Clínica: Esta ley establece los derechos y obligaciones de los pacientes en el ámbito sanitario, incluyendo aspectos relacionados con la información y documentación clínica, así como la confidencialidad de los datos médicos [16].

- Ley 14/2007, de 3 de julio, de Investigación Biomédica: Esta ley regula la investigación biomédica en España, incluyendo aspectos relacionados con la protección de datos personales y la ética en la investigación [17].

- Ley 28/2009, de 30 de diciembre, de modificación de la Ley 29/2006, de 26 de julio, de garantías y uso racional de los medicamentos y productos sanitarios: Esta ley establece las normas para el uso de medicamentos y productos sanitarios en España, incluyendo aspectos relacionados con la gestión de datos y la seguridad de la información en el ámbito sanitario [18].

Además, en el caso de España está repartido su territorio en comunidades autónomas. Existen algunos ámbitos de la política que se delegan a las comunidades, como es el caso de las competencias a nivel de sanidad, que cada comunidad autónoma los gestiona independientemente, por lo que no solo existen las leyes nombradas anteriormente que regulan el ámbito sanitario a nivel nacional, si no que existen otras leyes propias de cada Comunidad Autónoma que legislan en conjunto con:

- Ley 10/2014, de 29 de diciembre, de la Generalitat, de la Generalitat Valenciana, de Salud de la Comunidad Valenciana: Esta ley establece el marco general para la protección de la salud y la regulación del sistema sanitario en la Comunidad Valenciana [19].

Además de cumplir las leyes comentadas anteriormente, como muchos de los productos que se venden, deben cumplir unos estándares de calidad para que puedan ser implementados. En el caso de los dispositivos médicos deben cumplir las normas de calidad ISO 13485, las cuales provienen del estándar de calidad internacional que establece los requisitos para un sistema de gestión de calidad en una organización, la ISO 9001.

Los requisitos adicionales que ofrece la ISO 134 son algunos como el diseño y desarrollo de dispositivos, control de procesos, control de documentación, gestión de riesgos, validación de procesos, capacitación del personal, control de proveedores y vigilancia posventa.

Cumplir dicha norma es el camino más común para cumplir con la gestión de calidad (QMS) para dispositivos médicos en lugares como Europa, Canadá y Australia. Además, esto también nos servirá a comercializar más fácilmente nuestro sistema en otras regiones que no se obligatorio dicha certificación [20].

3. Materiales

En este capítulo se va a describir el entorno específico que se ha desarrollado para llevar a cabo la automatización. Además, también se hará una selección entre las diferentes herramientas disponibles en el mercado y por último se conocerá la aplicación que queremos aplicar la automatización.

3.1. Entornos de desarrollo para QA.

El desarrollo de aplicaciones sanitarias es muy dinámico y constantemente aparecen nuevos requisitos o nuevas necesidades. Es por ello, que los clientes de estas aplicaciones buscan mejoras en los sistemas constantemente tanto de corrección de errores como de mejoras para el programa, y por lo tanto el entorno de test debe estar en constante evolución para atender las necesidades de dichos clientes.

Tener a disposición este servicio a los clientes implica el acceso a los servidores de muchas personas de los diferentes equipos que conforman el proyecto. En ese caso sería necesario que las personas que se encargan del desarrollo estuvieran modificándolo para hacer su trabajo, o bien que el equipo de arquitectura estuviera haciendo pruebas de cargar las evoluciones o borrar los correctivos subidos, o por último las personas que forman el equipo de QA estuvieran haciendo pruebas manuales y modificando parámetros de la configuración.

Es por todo esto que no es viable hacer las pruebas automatizadas en un entorno en uso por todos, ya que la herramienta que se encarga de automatizar debe seguir unos pasos que nosotros le hemos marcado previamente con una configuración muy concreta, y cualquier mínimo cambio hará que la prueba de un resultado erróneo porque no podrá acabar de ejecutarse.

Para ello la opción que se ha optado es crear un entorno seguro, entendiendo por entorno seguro aquel que solo va a ser usado para la ejecución de las pruebas automatizadas y por lo tanto, dónde solo va a acceder el equipo que se encargue de grabar las pruebas.

Con esta premisa, se asegura que el primer problema planteado anteriormente está resuelto y que las pruebas automatizadas no van a dar un fallo en su ejecución por haber hecho algún cambio de configuración en el mismo.

Sin embargo, este no es el único problema que se puede producir con el entorno, otro de los problemas graves que se nos planteó a la hora de empezar con la tarea de automatización fue el tema de los identificadores que se quedan guardados de cada una de las acciones que se realizan en el programa.

El programa de ehCOS, como todas las aplicaciones web, está desarrollado de una forma que cada acción que se pueda hacer en la pantalla, desde un simple clic en un botón

hasta escribir un texto en una casilla lleva un identificador, es decir, que cada acción queda registrada con un valor que lo identifica, a lo que llamaremos a esto ZK.

La peculiaridad viene a la hora de trabajar con ese identificador en este programa, ya que no es un valor fijo si no que va variando, es decir, que se procede a reiniciar el sistema por cualquier motivo los identificadores cambian por completo y la misma acción antes y después de reiniciar el servidor no va a tener el mismo valor de identificación.

Esto hasta ahora no había supuesto ningún problema ya que en los casos de las pruebas manuales en los que el sistema no funcionaba correctamente y saltaba un mensaje de error, se procedía a mirar en ese mismo instante dónde se encontraba el código de error que aparecía por pantalla y a que acción se refería. Con esto se guardaba el log del código entero que había generado el error, no el identificador que varía, y esto servía para adjuntarlo junto a la incidencia que se creaba en el programa de gestión de incidencias.

Lo que para las pruebas manuales no era un gran problema sí que se había convertido a la hora de querer automatizar las pruebas, ya que la idea principal era que las pruebas automatizadas se ejecutaran cuando no hay nadie usando los entornos, es decir, de noche, y los resultados quedarán guardados para poder analizarlos al día siguiente para ver si había funcionado todo bien o si había ocurrido algo.
Esto no sería muy útil si no se hacía ningún cambio porque los identificadores se usaban para dos cosas: para grabar todos los pasos de los casos de prueba y para analizar los errores de las cosas que no funcionaban.

Si los ZK continuaban siendo algo variable en el momento el sistema se reiniciará, ya que era algo que se hacía asiduamente por varias razones, entre ellas por que el sistema estaba saturado e iba a una velocidad inferior a la que debería realizar las acciones o por que había que hacer mejoras, todo el trabajo que se había hecho no tenía sentido ya que había que volverlo a hacer con los nuevos identificadores

La solución a este segundo problema en cuanto a la teoría parecía algo sencilla, había que cambiar los identificadores para que fueran más estáticos y a la hora de reiniciar el sistema las acciones tuvieran el mismo valor que antes, así las pruebas automatizadas podrían ejecutarse de forma independiente a las veces que se reinicie el servidor.

Para ello se han creado unas librerías de Java que se han unido a las que ya había existentes en el proyecto. La función de estas librerías es que cuando se realiza una acción en la pantalla el atributo que se le asigna no es el que tenía la librería original que había subida en el entorno, sino que la librería original apunta a la librería nueva que nosotros hemos creado y subido.
Esto nos permite que esas acciones que nosotros queremos cambiar el identificador consulten a nuestras propias librerías, dónde se les asignará lo que nosotros queramos, en este caso un ZK más estático.

Este cambio, aunque podía haber sido implementado en todos los entornos que tiene el sistema, se ha decidido que la mejor opción es que solo se implante de momento en el entorno dónde se van a ejecutar las pruebas automatizadas, ya que no es algo prioritario adaptarlo a todos los entornos que se están usando actualmente. Esta decisión se ha tomado para prevenir la creación de errores en los otros entornos que están en uso, ya que es una mejora que pida el cliente porque no es visible para él, sino que es un cambio para una forma de trabajo interna para tener más facilidades a la hora de agilizar la parte de las pruebas del sistema.
No obstante, es algo que se puede realizar en un futuro, pero que no es prioritario en estos momentos.

3.2. Criterios y selección de la herramienta.

En relación con los entornos para pruebas de QA existen en el mercado para esta tarea nos surgieron varias opciones. Como la aplicación a testear es un programa propio, el entorno de test debía tener una serie de requisitos para que se adaptaras a nuestras necesidades:

- La primera de las características que se buscaba en la herramienta era que resolviera los problemas planteados anteriormente de los identificadores variables.

- La segunda de ellas es que tuviera la opción de grabar los casos de prueba, ya que así sería más sencillo, poder consultar la grabación para ver que parte no se había ejecutado de forma correcta.

- La tercera característica era buscar la opción de una herramienta con o sin licencia. En principio no debía ser algo determinante ya que el producto estaba dispuesto a asumir el coste que ello supusiera, siempre y cuando el precio fuera algo razonable, ya que el beneficio económico que iba a suponer el ahorro de tiempo al hacer todas las pruebas a mano sería mucho mayor que el pago de dicho programa; aunque si encontrábamos un programa opensource sería mucho mejor.

En base a esto se ha hecho una selección de los programas que en un principio son más útiles para nuestras necesidades y se va a analizar las ventajas e inconvenientes de cada uno de ellos para poder encontrar el que mejor se adapte a las peculiaridades de EHCOS.
Los programas para analizar serán los siguientes:

- Flood IO.
- LoadNinja.
- WebLOAD.
- Katalon.
- Selenium IDE.

3.2.1. Flood IO.

Para entender mejor la función de esta herramienta hay que entender que quiere decir su nombre que es un término en inglés que significa inundación. Esto trasladado al ámbito de la informática se refiere a una técnica que su principal objetivo es saturar o inundar una red cuyo objetivo es probar el rendimiento del sistema.

En esta herramienta nos encontramos también todos los inconvenientes mencionados anteriormente.

En primer lugar se trata de un sistema de pago por lo que no podríamos acceder al código para hacer nuestro ajustes. En segundo lugar se trata de un sistema en la nube por lo que al probarlo en nuestros entornos se ha visto que no se nos ha ejecutado nada ya que los tenemos de forma local. Además de que se trata de una herramienta de solo pruebas de carga [21].

Figura 6. Ejemplo de test carga en Flood.io [22].

3.2.2. LoadNinja.

Se trata de una herramienta para evaluar el rendimiento y la carga para aplicaciones web dónde permite medir el comportamiento de cómo trabaja el sistema bajo cargas intensas. Este software es desarrollado por la empresa SmartBear Software y está diseñado expresamente para hacer tests de carga por lo que se intentó probar en nuestra aplicación y no pasaba de la portada.

Otra de las características que tenía LoadNinja es que no se trataba de una herramienta opensource, si no que había que pagar una licencia para poder acceder a ella. Lo que para el proyecto en un principio no iba a ser un inconveniente luego sí que se iba a convertir en uno ya que al ser de pago no se podrían hacer los ajustes oportunos para modificar los identificadores variables [23].

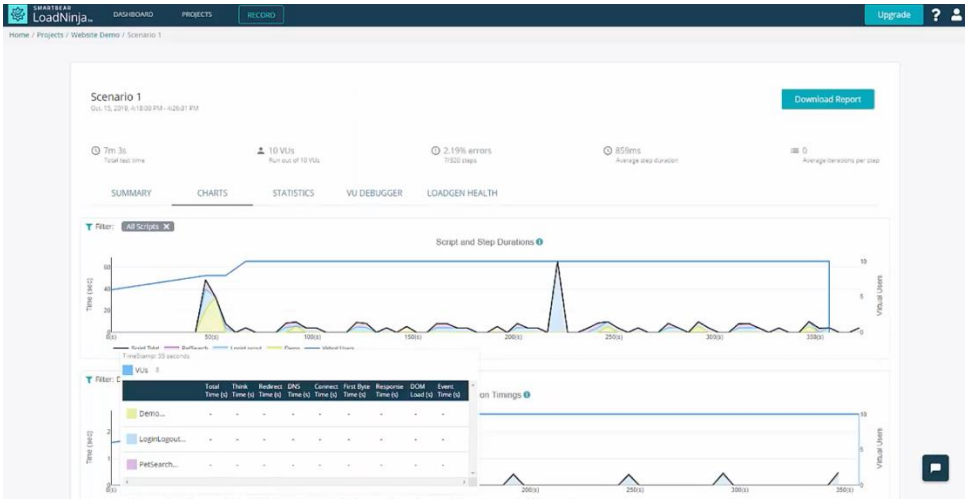


Figura 7. Ejemplo de test carga en LoadNinja [24].

3.2.3. WebLOAD.

WebLOAD es también una herramienta diseñada para hacer pruebas de rendimiento y carga a servicios y aplicaciones web. Dicha herramienta está desarrollada por la empresa RadView Software.

Al probar la herramienta e instalarla en nuestros ordenadores nos dimos cuenta que no nos dejaba entrar a nuestro sistema e iniciar la sesión.

Figura 8. Esquema de cómo trabaja WebLOAD [25].

La problemática que nos encontramos al tener que usar esta herramienta es un requisito técnico que tiene ehCOS por seguridad. En nuestro caso los entornos que se tienen de

rendimiento no están en la nube, es decir, no se encuentran expuestos en internet. Estos entornos están ubicados en máquinas físicas que se encuentran en el datacenter que está ubicado en las oficinas. Esto se hace por una cuestión de seguridad que no sean accesibles desde internet.

Además, también se trata de una herramienta dónde hay que pagar una licencia para poder usarla y en el caso de que hubiera funcionado nos hubiéramos topado con la misma problemática que la herramienta anterior en cuanto a los ZK variables.

3.2.4. Katalon

Se trata de la herramienta que más se ajustaba a nuestras necesidades y que además se llegó a poner en funcionamiento de forma correcta.

Katalon es una herramienta que sirve tanto para hacer pruebas de rendimiento, es decir, hacer los test de carga que hacían las anteriores herramientas como para hacer test de prueba.

A diferencia de las otras herramientas, esta sí que nos permitía grabar los flujos para poder después reproducirlos para ver si funcionaban y además estaba integrado con JIRA que es la herramienta que se utiliza para la gestión del proyecto.

Uno de los inconvenientes que se encontró en esta herramienta es que no era de código abierto si no que había que pagar una licencia. Esto por una parte era un inconveniente porque al final era un gasto para el proyecto pero si se acoplaba a nuestras funcionalidades iba a ser la elegida.

Finalmente no lo fue porque al poco tiempo de realizar las primeras pruebas y ver que todo funcionaba llegó la noticia que la herramienta iba a dejar de tener soporte por lo que no era muy lógico empezar en un proyecto así si la herramienta no iba a tener más continuidad por lo que por eso se decidió descartara aunque aparentemente funcionara.

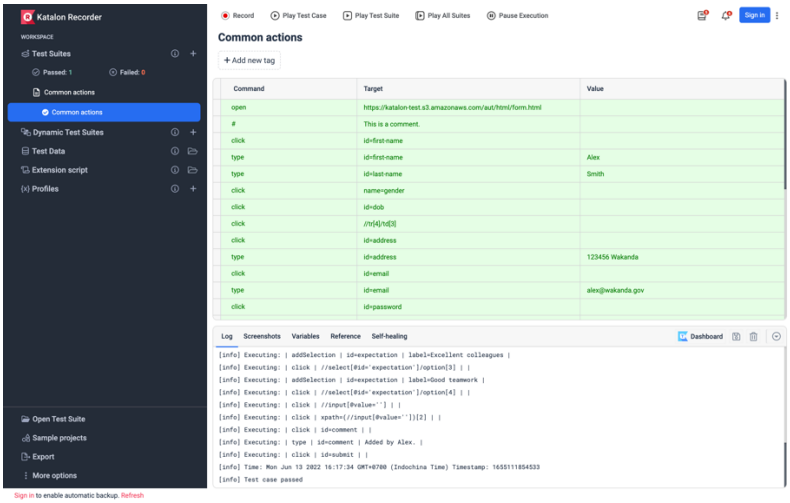


Figura 9. Grabador de flujos en Katalon [26].

3.2.5. Selenium IDE

Figura 10. Extensión de selenium IDE [27].

Selenium IDE se trata de una herramienta para automatizar pruebas que se puede obtener mediante una extensión en nuestros navegadores principales como puedan ser Google Chrome o Mozilla Firefox.

La primera ventaja que encontramos en esta herramienta es que al no tener un propietario privado se trata de un proyecto de colaboración entre varios desarrolladores del mundo por lo que no tendremos que pagar una licencia para poder usarlo. Otra ventaja de sea una comunidad la que mantiene el sistema es más difícil que nos pase como la anterior herramienta y se quede sin soporte.

Además, al ser de código abierto otra de las ventajas que encontramos es que se puede acceder a la herramienta para realizar hacer los ajustes necesarios y solucionar el tema de los identificadores y modificarlo para que se adapte a nuestras necesidades.

Esta herramienta sí que permite tener en un banco de pruebas para poder grabar todos los flujos que se necesiten y así después poder ejecutarlos.

Otra de las características que queríamos evitar era tener que utilizar dos herramientas diferentes, una para todos los flujos de prueba y otro para crear los test de carga y poder hacer las pruebas de rendimiento que era las funciones principales de las anteriores herramientas estudiadas.

Es por ello por lo que Selenium ofrece la función de Selenium Grid que permite también ejecutar los test de carga. Esto lo hace levantando una granja de contenedores (que un contenedor es como un navegador) y cada uno de ellos ejecuta un flujo y así aumentamos el tráfico en el servidor y podemos evaluar el rendimiento de la aplicación.

Que estas dos funciones se encuentren integradas en una misma herramienta nos permite poder usar los mismos scripts que tenemos para realizar las pruebas de funcionalidad que para los test de carga y así evitar tenerlos por duplicado [28].

3.2.6. Comparación de las diferentes herramientas.

Herramientas:	Flood IO	LoadNinja	WebLOAD	Katalon	Selenium IDE
Opensource:	NO	NO	NO	NO	SI
Licencia:	Anual	Anual	Anual	Anual	Gratuito
Test de carga	Si	Si	Si	Si	Si
Test funcionales	No	No	No	Si	Si
Tiene soporte?	Si	Si	SI	No	Si
Prueba:	No funciona	No funciona	No funciona	Funciona	Funciona

Tabla 2. Comparación de las 5 herramientas.

Después de haber comparado las diferentes herramientas podemos observar cómo hemos encontrado una que se ha adaptado a nuestras necesidades. La mayoría de las herramientas que hemos analizado no han sido útiles para nuestra propuesta ya que su función principal era realizar test de carga y no de funcionalidad.

Es por ello por lo que hemos tenido que descartar tres herramientas de forma directa ya que para nosotros era imprescindible que tuvieran la posibilidad de hacer los test de funcionalidad, aunque también valorábamos muy positivamente que se pudiera comprobar el rendimiento con la herramienta elegida.

Solo quedaban dos opciones posibles en la que nos permitían realizar tanto test de carga como de funcionalidad y teniendo en cuenta que una de ellas llegó no hubo más soporte por parte de los desarrolladores, la opción elegida ha sido Selenium IDE que teóricamente cumple con todos los requisitos que pedíamos desde un primer momento.

3.3. Características de la herramienta seleccionada.

Una vez hemos seleccionado la herramienta deseada vamos a conocer a fondo para saber cómo se usa y sacar al máximo su funcionalidad. Primero vamos a ver su pantalla principal y luego saber que funcionalidades tiene cada parte.

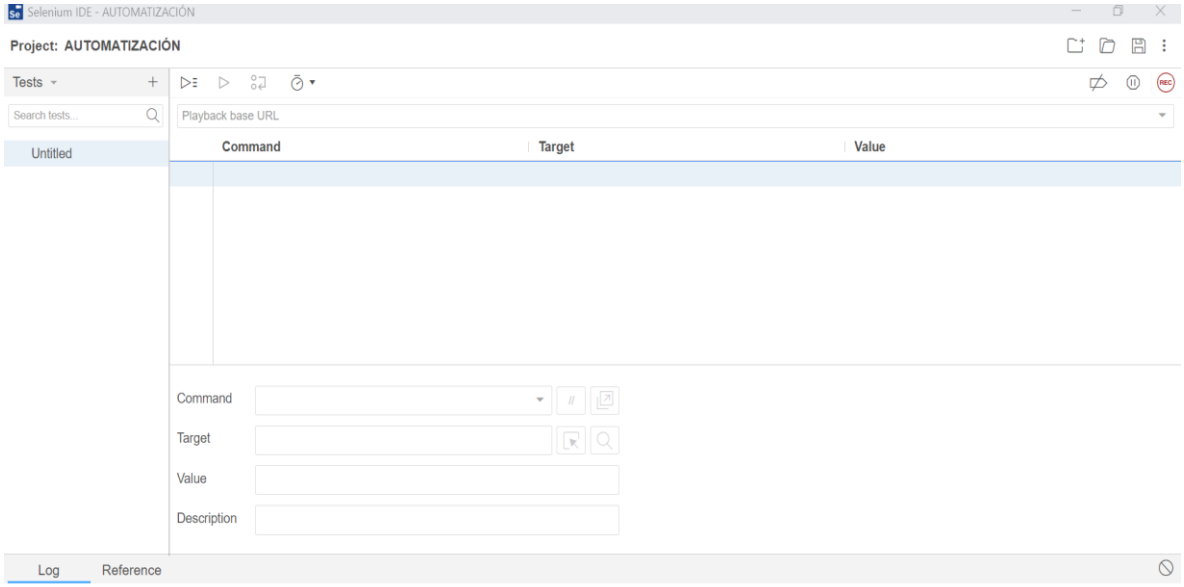


Figura 11. Pantalla principal de Selenium IDE.

La parte de la izquierda de la pantalla se trata de un árbol dónde nos aparecerán los diferentes test que se hayan grabado. Esto será muy útil ya que la idea es separar cada test en diferentes partes para así ser lo más eficiente posible. Por lo tanto al acabar cada caso de prueba en este árbol aparecerán las diferentes partes en las que se haya dividido, como por ejemplo una que sea la creación de un paciente, otra cómo ingresarlo en una sección concreta y así con las que necesitemos. Aquí también habrá una que será la principal, que recibirá el nombre de flujo y que será la que se ejecute, por lo tanto

dentro de esa tendremos que ir llamando por orden a las otras que hayamos creado hasta completar el caso de prueba. En la parte práctica lo veremos de forma más visual.

En la parte inferior se observa la parte del log. Aquí veremos todo lo que se está ejecutando en cada momento. Cuando estamos ejecutando alguno de los flujos que tenemos grabado en el árbol de la izquierda en esta parte nos aparece de forma instantánea el paso que está ejecutando en cada momento. Si hay algún paso que no se haya podido ejecutar ya sea porque no lo haya encontrado la acción que debe realizar o porque ha fallado es en esta parte dónde nos aparecería en que paso ha saltado el error.

En la parte superior nos encontramos diferentes funcionalidades que son:

- Asignar el nombre del caso que estemos automatizando que será con el que se guarde.
- Botón para abrir proyectos ya existentes y el botón para guardar el caso que se está realizando.
- Una casilla para introducir una URL que es dónde se ejecutarán los pasos que hayamos configurado en cada flujo.
- El botón para ejecutar los pasos que están grabados en cada flujo y otro para configurar la velocidad a la que se quiere que se reproduzcan, ya que muchas veces según la velocidad a la que lo tengamos configurado puede crear fallos por que se ejecutan demasiado rápido sin que de tiempo a que las diferentes pantallas vayan cargando.
- Para acabar con la parte superior tenemos el botón para grabar y para pausar la grabación. En el momento se pulse botón de REC se abrirá en el navegador que haya predeterminado la URL que se haya configurado en la casilla para ello y será el momento dónde se empezará a realizar todos los pasos que queramos grabar en esa parte. A medida que los vamos realizando iremos viendo cómo se ponen en la parte central de la ventana que luego explicaremos. Una vez hayamos acabado de grabar la parte del flujo que queramos será tan sencillo como pausar la grabación.

Por último tenemos la ventana de la parte central que es dónde se reflejará lo que hemos grabado. Como hemos comentado anteriormente mientras el botón de grabar está activado y nosotros estamos siguiendo los pasos de las pruebas aquí se nos reflejarán todo lo que hemos realizado.

En la parte de command se nos guardará la acción que hemos realizado, las más comunes serán un click o type si hemos escrito en alguna casilla. En la parte del target se nos guardará el identificador de la acción y en la parte de value el valor que tenga, pero eso solo ocurrirá en las casillas dónde hemos escrito que quedará reflejado aquello que hayamos puesto.

Además desde esta parte tenemos varias funciones adicionales. Una de ellas es cambiar los identificadores en la parte el target para que sea más estáticos y apunten a las librerías que nosotros queramos.

Otra función que podremos realizar será añadir pasos intermedios que no hayan sido grabados y que queramos hacerlo a posteriori. Esto lo podemos realizar de dos formas, la primera de ellas de forma manual que será añadiendo con el botón derecho un paso vacío y escribiendo nosotros que queremos que haga, el paso más común será añadir una pausa. La otra forma de hacerlo será en lugar de escribirlo a mano cuando tengamos el paso vacío será grabar los pasos que queramos y se añadirán automáticamente, para ello deberemos tener abierta la URL en la pantalla exacta dónde queremos añadir los pasos que faltaban.

Por último la última función importante que podremos realizar será ejecutar el flujo empezando por dónde nosotros queramos sin necesidad de tener que ejecutarlo todo. Esto será una gran ventaja si queremos probar una parte muy concreta y el flujo es muy largo. Para ello lo que deberemos hacer será abrir la URL y dejarla por la pantalla del paso anterior al que nosotros queremos ejecutar y una vez lo tengamos situarnos en el paso por dónde queremos empezar y mediante el botón derecho seleccionar la opción de ejecutar desde aquí.

3.4. Descripción de la herramienta a testear.

Enfocando más concretamente en este proyecto, vamos a buscar una aplicación práctica para poder trasladar toda la teoría a un caso de la vida real.

En este caso será para el programa de EHCOS. Un sistema de ámbito hospitalario que empezó siendo dueño la empresa EVERIS, la cual fue luego comprada por la multinacional NTT DATA, la actual propietaria de este.

Actualmente se trata de un programa que se encuentra en pleno rendimiento en algunos lugares por todo el mundo. Entre ellos ahora mismo hay dos proyectos abiertos en España por parte de las comunidades autónomas de Asturias y Comunidad Valenciana. Además fuera del territorio nacional hay dos grandes proyectos también en marcha, como son el sistema hospitalario de Argentina y algunos centros hospitalarios en Chile.

Al tratarse de un entorno hospitalario y como dentro del mismo hay que abarcar muchas áreas y ámbitos de atención, la aplicación está segmentada en diversos productos en la que cada uno tiene una función concreta.

Alguna de esas productos están integradas para que funcione en conjunto con otros, sin embargo hay también otros que están diseñados para que trabajen de forma independiente. Vamos a pasar a mencionar las funciones de cada una de ellas:

- ehCOS Remote: es la parte del sistema que se encarga de gestionar la aplicación que tiene el paciente desde su casa y todas las funciones que pueda hacer desde el mismo. Un ejemplo básico que sería responsabilidad de este producto sería pedir citas desde la aplicación o la función que se está intentando implementar que es la videollamada entre el médico y el paciente desde la propia aplicación.

- ehCOS Triaje: es la parte del sistema que se encarga de gestionar a los pacientes cuando entran en urgencias de un hospital para valorar su nivel de enfermedad para actuar de una forma o de otra. Aunque esta función también se encuentra dentro del producto de otros productos, en este caso al tenerlo de forma independiente es más completo ya que se basa en el Sistema Manchester a la hora de valorar el riesgo de los pacientes. Este producto se puede integrar sin ningún problema con ehCOS Clinic.
- ehCOS SmartICU: es la parte del sistema que se encarga de gestionar a los pacientes más graves que están en el hospital. Un ejemplo es que en este producto se tratan los datos de forma diferente como la toma de constantes del paciente que debe ser más exhaustiva que en otras partes.
- ehCOS Emergency: es el producto que gestiona los servicios de emergencia y permite que estén conectados todos los profesionales para ofrecer un mejor servicio.
- ehCOS Clinical Share: es la herramienta que permite tener conectados los diferentes centros que dispongan de nuestro producto. Esto nos permite poder compartir toda la información de los pacientes para poder tener la información presente en todos los centros.
- ehCOS EMPI: es el producto que se encarga de almacenar todos los datos de los pacientes en un sistema central para que los diferentes centros puedan acceder a toda esa información.
- ehCOS Clinic: es el producto más grande que nos encontramos dentro de ehCOS y permite integrarse con algunos de los productos mencionados anteriormente. Este sistema trata de gestionar toda la historia clínica electrónica del paciente, es decir, toda la parte de atención en los centros hospitalarios o ambulatorios. Dentro de este producto podemos encontrar diferentes áreas en las que se divide todas las posibles atenciones que pueda necesitar el paciente. Estas estarán activas o no dependiendo si en el centro que está en uso disponen de dichos servicios. Algunos ejemplos de áreas que nos podemos encontrar son:
    - Atención en urgencias.
    - Atención en hospitalización.
    - Atención en consultas externas.
    - Servicio de farmacia.
    - Quirófano.
    - Cuentas del paciente.
    - RIS: Pruebas de Imágenes

A continuación podemos ver un resumen de los productos de ehCOS y dónde se encuentran:

Figura 12. Productos de ehCOS.

Figura 13. Lugares de implementación de los productos de ehCOS.

En este caso se aplicará la automatización de pruebas a la parte de ehCOS Clinic, ya que al ser el producto más grande y ser el que más áreas tiene dentro de él, debemos de probar todos los casos posibles para evitar al máximos los posibles fallos. Aunque la idea

de este proyecto es ampliar esas pruebas automatizadas a la mayoría de los productos, siempre que se pueda y sea necesario

4. Desarrollo

En este capítulo se va a describir cómo y dónde se han diseñado las pruebas y un ejemplo de cómo se ha implementado una de las pruebas en la herramienta seleccionada.

4.1. Diseño del sistema de pruebas.

Como hemos comentado anteriormente, el ecosistema de ehCOS es una herramienta de uso muy grande que cuenta con más de 2.500 casos de prueba, es necesario contar con un banco de pruebas tan amplio para poder probar todas las casuísticas.

En aras de mantener el sistema de pruebas útil y ordenado, las pruebas están desarrolladas en Qmetry. Qmetry se trata de una plataforma de gestión de pruebas de software y calidad de aplicaciones. En nuestro caso está integrado dentro de la herramienta JIRA, que es una herramienta de gestión de proyectos y de seguimiento de problemas de la empresa Atlassian. Además también se usará otra herramienta llamada Jenkins dedicada también a la gestión de proyectos.

Esto nos permite tener integrado en un mismo lugar los resultados de las pruebas que se van realizado enlazándolo de las incidencias que surgen de los mismos. Además entre otras cosas también nos permite crear tareas de todas las pantallas que existen en ehcos y poder enlazarlas con los casos de prueba para saber que casos ejecutar en caso de que haya habido modificaciones en alguna pantalla. Tener ambas herramientas integradas nos permite conseguir una herramienta muy útil para el proyecto en su conjunto.

Dentro de Qmetry podríamos destacar que la organización del banco de pruebas se divide en dos ventanas: Test Cases y Test Cycle.

- Test Cases: es la parte dónde se encuentran todas las pruebas del sistema. Están ordenadas por carpetas y sub carpetas dependiendo de la parte del sistema al que pertenezcan. Así se pueden tener todas las pruebas clasificadas y ordenadas para que sea más sencillo encontrarlas.

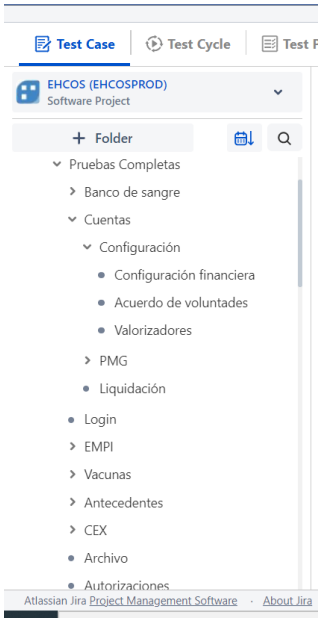


Figura 14. Árbol de las pruebas en Qmetry.

- Test Cycle: es la parte dónde se asignan las pruebas a la persona responsable para que pueda probarlas. Aquí nos encontramos carpetas con los diferentes evolutivos que se están probando y dentro las diferentes partes que hay que probar, indicando en algunas el entorno dónde hay que ejecutarlas. Así se tiene un control ordenado de que pruebas se ha probado en cada evolutivo, en que entorno se ha hecho, cuando se han ejecutado y quien ha sido el responsable de ello.

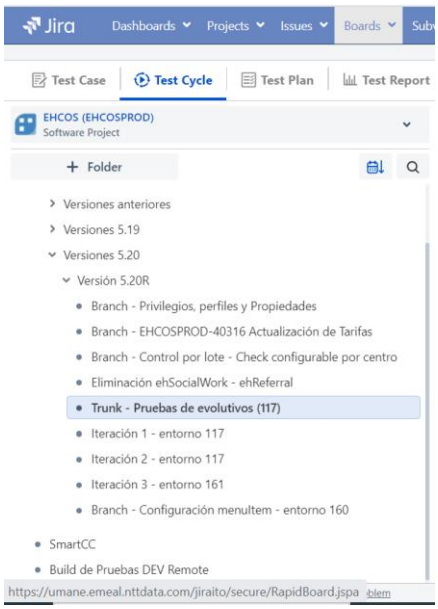


Figura 15. Árbol dónde se asignan las pruebas para probar.

Una vez tenemos nuestro banco de pruebas organizado para que sea fácil su acceso vamos a proceder a lo más importante, que será diseñar las pruebas en sí. Para ello lo primero que tener en cuenta es que a la hora de diseñar los casos de prueba hay dos cosas que son fundamentales.

La primera de ellas es la forma de escribir las pruebas. Las pruebas que vamos a hacer no van a ser usadas solo una vez, sino que la idea es que eso se quede guardado para que en futuras versiones también se ejecute. Además no solo vamos a ser nosotros la persona que tenga que hacer esa tarea, ya sea bien porque en un futuro tenemos otras funciones dentro del proyecto o bien porque ya no formamos parte de él.

Es importante tener esto en cuenta a la hora de escribir las pruebas, ya que deben estar muy bien detalladas y además tienen que estar separadas por pasos para que sea más fácil interpretarlas. Esto nos permitirá que cualquier persona pueda entenderlas y ejecutarlas sin necesidad de consultar a la persona responsable que la creo.

La segunda cosa fundamental a la hora de crear las pruebas es pensar en que queremos probar. Para que un banco de pruebas tenga una función importante y sea útil no solo tenemos que pensar en las cosas que se usan en el día a día. La diferencia entre unas pruebas bien hechas y unas que no lo están tanto, son esos casos un poco más raros que no se suelen realizar tanto en el día a día pero que muchas veces son esos casos que no se han contemplado a la hora de hacer en el desarrollo y son los que generan los errores y nosotros debemos detectar en las pruebas antes de que el programa pase a producción.

Vamos a proceder ahora mostrar un ejemplo de cómo se han desarrollado las pruebas de forma práctica:

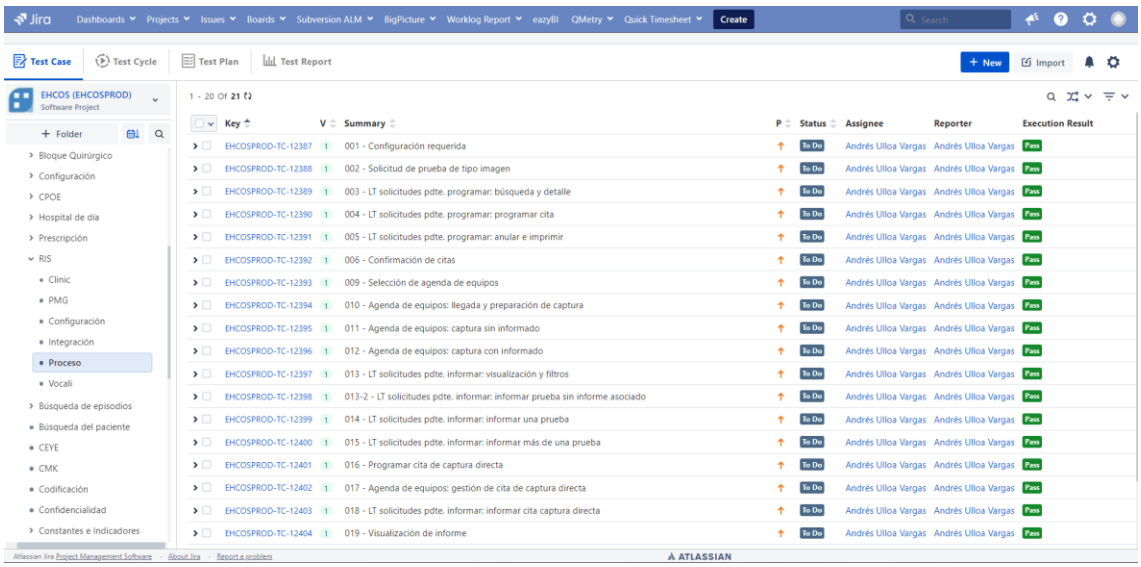


Figura 16. Listado casos de prueba en una carpeta.

En esta captura de pantalla podemos observar unas de las carpetas comentadas anteriormente para organizar los casos de prueba. En este caso se trata de la carpeta del RIS, en concreto la parte de proceso como nos indica el árbol de la izquierda.

Podemos ver cómo están los 21 casos correspondientes ordenados en una tabla donde hay diferentes columnas para las diferentes acciones que podemos realizar en esos casos:

- La primera columna es una flecha que nos permite desplegar los casos de prueba junto a un check cuya función es simplemente para poder realizar cualquier acción seleccionando dicho check.

- La segunda columna se trata del identificador del caso de prueba, es un código que podemos copiar y pegar en el buscador del JIRA para poder acceder a él sin necesidad de ir buscando en que carpeta se encuentra. Esto también nos sirve para enviarle el caso a otro compañero del equipo ya que se trata de un link.

- La tercera columna se trata de la versión del caso de prueba, si ha habido cambios en él y hay varias versiones reflejará cual está asignada a la carpeta.

- La cuarta columna es el summary o también llamado resumen. La función que nosotros le damos es el título del caso de prueba para con solo leerlo saber que se va a probar en él. También se suele numerar para así poder ordenarlos en el orden de ejecución adecuado.
- La quinta columna se trata de la prioridad que necesitamos ejecutar el caso, ya que hay algunos que son más prioritarios que otros. En nuestro caso podemos elegir entre las opciones: bajo, medio, alto y bloqueado. Se representa en la tabla mediante símbolos para que sea más visual.

- La sexta columna es el estado que se encuentra el caso de prueba. Puede ser que se encuentre ya ejecutado, que este en progreso, pausado, aún por hacer o caducado.

- La séptima columna es a quien se le ha asignado el caso de prueba y será el responsable de ejecución.

- La octava columna es quien ha creado el caso de prueba.

- Y por último la novena columna es como ha sido la ejecución del caso de prueba. Tenemos la opción de no ejecutado, pasado, bloqueado o fallado.

Ahora vamos a ver un caso concreto de un caso de prueba:

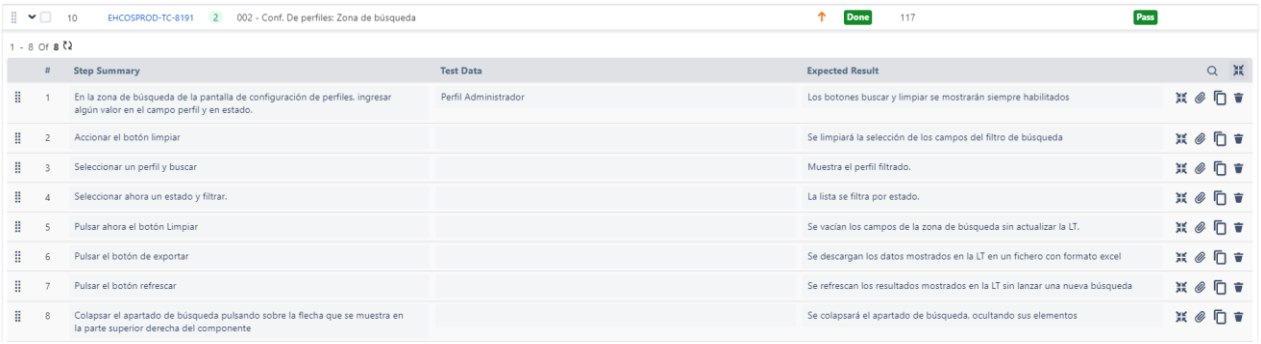


Figura 17. Definición de un caso de prueba.

Podemos observar como el caso también se divide en una tabla dónde nos podemos encontrar los siguientes apartados.

En primer lugar nos encontramos con el step summary, que son los pasos que la persona debe de seguir para realizar el caso de prueba. Es importante que los pasos estén numerados para que todo funcione correctamente. Además a la hora de escribir los pasos es que estén muy bien redactados y no falte nada. Es importante que la persona que haya hecho el caso no de nada por supuesto y por pequeño o muy obvio que sea algún detalle como cerrar un pop up o alguna acción similar quede constancia en los pasos para que después a la hora de ejecutarlos no haya dudas posibles.

En segundo lugar nos encontramos con el Test Data, que son los datos que necesitamos para la prueba, es decir, son aquellas condiciones que debemos tener en cuenta para ejecutarlo. En la mayoría de los casos solo se encuentra información en el primer paso, ya que es la precondición que hay que tener en cuenta antes de ejecutarlo. Muchas veces esto se trata de saber las propiedades y privilegios que debemos de tener activados o asignados al profesional y con que profesional debemos ejecutar el caso. Esto no quiere decir que no haya situaciones dónde tengamos información del Test Data en algún paso intermedio ya que puede ser que a mitad del caso tengamos que cambiar el profesional o bien cambiar algún dato de configuración, por lo que esta columna sería el sitio adecuado para reflejarlo.

La columna del Expected Result refleja el resultado que nos debe de salir al realizar cada paso. Aquí también debemos poner cada pequeño cambio que se tenga que producir al realizar cada paso correspondiente. Esto se debe a que muchas veces al realizar un paso tiene varias cambios o consecuencias y si no quedan todas reflejadas en dicha columna la persona que está probando los casos puede pensar que el sistema funciona bien, pero en realidad falta por aparecer un elemento importante en la pantalla pero como no está reflejado el caso no lo pondrían en estado fallado cuando ese sería su estado real.

Asimismo, hay varios elementos en la pantalla dónde se encuentran los casos de prueba que nos facilitan crearlos, como pueda ser el de ordenar los pasos, eliminar algún paso que ya no necesitemos o copiar algún paso para volver a copiarlo y hacer las pequeñas modificaciones y así no tener que escribir otra vez el caso entero.

4.2. Implementación del sistema de pruebas y definición del caso práctico.

Para poder explicar de forma detallada como se ha implementado todo lo visto anteriormente nos centraremos en un caso de prueba dónde se realicen varias acciones dentro del mismo. Para ello primero explicaremos los pasos que debemos seguir para poder grabar y explicaremos como ha sido el proceso de forma más gráfica para que sea más sencillo entenderlo.

Tendremos que recordar que a la hora de empezar a grabar un flujo hay que entender que para que sea más sencillo poder hacer esto de forma más eficiente y poder reutilizar pasos los grabaremos por partes.

Por lo tanto el primer paso que deberemos hacer será grabar esos subflujos. En el árbol de la izquierda crearemos un nuevo subflujo dónde la asignaremos el nombre de la acción que vayamos a realizar dentro de él. Una vez lo tengamos creado pulsaremos el botón de grabar y seguiremos los pasos de esa parte como si se tratara de un caso de prueba manual, cuando acabemos dejaremos de grabar y tendremos todos los pasos que hayamos hecho dentro de esa parte.

Este paso lo tendremos que repetir tantas veces como en partes hayamos dividido el caso de prueba, para al final de este paso conseguir tener el caso de prueba dividido en distintos subflujos en el árbol de la izquierda. Otra cosa que hay que tener en cuenta es que podemos tener varios subflujos grabados y guardados en árbol pero si luego no los usamos en el flujo principal no se ejecutarán, por lo que no es obligatorio tener solo los que se van a utilizar.

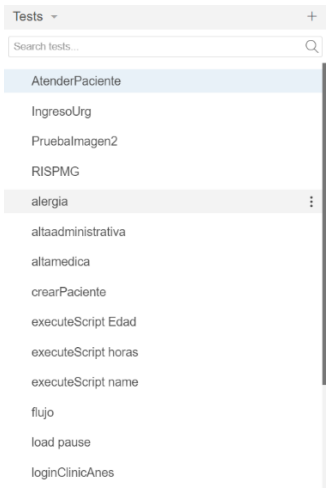


Figura 18. Árbol de Selenium ID con los flujos grabados (parte 1).

Figura 19. Árbol de Selenium ID con los flujos grabados (parte 2).

El segundo paso que deberemos realizar será abordar el problema de los identificadores variables que teníamos. Para ello lo que tendremos que hacer será entrar en cada subflujo que hayamos creado e ir seleccionando uno a uno todos los pasos que haya creados y cambiarle el template. En algunos nos aparecerán varias opciones pero nosotros deberemos escoger siempre que se pueda la que lleve el identificador "name" que será el que apunte a nuestra librería y será más estático que el que aparece por defecto.

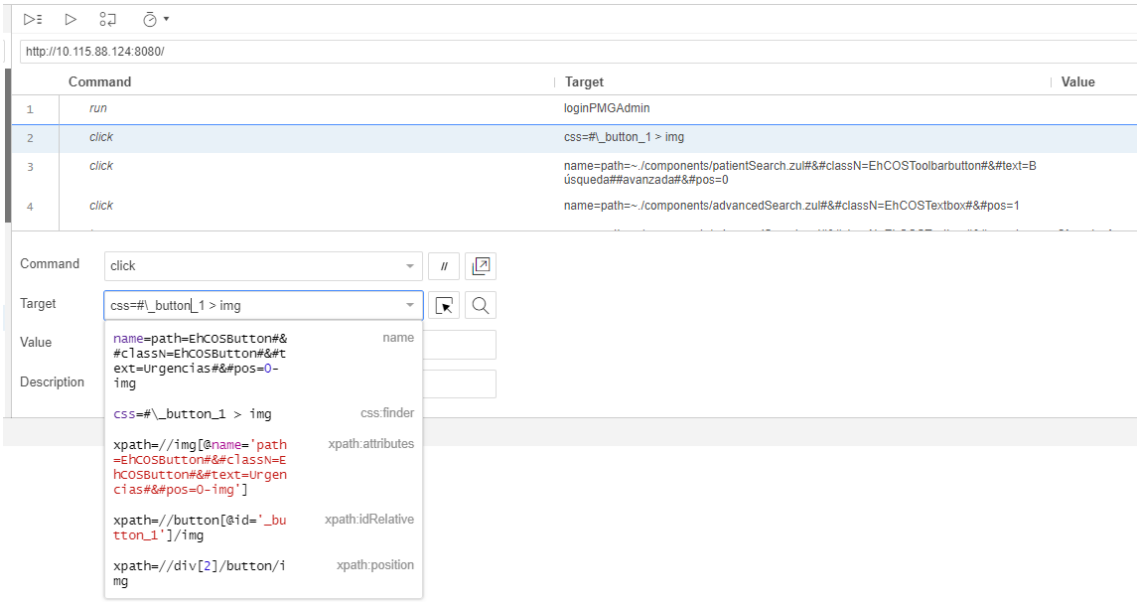


Figura 20. Cambio de identificador en Selenium.

La anterior figura se trata del segundo paso en el que pulsamos el icono de urgencias para crear un paciente, es un paso muy común que se usa en varios subflujos. Como podemos observar el identificador que se asigna automáticamente no lleva el identificador name por lo que será variable y si lo ejecutamos varias veces es posible que no lo encuentre, por lo tanto seleccionaremos en el target la primera opción dónde sí que está el identificador name y además nos aporta más información y podemos distinguir que hace ese paso en concreto.

Teóricamente después de realizar los dos pasos anteriores solo quedaría juntar los diferentes subflujos en el flujo principal y ejecutarlos para acabar el caso de prueba pero en la realidad no es así.

El paso que toca ahora es probar los diferentes subflujos con los identificadores ya bien puestos para ver si funcionan, aunque si hemos usado un poquito el programa para probarlo con anterioridad vamos a saber que no va a funcionar.

El problema que se nos plantea ahora son los tiempos de espera. Cuando nosotros realizamos el caso de prueba de forma manual los tiempos de espera dependen de nosotros mismos, es decir, si nosotros pulsamos un botón y sabemos que se nos tiene que abrir una nueva ventana o se nos tiene que habilitar un campo somos conscientes que no vamos a poder realizar el siguiente paso hasta que esa acción no pase, y como lo vemos de forma visual nos esperamos el tiempo que sea necesario. Es verdad que el programa trae una función para poder regular la velocidad a la que se ejecutan los pasos y por lo tanto desde ahí podíamos configurarlo pero no sería nada eficiente ya que nos tendríamos que guiar por el paso más lento y ralentizar una media de 300 pasos que pueda tener cada grabación de caso de prueba no es eficiente ya que el tiempo se incrementaría en más de 5 veces lo que debería tardar en ejecutarse.

Por eso el tercer paso que deberemos hacer será revisar los pasos grabados y dónde podamos detectar que el paso siguiente depende del anterior es decir, que se tiene que activar algo, abrir una nueva ventana o guardar algún documento, entre esos pasos colocaremos una pausa o ejecutaremos un subflujo que hayamos creado previamente que también será una pausa.

Esto no será del todo efectivo ya que después de eso tendremos que ejecutar por subflujos todos los que tengamos y ver en que partes va más rápido de lo que debería ir y no le da tiempo al programa a realizar las acciones pertinentes y colocaremos las pausas oportunas.

El último paso que deberemos realizar será colocar por orden en el flujo principal las diferentes partes que en las que tengamos dividido el caso de prueba en forma de subflujos y probarlo para ver que todo funcione y ajustar algún paso en el caso que sea necesario.

Ahora vamos a pasar a visualizar de forma práctica uno de los casos que se ha automatizado. Se trata de un caso de prueba de la parte de RIS.

EHCOSPROD-TC-12388 1 002 - Solicitud de prueba de tipo imagen — To Do — Andrés Ulloa Vargas — Andrés Ulloa Vargas — Pass

1 - 4 Of 4

#	Step Summary	Test Data	Expected Result
1	Acceder a urgencias como médico, atender un paciente y abrir la pantalla del gestor de solicitudes. Seleccionar en tipo de prueba Diagnóstico por imagen. Seleccionar una de las pruebas y completar los campos mínimos (tipo de prueba, prueba, prioridad, motivo o propósito de la prueba, sospecha/diagnóstico). Añadir al carro y solicitar.		Al seleccionar el tipo de prueba se muestran los campos de contraste, tipo, lateralidad y requiere anestesia en la segunda fila. Se crea la solicitud en estado Pendiente programar. Se añade a la lista inferior.
2	Registrar al paciente activo una alergia a uno de los materiales de contraste configurados.		Se crea la alergia y se observa la alerta roja en la barra de identificación.
3	Desde el gestor de solicitudes, pedir una nueva prueba de tipo imagen, e indicar que sí se requiere contraste, pero no indicar cuál. Añadir la prueba al carro. Seleccionar otra prueba de imagen e indicarle el contraste al que el paciente es alérgico. Añadir la prueba al carro. Aceptar el cambio de contraste, indicar que no requiere. Añadir al carro. Solicitar las pruebas.		Con la primera prueba se muestra un mensaje de aviso indicando que el paciente tiene alergias registradas a contrastes y da la opción de modificar la solicitud. Con la segunda prueba se muestra un mensaje indicando que el paciente tiene alergias registradas a ese contraste, y da la opción de modificar. Al eliminar el contraste ya no muestra aviso y añade la prueba al carro. Se solicitan ambas pruebas correctamente y se crean en estado Pendiente programar.
4	Desde la lista de solicitudes del episodio, ver el detalle de cada prueba y verificar que los datos cargados son correctos.		Se verifica que el detalle se visualiza correctamente, de las tres pruebas solicitadas.

Figura 21. Caso de prueba de RIS que se va a automatizar.

Una vez hemos entendido el caso de prueba tendremos que analizar cómo podemos dividirlos.

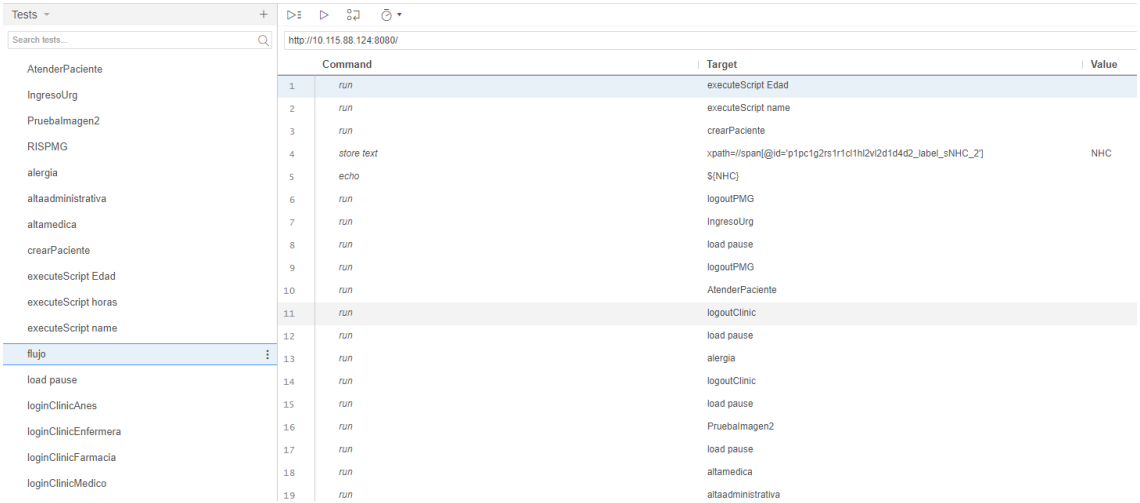
El primer paso que es solicitar una prueba al paciente tendremos que dividirlo en: crear al paciente, ingresar al paciente y por último atenderlo para solicitarle la prueba.

El segundo paso que hay que registrarle una alergia se podrá hacer todo en el mismo subflujo.

El tercer paso y el cuarto se podrán hacer juntos en el mismo subflujo.

Por último tendremos que añadir también darle el alta médica al paciente y luego el alta administrativa para que se queda completado y cerrado el episodio del paciente.

Por lo tanto el flujo principal quedará de esta forma:



Tests

Search tests...

AtenderPaciente
IngresoUrg
PruebaImagen2
RISPMG
alergia
altaadministrativa
altamedica
crearPaciente
executeScript Edad
executeScript horas
executeScript name
flujo
load pause
loginClinicAnes
loginClinicEnfermera
loginClinicFarmacia
loginClinicMedico

http://10.115.88.124:8080/

	Command	Target	Value
1	run	executeScript Edad	
2	run	executeScript name	
3	run	crearPaciente	
4	store text	xpath=//span[@id='p1pc1g2rs1r1cl1hl2vl2d1d4d2_label_sNHC_2']	NHC
5	echo	${NHC}	
6	run	logoutPMG	
7	run	IngresoUrg	
8	run	load pause	
9	run	logoutPMG	
10	run	AtenderPaciente	
11	run	logoutClinic	
12	run	load pause	
13	run	alergia	
14	run	logoutClinic	
15	run	load pause	
16	run	PruebaImagen2	
17	run	load pause	
18	run	altamedica	
19	run	altaadministrativa	

Figura 22. Flujo principal del caso de prueba.

Vamos a explicar la función de cada paso:

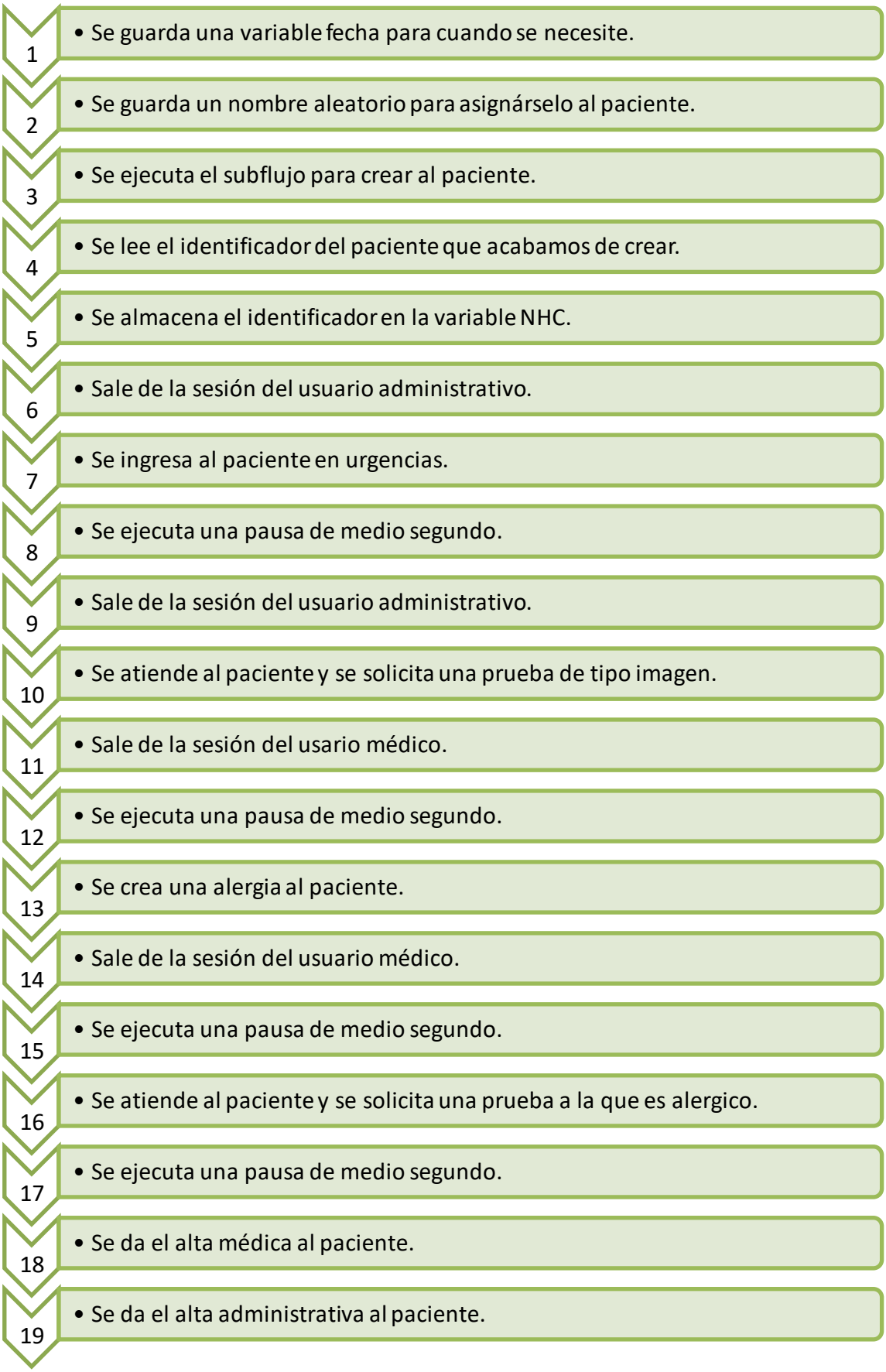


Figura 23. Esquema de los pasos grabados.

Una vez hemos visto el flujo principal que hace, vamos a ver como ejemplo los pasos de uno de los subflujos, en este caso hemos elegido el de crear al paciente que es el primero que ejecutamos en el sistema.

	Command	Target	Value
1	*run*	loginPMGAdmin	
2	*click*	name=path=EhCOSButton#&#classN=EhCOSButton#&#text=Urgencias#&#pos=0-img	
3	*click*	name=path=~./components/patientSearch.zul#&#classN=EhCOSToolbarbutton#&#text=Búsqueda##avanzada#&#pos=0	
4	*click*	name=path=~./components/advancedSearch.zul#&#classN=EhCOSTextbox#&#pos=1	
5	*type*	name=path=~./components/advancedSearch.zul#&#classN=EhCOSTextbox#&#pos=1	${nombre}
6	*click*	xpath=//i[@name='path=~./components/advancedSearch.zul#&#classN=EhCOSCombobox#&#pos=0-btnCmb']	
7	*click*	xpath=//td[@name='path=~./components/advancedSearch.zul#&#classN=EhCOSCombobox#&#pos=0#&#text=Hombre-div']	
8	*click*	name=path=~./components/advancedSearch.zul#&#classN=EhCOSButton#&#text=Buscar#&#pos=0	
9	*click*	name=path=Lista##de##pacientes#&#classN=EhCOSButton#&#text=Nuevo##paciente#&#pos=0	
10	*click*	name=path=Nuevo##paciente#&#classN=EhCOSTextbox#&#pos=2	
11	*type*	name=path=Nuevo##paciente#&#classN=EhCOSTextbox#&#pos=2	${nombre}
12	*click*	name=path=Nuevo##paciente#&#classN=EhCOSTextbox#&#pos=3	
13	*type*	name=path=Nuevo##paciente#&#classN=EhCOSTextbox#&#pos=3	${nombre}
14	*click*	name=path=Nuevo##paciente#&#classN=EhCOSDatebox#&#pos=0-input	
15	*type*	name=path=Nuevo##paciente#&#classN=EhCOSDatebox#&#pos=0-input	${dateEdad}\n

Figura 24. Parte 1 de los pasos para la creación del paciente.

17	*store attribute*	name=path=Nuevo##paciente#&#classN=EhCOSDatebox#&#pos=0-input@serverdate	dateString
18	*run*	executeScript horas	
19	*click*	xpath=//i[@name='path=Nuevo##paciente#&#classN=EhCOSLocator#&#pos=1-btnLoc']	
20	*click*	xpath=//div[@name='path=Nuevo##paciente#&#classN=ClinicListbox#&#pos=2#&#classN=Listcell#&#text=Desconocido##-##DESC#&#pos=0-div']	
21	*click*	name=path=Nuevo##paciente#&#classN=EhCOSButton#&#text=Guardar#&#pos=2	
22	*click*	xpath=//span[@name='path=Nuevo##paciente#&#classN=Tab#&#pos=3#&#label=Datos del seguro-span']	
23	*click*	name=path=Nuevo##paciente#&#classN=EhCOSButton#&#text=Nuevo#&#pos=1	
24	*click*	name=path=Nuevo##paciente#&#classN=EhCOSCheck#&#text=Seguro##Principal#&#pos=0-input	
25	*click*	xpath=//i[@name='path=Nuevo##paciente#&#classN=EhCOSLocator#&#pos=14-btnLoc']	
26	*click*	xpath=//div[@name='path=Nuevo##paciente#&#classN=ClinicListbox#&#pos=19#&#classN=Listcell#&#text=ADESLAS##-##ADESLAS#&#pos=0-div']	
27	*click*	name=path=Nuevo##paciente#&#classN=EhCOSTextbox#&#pos=35	
28	*type*	name=path=Nuevo##paciente#&#classN=EhCOSTextbox#&#pos=35	123
29	*click*	name=path=Nuevo##paciente#&#classN=EhCOSDatebox#&#pos=5-input	
30	*type*	name=path=Nuevo##paciente#&#classN=EhCOSDatebox#&#pos=5-input	${dateEdad}\n
31	*click*	name=path=Nuevo##paciente#&#classN=EhCOSButton#&#text=Guardar#&#pos=1	
32	*click*	name=path=Nuevo##paciente#&#classN=EhCOSButton#&#text=Guardar#&#pos=2	

Figura 25. Parte 2 de los pasos para la creación del paciente.

Vamos a pasar a explicar un poquito a grandes rasgos de que se encarga cada paso.

En los pasos del 1 al 8 lo que hacemos es ingresar al sistema como administrativo y buscar si tenemos un paciente con el nombre aleatorio que hemos creado.

En los pasos del 9 al 15 como no tenemos ningún paciente con esas características creamos un nuevo paciente y le asignamos el nombre y apellidos aleatorio que hemos creado, que será igual en los tres campos. Además también le asignamos una fecha de nacimiento que previamente también hemos creado en otro subflujo.

En los pasos del 17 al 21 lo que hacemos es ejecutar un subflujo para que guarde la fecha de hoy que nos servirá más adelante, además de completar el resto de los datos básicos que faltan de la ficha del paciente como pueda ser de dónde es y lo guardamos.

En los pasos del 22 al 32 completamos la parte de la ficha del paciente correspondiente al segundo, dónde introducimos su compañía de seguro, el plan que tiene contratado y la fecha vigente. Luego de esto guardamos y ya hemos completado la creación de un paciente en el sistema

5. Resultados

En este capítulo se van a describir como se han obtenido los resultados de las pruebas para poder analizarlos y obtener unas conclusiones fiables.

5.1. Sistema de pruebas.

Cuando ya tenemos el flujo grabado y hemos hecho los pequeños ajustes oportunos en cuanto a pausas vamos a obtener la información de cómo funciona y los errores que está detectando selenium.

Cuando nosotros empecemos la prueba en el log de la parte de debajo de selenium podremos ir viendo en tiempo real paso por paso el que se está ejecutando en cada momento y a qué hora.

Si un paso se ejecuta de forma correcta nos aparecerá un check verde a su lado en la ventana principal y un “OK” en verde en la ventana del log y pasará al siguiente paso.

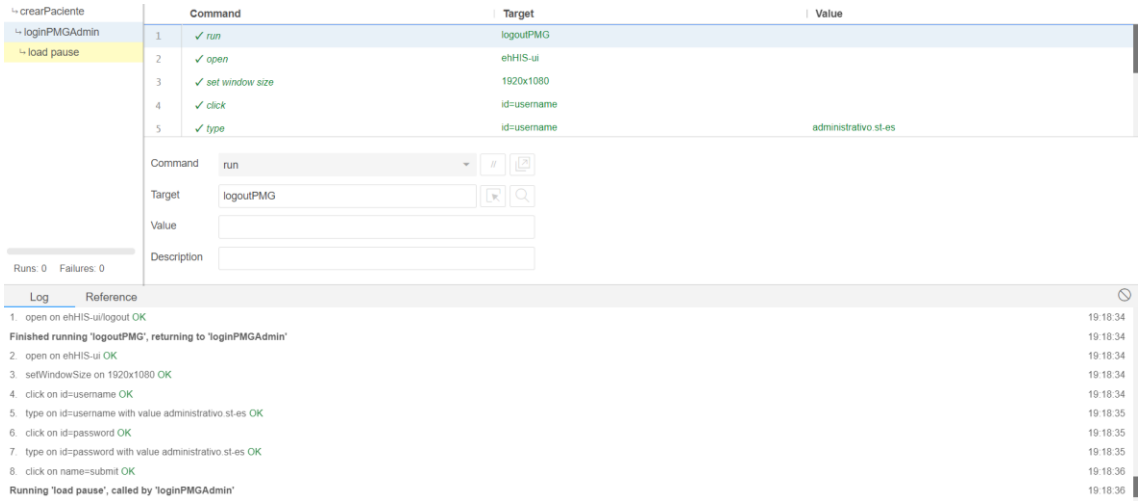


Figura 26. Pantalla principal de Selenium con ejecución correcta.

Si un paso no se ha ejecutado de forma correcta primero se quedará pensando unos segundos para ver si encuentra la acción que debe realizar y una vez pasado ese tiempo de margen nos aparecerá una cruz roja en el paso dónde se haya detenido en la ventana principal y marcará el paso como “Failed” en la ventana del log inferior junto a un pequeño mensaje sobre el error.

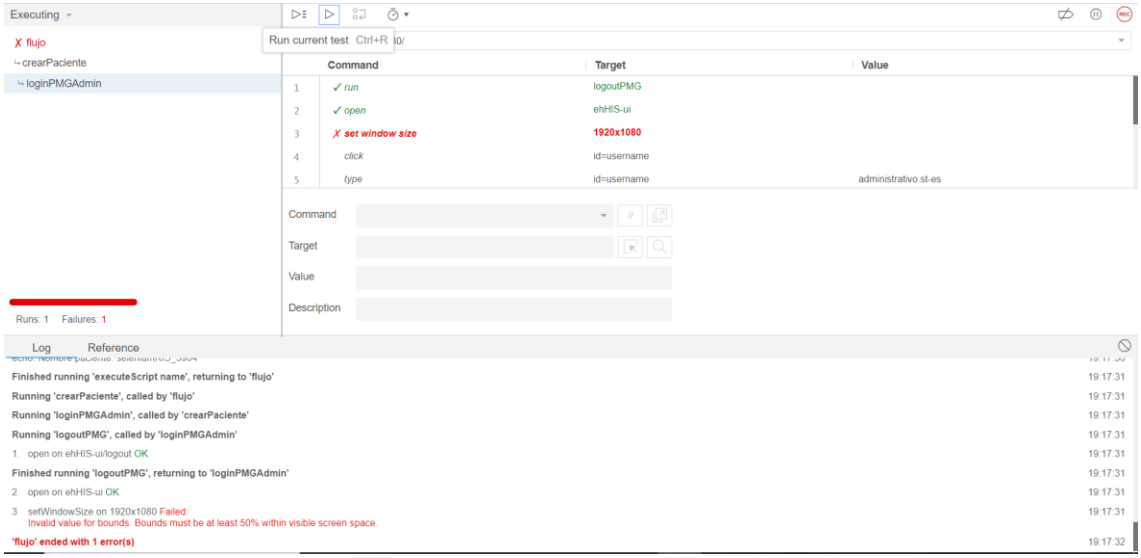


Figura 27. Pantalla principal de Selenium con fallo en la ejecución.

A continuación se va a hacer una muestra en formato de video para ver como el programa al ejecutar el caso de prueba hace los pasos de forma automática, se va a mostrar solo la primera parte del caso de prueba:

https://youtu.be/8Rcqznv3zys

5.2. Análisis de datos.

Una vez estamos seguro de que una o más pruebas de nuestro testbench funciona de manera manual, es decir, activando nosotros los pasos y viendo como los ejecuta la herramienta de pruebas sin ningún error, y repitiendo este paso varias veces para saber que es una prueba estable y no funciona una vez solo de casualidad procederemos a pasárselo a nuestro equipo de arquitectura.

Este equipo será en encargado de ponerlo en marcha dentro del servidor, para que todas las noches las pruebas que ya están acabadas se vayan ejecutando automáticamente y nos vayan reportando los resultados a nosotros para que podamos analizar cómo están funcionando y detectar los fallos en el caso de que los haya.

Para ello todas las mañanas nos llegará un correo a las personas encargadas de automatizas las pruebas dentro del equipo de QA para que podamos analizar cómo han ido las ejecuciones esta noche y si debemos modificar alguna cosa para la siguiente ejecución que será esa misma noche.

El correo que nos llegará tendrá dos partes, la primera de ellas será una tabla dónde podamos ver reflejados los casos que se han ejecutado, los pasos que se han podido llegar a ejecutar en dicho caso (ya que si hubiera algún error no se ejecutarían todos), la

hora de inicio y la hora de final a la que se ha ejecutado cada caso y por último el estado, es decir, si ha pasado las pruebas o ha habido algún error en cuyo caso nos lo indicará.

Ejecucion pruebas QA 23/05/2023

Resumen ejecucion pruebas

Nombre flujo	Numero pasos ejecutados	Hora Inicio	Hora Fin	Estado	Error
Flujo_Hospitalizacion	228	23/05/2023 01:58:14	23/05/2023 02:03:30	✅	
Flujo_CEX	34	23/05/2023 02:03:30	23/05/2023 02:04:33	❌	Flujo error Flujo_CEX
Flujo_Urgencias	257	23/05/2023 02:04:33	23/05/2023 02:10:33	✅	
Flujo_BloqueQuirur	128	23/05/2023 02:10:33	23/05/2023 02:15:29	❌	Flujo error Flujo_BloqueQuirur
Flujo_RIS	219	23/05/2023 02:15:29	23/05/2023 02:20:15	✅	
Flujo_Obstetricia	359	23/05/2023 02:20:15	23/05/2023 02:27:11	✅	
Flujo_Farmacia	108	23/05/2023 02:27:11	23/05/2023 02:29:58	✅	

Figura 28. Tabla de resultados pruebas automatizadas.

La segunda parte del correo será referente a los errores que se han producido al ejecutar las pruebas dónde encontraremos a que flujo pertenece el error, el paso de selenium en el que se ha producido que al cambiarle el identificador podemos identificar de forma bastante precisa en que parte del flujo se ha producido y por si eso no fuera suficiente una captura de pantalla por dónde se ha quedado.

**Se adjuntan logs asociado a las ejecuciones y capturas de los errores*

Adjunto error Flujo_CEX: Inicio
Error paso: By.xpath: //div[@name='path=Programación##de##citas#&#title=Nueva##búsqueda#&#classN=ClinicListbox#&#pos=5#&#classN=Listcell#&#text=Agenda##de##consulta#&#pos=0-div']:

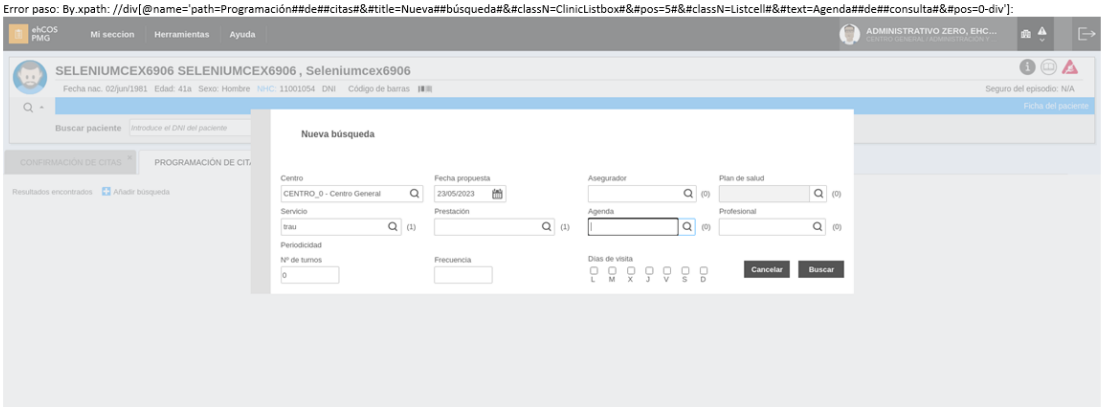


Figura 29. Captura del error del flujo pruebas básicas de CEX.

Al tener toda esa información solo con leer el correo cada mañana podemos detectar los errores de una forma muy sencilla y solucionarlos en poco tiempo.

Como ejemplo podemos ver un error de un flujo básico que se ha ejecutado en la parte de CEX y podemos detectar que el flujo no ha podido completarse por que en el sistema no quedaban huecos disponibles en la agenda para programar una cita por lo tanto la herramienta ha podido crear al paciente de una forma normal pero al intentar programar la cita se ha encontrado que la agenda que tenía seleccionada estaba llena por lo que no ha podido continuar y como la automatización no tiene la capacidad como una persona que en una prueba manual hubiera buscado otra agenda ha saltado un error. Al día siguiente se solucionó dicho error con una cosa tan sencilla como cambiarle la agenda que tenía definida y crear una solo para automatización con el máximo número de citas posibles por agenda para intentar prevenir más errores similares a este.

6. Conclusiones y líneas futuras.

En este capítulo vamos a presentar las conclusiones más relevantes derivadas de nuestra investigación en nuestro trabajo de final de grado. En este estudio hemos analizado todos los aspectos sobre la automatización de pruebas para software de ámbito sanitario con el objetivo de conseguir la herramienta que mejor se adapte a nuestras necesidades para así poder dotar a nuestro proyecto de una mayor eficiencia en el trabajo y por ende conseguir una reducción de costes en el mismo. En este estudio se han obtenido una serie de resultados y hallazgos referentes a este tema y se va a exponer las conclusiones más relevantes que se han obtenido en dicha investigación.

La primera de las conclusiones a las que se ha llegado es que en el mercado existen multitud de herramientas disponibles para realizar test de carga en aplicaciones web pero no hay tantas disponibles para poder realizar test funcionales y más que se adapten a las características específicas para nuestro proyecto. Pero a pesar de todo esto sí que hemos sido capaces de encontrar la herramienta que se adapta a lo que nosotros buscábamos y después de realizar los test oportunos podemos llegar a la conclusión que sí que hemos llegado al objetivo esperado, que desde un primer momento era encontrar un programa que funcionara y ponerlo en funcionamiento.

Después de realizar varias pruebas para comprobar que ya son estables y que funcionen siempre para que se ejecuten todas las noches también podemos hablar del tiempo que nos ha llevado automatizar la prueba. En un primer momento no pensábamos que nos llevaría tanto tiempo ya que muchas veces no llegamos a percibir las pequeñas decisiones que las personas tomamos a la hora de realizarlas de forma manual y que hay que tener en cuenta para adaptarlas a esta nueva forma de trabajo. A pesar del tiempo invertido en configurar todas las esperas entre pasos que ha sido lo que más tiempo nos ha llevado para que no surgieran errores, la automatización supone una gran carga de trabajo al principio pero luego ese tiempo se rentabilizará a la hora de sacar nuevas versiones y en esos momento no tengamos que volver a hacer todas las pruebas de forma manual.

Es por ello por lo que podemos decir que la automatización de pruebas en nuestro proyecto ha supuesto un ahorro de tiempo, por lo tanto también lo podemos relacionar con el ahorro económico que esto supone a la compañía y esto será un beneficio a la hora de poder tener precios más competitivos para abarcar nuevos mercados.

Además otra de las conclusiones que sacamos al adentrarnos en esta nueva tarea ha sido cambiar la metodología de trabajo. En un principio se empezó a automatizar los casos de prueba que ya teníamos creados pero no era la forma más eficiente de trabajo. Las pruebas estaban pensadas para realizarlas de forma manual en la que una persona pudiera tomar decisiones y así hacerlos lo más cortos posibles, pero esto no nos beneficiaba en nuestra tarea actual. Es por ello por lo que paralelamente a la automatización, hay un grupo de personas encargadas de intentar adaptar los casos ya

existentes a la nueva fase de pruebas que tenemos ahora, con el fin de poder crear flujos completos desde crear al paciente hasta darle el alta, cosa que no sucedía en los anteriores.

Analizando un poco el trabajo realizado hasta el momento lo primero que tenemos que ser conscientes es la importancia que tiene el equipo de QA y la función de testing que despeñan en este tipo de proyecto. Muchas veces le damos más importancia al diseño o al desarrollo de producto y es igual de importante una función que otra, porque igual de desastre sería para un producto que esté mal diseñado o desarrollado como que tenga errores que no le permitan funcionar de forma adecuada.

Además también la importancia que merece la automatización, todos los procesos que se puedan automatizar deben de hacerse, ya que es una forma de evitar riesgos por tener errores humanos, además del ahorro que ello supone. Es por ello por lo que la forma en la que hemos planteado las pruebas supone tener una mayor garantía de calidad y por lo tanto hacer un producto más seguro, que bien necesario es en este ámbito al tratarse de la salud.

Me gustaría recalcar también que hubiera sido un proyecto interesante realizar una tarea parecida en la universidad. Al igual que se trata la programación o la resolución de los diferentes problemas que te puedas encontrar, sería de gran utilidad que los alumnos salieran con algunos conocimientos sobre QA y automatización, ya que es un sector demandando por las empresas.

Por último concluir sobre el futuro de esta investigación, eso no ha sido solo un proyecto experimental para ver que funcionaba, sino que la finalidad era que tuviera una aplicación real para ponerla a disposición de la empresa. Actualmente hay un equipo de 6 personas de QA que está trabajando diariamente en automatizar todas las pruebas posibles con la herramienta seleccionada. Esto nos llevará a en futuro poder formar a los diferentes equipos de QA del resto de productos que se encuentran dentro de ehCOS para que ellos también puedan aprovecharse de dichas ventajas y así en un futuro conseguir que todos los productos estén automatizados y poder escalarlo a otros proyectos.

7. Bibliografía.

[1] Programa Naciones Unidas para el desarrollo. Objetivos del desarrollo sostenible [En línea]. Recuperado de: https://www.undp.org/es/sustainable-development-goals

[2]AIEDI Faktoria. Metas e indicadores del desarrollo sostenible [En línea]. Recuperado de: https://aiedifaktoria.com/metas-e-indicadores-del-ods-3/

[3]Naciones Unidas. Objetivos desarrollo sostenible [En línea]. Recuperado de: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[4] REAL ACADEMIA ESPAÑOLA: *Diccionario de la lengua española*, 23.ª ed., [versión 23.6 en línea]. https://dle.rae.es

[5] Schulmeyer, G. Gordon. Handbook of software quality assurance. Artech House, Inc., 2007.

[6] Tian, Jeff. Software quality engineering: testing, quality assurance, and quantifiable improvement. John Wiley & Sons, 2005.

[7] BMC. Muhammad Raza, 4 abril 2021 [En línea]. Recuperado de: https://www.bmc.com/blogs/quality-assurance-software-testing/

[8] Alexsoft software & engineering. Quality Assurance, Quality Control and Testing — the Basics of Software Quality Management.

https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/

[9] Laporte, Claude Y., and Alain April. Software quality assurance. John Wiley & Sons, 2018.

[10] Turning. What Is Software Quality Assurance, and Why Is It Important? Jayalakshmi Iyer, Sanika. 20 abril 2023. [En línea]. Recuperado de:

https://www.turing.com/blog/software-quality-assurance-and-its-importance/#:~:text=Ensures%20security%20and%20compliance%3A%20Software,to%20security%20and%20data%20privacy.

[11]Laporte, Claude Y. Libro Software qualty assurance. Captíulo 1 https://learning.oreilly.com/library/view/software-quality-assurance/9781118501825/c01.xhtml#c1_3

[12] Lee, Ming-Chang. "Software quality factors and software quality metrics to enhance software quality assurance." British Journal of Applied Science & Technology 4.21 (2014): 3069-3095.

[13] REAL ACADEMIA ESPAÑOLA: *Diccionario de la lengua española*, 23.ª ed., [versión 23.6 en línea]. https://dle.rae.es

[14]Antevenio. Que es el web scraping y para qué sirve, 2019. [En línea]. Recuperado de: https://www.antevenio.com/blog/2019/03/que-es-el-web-scraping-y-para-que-sirve/

[15] BOE. Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. [En línea]. Recuperado de: https://www.boe.es/eli/es/lo/2018/12/05/3

[16] BOE. Ley 41/2002, de 14 de noviembre, básica reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica. [En línea]. Recuperado de: https://www.boe.es/eli/es/l/2002/11/14/41

[17] BOE. Ley 14/2007, de 3 de julio, de Investigación biomédica. [En línea]. Recuperado de: https://www.boe.es/eli/es/l/2007/07/03/14

[18] BOE. Ley 28/2009, de 30 de diciembre, de modificación de la Ley 29/2006, de 26 de julio, de garantías y uso racional de los medicamentos y productos sanitarios.. [En línea]. Recuperado de: https://www.boe.es/eli/es/l/2009/12/30/28

[19] Colegio Notarial de Cataluña. Ley 10/2014, de 29 de diciembre, de la generalitat, de salud de la comunitat valenciana. [En línea]. Recuperado de:

https://www.colegionotarial.org/es/legislación/ley-102014-29-diciembre-generalitat-salud-comunitat-valenciana

[20]Emergo by UL. What is CE Marking for medical devices? [En línea]. Recuperado de: https://www.emergobyul.com/services/european-ce-marking-strategy-medical-devices#:~:text=What%20is%20CE%20Marking%20for,across%20all%20EU%20member%20states

[21]Flood.IO [En línea]. Recuperado de: https://www.flood.io/tools

[22] Ubik Ingénierie. Mayo 2020 [En línea]. Recuperado de: https://www.ubik-ingenierie.com/blog/running-a-video-streaming-performance-test-using-flood-io-and-ubikloadpack/

[23]Smart Bear Suport. Load Ninja Documentation. 5 Noviembre 2022. [En línea]. Recuperado de: https://support.smartbear.com/loadninja/docs/use-cases/index.html

[24]Smart Bear Load Ninja [En línea]. Recuperado de: https://loadninja.com

[25] Radview. Cloud Load Testing. [En línea]. Recuperado de: https://www.radview.com/solution/cloud-load-testing/

[26] Katalon Docs. Katalon Recorder GUI Overview [En línea]. Recuperado de:

https://docs.katalon.com/docs/plugins-and-add-ons/katalon-recorder-extension/get-started/katalon-recorder-gui-overview

[27] Chrome Web Store. Selenium IDE [En línea]. Recuperado de: https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd

[28] Selenium IDE. [En línea]. Recuperado de: https://www.selenium.dev/selenium-ide/

8. Anexo 1: Relación del TFG con los ODS.

En 2015, los 193 Estados miembros de la Organización de las Naciones Unidas (ONU) establece en el marco de trabajo de la Agenda 2030 una serie de objetivos globales para erradicar la pobreza y mitigar los efectos del cambio climático. Estos objetivos han sido presentados a nivel mundial como los Objetivos de Desarrollo Sostenible (ODS) y comprenden una serie de acciones clave en un total de 17 categorías que contienen metas e indicadores concretos.

La filosofía detrás de los ODS es que cualquier ser humano, sea cual sea su país de origen, posición social o recursos, realice acciones encaminadas a contribuir a las metas e indicadores.

En este marco, el presente Trabajo Fin de Grado contribuye a los siguientes ODS:

- Objetivo 3: Garantizar una vida sana y promover el bienestar de todos a todas las edades.

  El ámbito de la salud es uno de los temas más importantes a nivel mundial, ya que podemos encontrar multitud de países que no cuentan con los recursos suficientes para poder desempeñar las labores sanitarias de de una forma adecuada y segura.

  - Meta 3.8: Lograr la cobertura sanitaria universal, incluida la protección contra los riesgos financieros, el acceso a servicios de salud esenciales de calidad y el acceso a medicamentos y vacunas inocuos, eficaces, asequibles y de calidad para todos.

    Garantizar que la sanidad llegué de forma óptima a todos los países del mundo es uno de los objetivos que se debe plantear el mundo de hoy en día. El acceso a la sanidad es un derecho humano que todo el mundo debería tener al alcance independientemente del lugar dónde resida. Es por ello por lo que el presente trabajo de final de grado propone un método para implementar en un software de ámbito sanitario que permita crear un software más seguros que beneficie a las personas y reducir los costes del producto, por lo que países con menos recursos también puedan optar a conseguir tener estos programas a su alcance y así conseguir una mejor atención para sus ciudadanos.