UNIVERSIDAD NACIONAL
Escuela de Informática
Licenciatura

Prácticas y experiencias en automatización de pruebas funcionales de integración en C#.

Para optar por el grado de Licenciatura en Sistemas de Información.

Por:

Fabiana Salas Villalobos

Hugo Jiménez Chévez

ENERO, 2021

Primeramente, queremos darnos un agradecimiento mutuamente por el gran trabajo en equipo que logramos realizar, ya que, siempre se dio lo mejor de cada uno, por el esfuerzo demostrado y por los ánimos y los conocimientos que fueron fundamentales para la elaboración de este proyecto. Además, agradecemos sinceramente a nuestro profesor tutor Fulvio Lizano por aceptar llevar este proceso con nosotros y siempre darnos la retroalimentación necesaria para sacar lo mejor de nosotros. Por último, queremos agradecer a nuestras familias por el apoyo que nos brindaron a lo largo de este camino y por darnos las fuerzas para culminar con éxito este proyecto.

TABLA DE CONTENIDOS

Tablas

Ilustraciones

CAPÍTULO I
INTRODUCCIÓN

CAPÍTULO I: INTRODUCCIÓN

1. Antecedentes

La inquietud por la calidad de software ha crecido a medida que la imagen de las empresas pasó a estar cada vez más expuesta, esto ha llevado a buscar medios que permitan perfeccionar la calidad y uno de estos caminos fue el mejoramiento de las actividades relacionadas con el test de software a través de la automatización (Peres, 2018).

Analizando estudios previos acerca de la temática a desarrollar, el problema que aborda Rivera (2018) en su tesis es la reducción en el tiempo y esfuerzo que requiere el proceso de pruebas manuales mediante la implementación de pruebas automatizadas de regresión para la compañía de TV Chile. Sin embargo, este trabajo se encuentra enfocado en los procesos y necesidades de una empresa en específico por lo que no es un buen referente en el proceso general de automatización.

Cárdenas (2016) describe el proceso de implementación de una estrategia para el mantenimiento de casos de pruebas de regresión automatizados con el fin de lograr la optimización del proceso para una empresa de pagos Online; además se detallan los pasos a seguir para la ejecución de la automatización bajo el marco Scrum. No obstante, en el trabajo no se especifica el procedimiento de creación e implementación de la automatización. Luego de analizar estos trabajos previos, la conclusión es que existen pocos estudios que describan comparativamente procesos para generar casos de prueba automatizados para las pruebas funcionales de integración.

Por otra parte, el tema de esta investigación surgió gracias a la observación de una empresa que se encontraba realizando una reestructuración al departamento de aseguramiento de la calidad, para así lograr la implementación de un proceso de automatización de casos de prueba. Durante las etapas iniciales de esta reforma, se evidencio una gama poco abundante de investigaciones relacionadas al tema de la automatización, que le permitiera a la organización sustentar este proceso que se encontraban realizando.

Por lo que, durante la ejecución del Trabajo Final de Graduación (TFG) se pretende realizar una serie de experimentos en laboratorio, mediante un conjunto de prácticas y guías enfocadas en la automatización de pruebas funcionales de integración en C#, que permitan obtener experiencias que sirvan como referente del tema para los profesionales en el área de calidad.

2. Planteamiento del problema

Las pruebas automatizadas se han convertido en elementos necesarios para las organizaciones, esto debido a que ayudan a detectar fallos de manera eficiente y eficaz sin la necesidad de que las personas revisen el software componente por componente (Aristegui, 2010). De esta manera, se alcanzan grandes beneficios entre los cuales se pueden encontrar ejecución rápida y efectiva de las pruebas, mayor productividad, ahorro en costos y duración, aumento de rendimiento, entre otros (Cárdenas, 2016).

Vinasco, Roca y Pardo (2013) nos mencionan que las empresas u organizaciones necesitan un apoyo para obtener mejoras en sus procesos de calidad, ya que estas requieren lograr una mayor precisión en la elaboración y ejecución de sus planes de pruebas automatizados, esto debido a que es de suma importancia que las pruebas cumplan con los requisitos necesarios para el éxito del producto final. Sin embargo, al analizar la literatura existente sobre el tema, se nota una escasa gama de investigaciones relacionadas al ámbito de la automatización, específicamente, para la creación y ejecución de las pruebas funcionales de integración.

De esta forma, el problema bajo estudio en este TFG consiste en el desarrollo de prácticas y experiencias en automatización de pruebas funcionales de integración en C#, que sirvan como referente para incrementar la calidad de los productos de software.

3. Justificación

La automatización de casos de prueba es un elemento primordial para mejorar la calidad en el desarrollo de productos de software. Puesto que las aplicaciones cada vez se vuelven más complejas y gracias a las nuevas metodologías los requerimientos son

elementos cambiantes lo que produce que se generen gran cantidad de versiones y esto provoca una reducción en la confiabilidad del producto final (Cubas, 2017).

Sin embargo, si se realiza una estrategia integral de automatización les permitirá a las empresas realizar las pruebas de una manera más rápida, reduciendo esfuerzos ya que simplifica el trabajo repetitivo o complejo (Rodríguez, 2019). Además, si se ejecutan estas pruebas automatizadas con mayor frecuencia permiten detectar de manera temprana los errores existentes, lo que permite reducir costos y obtener una retroalimentación rápida sobre la calidad del sistema (Toledo, 2014).

Por lo tanto, los profesionales en informática específicamente en el área de la calidad serán los principales beneficiados con esta investigación, ya que se pretende mediante la experimentación generar experiencias que puedan servir como referentes sobre cuáles son los mejores métodos para incrementar la calidad a la hora de ejecutar pruebas automáticas de integración en C#.

De modo que, la importancia de este TFG radica en brindar un apoyo con el que se procura generar una mayor calidad en los productos de software mediante la automatización, utilizando así las prácticas, herramientas y experiencias que se pretenden sugerir y generar a través del presente trabajo. Puesto que, lo principal para las organizaciones es mejorar la calidad y el desempeño a la hora de desarrollar productos de software, ya que buscan producir un grado de madurez significativo en el mercado y así ofrecer lo mejor a sus clientes (Pantaleo, 2011).

4. Objetivos del Proyecto

4.1. Objetivo general

Desarrollar por medio de experimentación en laboratorio, un conjunto de prácticas y experiencias en automatización de pruebas funcionales de integración en C#, que sirvan como referente del tema a los profesionales del área de la calidad en software.

4.2. Objetivos específicos

1. Revisar la literatura relacionada con la automatización de pruebas de integración para así identificar los principales elementos y procesos relacionados con esta técnica, por medio de un proceso sistemático de revisión de literatura.
2. Diseñar un experimento de laboratorio donde se detallen scripts para las pruebas de integración automatizadas utilizando C#. Además, de guías y protocolos de gestión de datos para ser aplicadas a un sistema de prueba por definir.
3. Desarrollar el experimento diseñado siguiendo las guías hechas para tal fin, con el objetivo de registrar los datos obtenidos en las pruebas automatizadas.
4. Analizar los resultados obtenidos en el experimento de pruebas de integración automatizadas utilizando C# con el fin de documentar las principales prácticas y experiencias sobre el tema derivadas del TFG.

CAPÍTULO II
MARCO TEÓRICO

CAPÍTULO II: MARCO TEÓRICO

En este capítulo se presenta el marco teórico que sustenta el presente proyecto, este se encuentra organizado en 4 partes que poseen la definición de algunos de los términos primordiales para la formulación del TFG. La primera parte abarca la definición de calidad y su importancia en el ámbito del software. Seguidamente, se menciona la definición y tipos de pruebas funcionales, haciendo énfasis en las pruebas funcionales de integración.

Por otro lado, la tercera parte, está relacionada con los tipos de ejecución de pruebas, donde se destaca la definición de la automatización de pruebas. Y para finalizar se mencionan las definiciones de error, mantenimiento y eficiencia elementos primordiales en la automatización de casos de prueba.

1. Calidad del software

La calidad de software es un conjunto de características o atributos que se le otorgan a los programas para poder medir su utilidad y su funcionamiento. Además, gracias a ella se establecen estándares para determinar parámetros de evaluación que permiten identificar si el producto cuenta con un índice alto de calidad, con el que se puede asegurar la máxima eficacia de los productos finales (Pantaleo, 2011).

Por otra parte, la calidad permite tener una mejor visión del problema que se enfrenta con el fin de que se logre resolver de una manera eficaz y rápida cumpliendo con ciertos factores de calidad. Según Peres (2018) una garantía que brinda la calidad es la posibilidad de auditar sistemas lo que logra ganar confianza en la resolución de problemas.

2. Pruebas funcionales.

Las pruebas de software son un conjunto de actividades que permiten asegurar la calidad de los componentes de un sistema. Para esto, existen varios tipos de pruebas entre las cuales se encuentran: pruebas funcionales, pruebas no funcionales, pruebas de caja blanca, pruebas de caja negra, entre otras (Cárdenas, 2016).

Conviene subrayar, que las pruebas funcionales son aquellas que se encargan de evaluar las funciones que el sistema debe realizar. Además, este tipo de prueba observa el

comportamiento del software por ende están basadas en la ejecución, revisión y retroalimentación diseñadas para este (Olsen, 2018). Por otro lado, existen diferentes tipos de pruebas funcionales entre las cuales se destacan las pruebas funcionales de integración que se describirán a continuación.

2.1. Pruebas Funcionales de integración

Müller (2011) menciona que las pruebas funcionales de integración son aquellas que se ocupan de probar las interfaces entre los componentes de distintas partes del mismo sistema para asegurarse el correcto ensamblaje entre los distintos componentes.

El objetivo de estas pruebas es tener la posibilidad de localizar los errores de interfaz y poder comprobar el buen funcionamiento del conjunto de componentes de los sistemas. Existen varios tipos de estrategias para realizar las pruebas funcionales las cuales son: Top-Down, Bottom-Up, End-To-End, Funciones, entre otros (Gómez, 2015).

3. Tipos de ejecución de Pruebas de Software

Actualmente, existen dos tipos de ejecución de pruebas, estas son la manual y la automatizada. Las pruebas manuales son aquellas que son ejecutadas por los analistas de desarrollo o de pruebas que permiten verificar las funcionalidades del sistema. A diferencia de las pruebas automatizadas, que son realizadas a través de herramientas que van a permitir la comparación de resultados del sistema con los resultados esperados (Cárdenas, 2016).

3.1. Automatización de pruebas

Como se ha definido anteriormente, la automatización de pruebas se refiere a conseguir que las pruebas que se realizan de manera manual logren ser ejecutadas sin intervención de una persona, por medio de alguna herramienta destinada a realizar el proceso automáticamente (Cubas, 2017). Por lo que, la automatización de pruebas tanto funcionales como no funcionales logra optimizar los procesos de calidad reduciendo esfuerzos.

Por lo tanto, los procesos automatizados de pruebas generan grandes beneficios a las empresas trayendo ventajas asociadas a la mejora de la calidad y el performance de

producción. Sin embargo, esto no significa un aseguramiento de la confiabilidad del software, por lo que, siempre debe ser un complemento de la ejecución manual (Toledo, 2014).

De acuerdo a Cubas (2017), es primordial para el proceso de automatización desarrollar una estrategia de pruebas. En esta se debe determinar en qué partes de la aplicación es un beneficio realizar procesos automáticos, ya que no todas las pruebas son automatizables. Además, si no se automatiza con criterio no se va a obtener beneficio alguno.

4. Error en informática

Un error o bug, es un término que se utiliza en la informática para hacer referencia a un defecto o fallo que ocurre a la hora de ejecutar un sistema o programa. Este tipo de bugs pueden ocurrir por diferentes motivos, por ejemplo, una persona puede cometer un error que a su vez puede repercutir en un defecto que puede dañar la funcionalidad de un sistema y además esto haría que el programa no cumpla con los requisitos establecidos. (Müller, 2011).

Por lo que, a la hora de realizar revisiones de software hay que tener presente que los errores pueden ser de diferentes tipos, por lo que se debe de plantear una estrategia a la hora de abordarlos. Es decir, las verificaciones de software van a convertirse en el principal filtro para detectar errores en el desarrollo del software en todas las etapas (Pérez, 2006).

Cabe resaltar, que la fiabilidad de un programa se encuentra estrechamente relacionada con la cantidad de defectos que se encuentren en el mismo. Ya que, al definirse la fiabilidad como la probabilidad que durante un determinado periodo de tiempo el sistema funcione tal y como el usuario lo espera, al encontrarse errores influye directamente en la confiabilidad del producto final ya que la fiabilidad disminuye o deja de existir (Sommerville, 2005). Además, los errores que se encuentran en las aplicaciones reducen la eficiencia y la satisfacción de los usuarios, ya que, a mayor cantidad de errores el sistema pierde usabilidad y genera un ambiente defectuoso (Grau, 2000).

5. Mantenimiento

El mantenimiento es la característica que permite que un producto de software pueda ser modificado de manera efectiva, esto debido a las necesidades cambiantes y evolutivas de los sistemas, y así de esta manera poder evitar su degradación (Cubas, 2017). Según Sommerville (2005), un software mantenible es un software adaptable a los nuevos requerimientos, es decir, que fue diseñado para la introducción de nuevos cambios a un bajo costo y con una probabilidad baja de introducir errores al sistema con los cambios realizados.

La importancia de aplicar mantenimiento a las herramientas es que según Hayes (2004) el 25 % de una aplicación es reescrita, esto debido a diferentes factores que se pueden presentar. Algunos ejemplos pueden ser defectos en la aplicación, un módulo no se terminó a tiempo, entre otros. Por ende, dar mantenimiento a las aplicaciones puede asegurar su calidad.

Asimismo, el mantenimiento de software representa para los desarrolladores un mayor esfuerzo que cualquier otra actividad en la realización de un sistema o en la elaboración de código o de scripts, ya que estos deben volver a configurarse y además identificar el error presentado. Por lo que, el código desarrollado debe proporcionar extensibilidad y ser adaptable para que los equipos puedan tener la capacidad de brindar un mantenimiento adecuado (Pressman, 2010).

6. Eficiencia

Es la capacidad de realizar una serie de pasos y procedimientos para garantizar la calidad en los productos creados. Además, representa el desempeño relativo a la cantidad de recursos utilizados, estos se miden por el tiempo de respuesta, la cantidad de recursos utilizados, entre otros (Cubas, 2010). Algunos de los elementos primordiales en términos de eficiencia se describen a continuación.

6.1. Utilización de recursos

La utilización de recursos se puede definir como la cantidad de recursos que el sistema consume para su óptimo funcionamiento. Asimismo, se puede decir que son las

funciones que se llevan a cabo bajo condiciones determinadas, ya que de estas pruebas de carga se puede determinar si un sistema puede o no cumplir con las necesidades de los clientes (Cubas, 2017).

6.2. Capacidad

Es la capacidad que tiene un parámetro o aplicación en cumplir los requisitos esperados en óptimas condiciones (Cubas, 2017). No obstante, la capacidad de un aplicativo radica en la manera que este puede ser manejable y configurable, esto con el objetivo de que exista facilidad de utilizar el sistema en óptimas condiciones. Es decir, trata de analizar el comportamiento de dichas operaciones durante su ejecución (Pressman, 2010).

6.3. Rendimiento

El rendimiento del software se puede medir mediante la capacidad que este tiene en dar un tiempo de respuesta, en la velocidad en la que realiza el proceso y el uso de recursos que utilice durante la ejecución, por otro lado, representa la eficiencia con la que trabaja (Pressman, 2010).

De otra manera, el rendimiento del software se encuentra estrechamente relacionado con la confiabilidad del mismo. Esto debido, a que los niveles altos de confiabilidad sólo pueden alcanzarse por medio del rendimiento del sistema (Sommerville, 2005).

.

CAPÍTULO III
METODOLOGÍA

CAPÍTULO III: METODOLOGÍA

1. Tipo de investigación

El presente TFG se realizará mediante experimentación, la cual es un método que permite identificar las causas por las que se producen determinados resultados, mediante la manipulación de variables que permiten controlar y medir cualquier cambio en otras variables. Por otra parte, un experimento en laboratorio permite estudiar la relación causa y efecto de las variables, mediante el control riguroso de las condiciones. Es decir, que la principal ventaja de este tipo de experimento es que permite modificar iterativamente una variable para estudiar su impacto en otras variables (Williamson y Johanson, 2018).

1.1. Definición de variables

Entrando en detalle, en los experimentos de laboratorio se manipulan dos tipos de variables, las independientes y las dependientes. Las variables independientes se pueden definir como las causales, estas son manipuladas por el investigador con el fin de analizar su impacto en otras variables. Por otro lado, las variables dependientes se definen como variables de efecto, ya que son el factor que mide cómo impacta una causa en particular (Williamson y Johanson, 2018). La definición de variables para este proyecto es la siguiente.

1. Variable independiente: Framework de automatización.
2. Variables dependientes:
    a. Eficiencia de ejecución.
    b. Mantenimiento del script.
    c. Cantidad de errores encontrados

A continuación, profundizaremos en los frameworks de automatización. Lo respectivo a las variables dependientes ya fue abordado en el marco teórico de este TFG.

1.2. Frameworks de automatización

Los frameworks son un conjunto de estándares y reglas que facilitan la ejecución y comprensión de los scripts automatizados (Kidd, 2018). Por lo que, existen diversos

enfoques en términos del proceso de automatización, los cuales deben ser tomados en cuenta a la hora de seleccionar la estrategia y las herramientas más adecuadas para el desarrollo de este proceso (Toledo, 2014). A continuación, se desarrollarán cada uno de los frameworks que se utilizarán en el presente TFG.

1.2.1. Linear Scripting - Record & Playback

Según Toledo (2014), la idea principal de este enfoque es la utilización de herramientas que sean capaces de capturar las acciones que realiza el usuario sobre el sistema que se está probando, para luego poder convertir estas acciones en un script que sea reproducible. La ilustración 1 muestra el proceso de este enfoque en tres pasos.

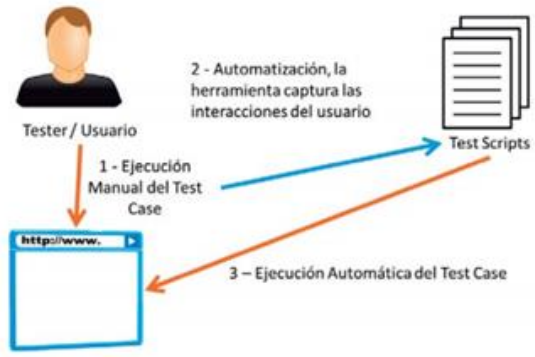

Ilustración 1 Enfoque de Record & Playback (Toledo, 2014, p.109).

La principal ventaja que tiene este enfoque es que requiere la menor cantidad de entrenamiento y preparación, ya que al ser tan sencillo de utilizar necesita una curva de aprendizaje corta (Hayes, 2004). Además, no se necesita escribir código personalizado por lo que cualquier persona podría ejecutarlo sin la necesidad de ser expertos en automatización (Kidd, 2018).

1.2.2. The Test Library Architecture Framework

La idea principal de este enfoque es la modularización, pero esta no se encuentra enfocada en dividir la aplicación en varios scripts, sino en encontrar funcionalidades similares y agruparlas en funciones. Estas funciones estarán almacenadas en bibliotecas, y los scripts de pruebas pueden llamar a la biblioteca para hacer uso de las funciones que sean

necesarias, lo que permite la reutilización de código y la facilidad en creación de múltiples scripts (Kidd, 2018).

3. Descripción de instrumentos

Tal como lo mencionamos antes, la metodología que se utilizará para la recolección de datos es la experimentación, la cual permite la manipulación de variables explorando la relación causal. Los experimentos son una manera directa, precisa y confiable de recolectar datos para una investigación (Williamson y Johanson, 2018).

Los instrumentos de recolección de datos a utilizar son diarios y listas de cotejo. Los diarios son un tipo de cuestionario auto administrado que permite la recopilación de experiencias, en estos se llena información elemental cada vez que ocurre un evento o en intervalos especificados de tiempo (Sajjad, 2016). Para efectos del presente TFG, los diarios como su definición lo menciona serán la principal herramienta para la recolección de experiencias durante toda la ejecución del proyecto. De este modo, se permitirá un análisis e interpretación de los datos con el fin de obtener los resultados esperados (Ver Anexo 1).

Asimismo, las listas de cotejo son un tipo de instrumento que permite la agrupación de criterios para la evaluación ya que actúan como un mecanismo de revisión mediante indicadores prefijados (Cabrera y Lara, 2015). Por lo tanto, esta herramienta será utilizada para la evaluación del comportamiento de las variables independientes mediante criterios predefinidos en una plantilla a utilizar (Ver anexo 2).

4. Procedimientos para analizar la información del diagnóstico

Durante la ejecución de este TFG se seguirán los siguientes pasos con el propósito de cumplir los objetivos planteados.

1. En primer lugar, se procederá a realizar un análisis exhaustivo de literatura relacionado con la automatización de pruebas de integración en C#, esto con la finalidad de adquirir conocimientos sobre el tema a desarrollar.
2. Luego, se dará paso al diseño del experimento, en el cual se creará una guía donde se detallen los protocolos de gestión de datos que se emplearán.

3. El siguiente paso a realizar, es la configuración del ambiente con los aspectos necesarios para ejecutar los casos de prueba automatizados. Estas necesidades del ambiente serán definidas más adelante en este apartado.
4. Luego, se iniciará con la creación de los scripts automatizados bajo los dos frameworks de automatización definidos anteriormente. La programación de estos scripts se realizará bajo la metodología ágil denominada Kanban, la cual permite una simplificación en la planificación y asignación de responsabilidades mediante la utilización de un tablero de tareas que representará el flujo de trabajo (Muradas, 2018).
5. Posteriormente, se iniciará con el desarrollo del experimento siguiendo la guía diseñada para la ejecución de los scripts automatizados con el objetivo de registrar los datos obtenidos.
6. Una vez registrados los resultados, se realizará un análisis de datos el cual consiste en organizar y manipular la información obtenida para que esta pueda ser analizada e interpretada.
7. Para finalizar, se elaborarán las conclusiones que incluirán las principales prácticas y experiencias sobre el tema derivadas del TFG.

5. Necesidades del ambiente

Para la configuración del ambiente, se necesitará la creación de una máquina virtual que posea Windows 10 (Microsoft, s.f.). Además, se utilizarán varias herramientas entre las cuales se encuentra Microsoft Azure DevOps la cual ofrece una alta gama de servicios, como lo son:

- Overview: Permitirá la creación de los dashboard para mostrar los resultados de las pruebas automatizadas.
- Repositorios: Contendrá el código fuente del sistema de pruebas y los scripts.
- Board: Ayudará a utilizar la metodología ágil Kanban, mediante la creación de las diferentes tareas.

- Pipelines: Permitirá la ejecución de las pruebas de manera automática, ya que en esta sección se maneja la integración continua.
- Test plan: Tendrá almacenados los casos de pruebas que se necesiten para realizar el experimento.

Asimismo, esta herramienta posee grandes beneficios como lo son la optimización de los recursos, la creación de productos de calidad en el menor tiempo posible y la integración del sistema de manera automática. Por otra parte, Azure permite controlar el ciclo de vida de los proyectos logrando implementar una integración continua, parte fundamental para la ejecución de los scripts (Microsoft, s.f).

Por otra parte, se necesitará instalar Visual Studio Community como IDE principal para la programación de los scripts automatizados bajo el framework de Test Library Architecture. A su vez, se utilizará la herramienta Selenium tanto para la grabación de los casos de prueba bajo el framework Record and Playback, como para la ejecución de pruebas en Visual Studio.

6. Cantidad de Scripts a desarrollar

Dadas las consideraciones de recursos disponibles para el TFG, se tomó la decisión de elaborar 10 scripts automatizados por cada tipo de framework a utilizar. Por lo que, el número total de scripts que se desarrollarán y ejecutarán en este TFG es de 20.

7. Sistema de prueba a utilizar

Para el desarrollo del experimento, se utilizará un sistema de prueba tanto para la definición de los casos de prueba como para la puesta en ejecución de los scripts desarrollados. Esta aplicación se denomina University Management System, la cual tiene como principal objetivo simular gestiones de un sistema universitario. Por otra parte, este sistema se encuentra en un repositorio público de github con acceso libre para su uso y modificación (Minhaz, 2016).

CAPÍTULO IV
PROPUESTA DE SOLUCIÓN

CAPÍTULO IV: PROPUESTA DE SOLUCIÓN

1. Diagnóstico

El diagnóstico del presente TFG fue realizado por medio de un análisis exhaustivo de literatura, esté relacionado a la automatización de pruebas funcionales de integración en C# y a la utilización de frameworks durante la etapa de desarrollo de los scripts. Sin embargo, como se menciona anteriormente existe una gama escasa de investigaciones relacionadas a estos temas.

La automatización se creó inicialmente con el propósito de reducir el esfuerzo humano, ya que al automatizar existe la ventaja de ejecutar un número ilimitado de veces las pruebas realizadas lo que permite un ahorro en tiempo y costos. Por lo tanto, el principal objetivo de las pruebas automatizadas es buscar la calidad de aquellos sistemas que tienen alta demanda laboral o son difíciles de realizar pruebas manuales por lo extenso que pueden llegar a ser (Rodriguez, 2019).

No todas las pruebas son automatizables; el costo de desarrollar y mantener los scripts de pruebas es elevado y no debe ser tomado a la ligera. Consecuentemente, se debe realizar una evaluación inicial sobre la inversión, el enfoque, los beneficios y el conocimiento del proceso manual; esto con el objetivo de determinar si vale la pena realizar el proceso de automatización (Rodriguez, 2019).

De la misma forma, la falta de automatización puede incurrir en muchas desventajas para las empresas u organizaciones, una de estas es que se requiere invertir en recursos humanos lo que genera una elevación en los costos. Además, el proceso de revisiones se vuelve más lento ya que las pruebas de los sistemas se realizan de manera manual y la velocidad de las personas es un elemento variable (Arteaga, 2013).

Por otra parte, antes de empezar a desarrollar pruebas de automatización es elemental realizar un diagnóstico previo con el fin de determinar cuáles van a ser las herramientas y técnicas o frameworks a utilizar. Asimismo, es importante recalcar que los frameworks de automatización como se menciona anteriormente son un conjunto de pautas

y reglas que son utilizadas para la creación y diseño de casos de prueba automáticos (Mubarak y Zhanfang, 2019).

A pesar de que las pruebas automatizadas se puedan codificar y grabar solo con la utilización de herramientas de automatización, integrar el manejo de frameworks brindará beneficios adicionales al proceso que de otro modo se perderían (Mubarak y Zhanfang, 2019). Por lo tanto, la utilización de un framework de automatización y una herramienta de automatización de pruebas adecuada son esenciales para mejorar el proceso de automatización.

En otras palabras, los frameworks de automatización permitirán lograr una mayor reutilización de componentes de prueba, desarrollar scripts de fácil mantenimiento y obtener scripts de alta calidad. Además, ayudarán a aumentar la velocidad y la eficiencia mejorando así la precisión en la detección de errores y la reducción de riesgos (Mubarak y Zhanfang, 2019).

2. Propuesta de solución

La solución que se plantea para este TFG (Desarrollar prácticas y experiencias en automatización de pruebas funcionales de integración en C#) es la realización de un experimento de laboratorio. En primer lugar, es importante recordar como se menciona anteriormente, que un experimento de laboratorio es un método que nos permite identificar las causas por las que se producen determinados resultados mediante la manipulación de variables (Williamson y Johanson, 2018). Asimismo, en este TFG se utiliza una metodología mixta que implica un método de recolección de datos tanto cualitativo como cuantitativo. De la misma forma, el objetivo principal de este experimento es, tal como lo mencionamos arriba, producir por medio de estos resultados obtenidos durante su ejecución un conjunto de prácticas y experiencias referentes al tema de automatización de pruebas de integración en C# bajo los frameworks mencionados con anterioridad.

2.1. Variables y frameworks de automatización

De la misma forma, como se describe en apartados anteriores la definición de variables para este TFG es la siguiente.

1. Variable independiente: Framework de automatización.
2. Variables dependientes:
    a. Eficiencia de ejecución.
    b. Mantenimiento del script.
    c. Cantidad de errores encontrados

En el contexto de la variable independiente, se definen las condiciones del experimento, es decir, los tipos de frameworks a utilizar, mismos que producirán diferentes resultados en las variables dependientes. En nuestro caso, las condiciones de la variable independiente son “Record and Playback” y “Test Library Architecture”.

2.2. Casos de Prueba

Los casos de prueba son un conjunto de condiciones o variables que se utilizan para determinar si un sistema bajo prueba cumple con los requerimientos especificados por el usuario. Además, los casos de prueba se encuentran estrechamente relacionados con los scripts de automatización, es así que de un caso de prueba se pueden obtener uno o múltiples test scripts (Franco, 2010).

Para fines del presente TFG, se cuenta con 10 casos de prueba que poseen la mayor cobertura posible del sistema de prueba seleccionado. Además, los casos de prueba serán utilizados para elaborar los scripts bajo ambos frameworks de automatización. Por lo que, los pasos del script son los mismos sin importar el framework. En la Tabla 1 se puede observar el código y detalle de los 10 casos de prueba, mismos que pueden ser consultados en detalle en el Anexo 3.

Tabla 1 Resumen de Casos de Prueba

Código	**Descripción**
TC01	Crear Departamento: Caso de prueba que se encarga de detallar el proceso de creación de un departamento.
TC02	Ingresar código de departamento repetido: Caso de prueba que se encarga de detallar el proceso de validación del ingreso de códigos repetidos en diferentes departamentos.
TC03	Crear curso: Caso de prueba que se encarga de detallar el proceso de creación de un curso.
TC04	Crear profesor: Caso de prueba que se encarga de detallar el proceso de creación de un profesor.
TC05	Asignar curso a profesor: Caso de prueba que se encarga de detallar el proceso de la asignación de curso a un profesor.
TC06	Crear estudiante: Caso de prueba que se encarga de detallar el proceso de registro de estudiantes.
TC07	Crear horario de clase: Caso de prueba que se encarga de detallar el proceso del horario de las clases.
TC08	Registrar estudiante en un curso: Caso de prueba que se encarga de detallar el proceso de registro de los estudiantes en los cursos.
TC09	Añadir notas al estudiante: Caso de prueba que se encarga de detallar el proceso de asignación de las notas de los estudiantes.
TC10	Ver Notas del estudiante: Caso de prueba que se encarga de detallar el proceso visualizar las notas registradas de los estudiantes.

Fuente: Elaboración propia

2.3. Scripts automatizados de prueba

Los scripts automatizados son aquellos que verifican la funcionalidad de los sistemas, y estos se programan a partir de la confección de los casos de prueba, ya que de estos se especifican el procedimiento que deben seguir (Rivera, 2018). Según Cardenas (2016), algunos beneficios que nos podemos encontrar con la confección de scripts son los siguientes:

1. Ejecución rápida y efectiva.
2. Se puede reutilizar código.
3. Mayor productividad.
4. Se pueden adaptar a las diferentes metodologías.
5. Se aumenta el rendimiento.

Por otro lado, para efectos de este TFG se utilizarán 10 scripts por cada framework para un total de 20. Estos scripts serán almacenados en un repositorio de Github, la cual es una plataforma que se utiliza para controlar las versiones del código de forma gratuita, está ofrece diferentes servicios como lo son poder alojar el código, clonar los repositorios, y modificar el código (Cardenas, 2016).

Para el desarrollo de los scripts se necesita utilizar diferentes herramientas de acuerdo al framework seleccionado. Para el framework Record and Playback, se debe utilizar Selenium Web Driver para grabar los pasos de los casos de prueba y posteriormente exportarlos en C#, luego se necesitará Visual Studio IDE para visualizar el código generado y ejecutarlo de manera automática. Por otra parte, para el framework Test Library Architecture se necesita utilizar Visual Studio IDE y Selenium para el desarrollo del código y para la ejecución de los scripts. Por lo que, se puede decir que la creación de los test scripts tienen diferentes formas de ser elaborados de acuerdo al framework, aunque su ejecución sea igual.

En la Tabla 2 se puede observar el código y nombre de los 10 scripts desarrollados bajo el framework Record and Playback, mismos que pueden ser consultados en detalle en el Anexo 4

Tabla 2 Scripts framework Record and Playback

Código	**Nombre Scripts**
01	TC01CrearDepartamentoTest.
02	TC02IngresarcodigodepartamentorepetidoTest.
03	TC03CrearcursoTest.
04	TC04CrearprofesorTest.

05	TC05AsignarcursoprofesorTest.
06	TC06CrearestudianteTest.
07	TC07CrearhorarioclaseTest.
08	TC08RegistrarestudiantecursoTest.
09	TC09AñadirnotasestudianteTest.
10	TC10VerNotasdelestudianteTest.

Fuente: Elaboración propia

En la Tabla 3 se puede observar el código y nombre de los 10 scripts desarrollados bajo el framework Test Library Architecture, mismos que pueden ser consultados en detalle en el Anexo 5.

Tabla 3 Scripts framework Test Library Architecture

Código	**Nombre Scripts**
00	Librería
01	TC01Crear Departamento.
02	TC02Ingresarcodigodepartamentorepetido.
03	TC03Crearcurso.
04	TC04Crearprofesor.
05	TC05Asignarcursoprofesor.
06	TC06Crearestudiante.
07	TC07Crearhorarioclase.
08	TC08Registrarestudiantecurso.
09	TC09Añadirnotasestudiante.
10	TC10VerNotasdelestudianteTest.

Fuente: Elaboración propia

2.4. Protocolo de gestión de datos

A continuación, se encuentran los pasos a seguir para el diseño, ejecución y análisis del experimento a realizar en este TFG.

1. Se dará inicio a la etapa de diseño del experimento. Inicialmente, se realizarán las plantillas de las herramientas de recolección de datos que serán utilizadas en la etapa de ejecución del experimento.
2. El siguiente paso a realizar, es la configuración del ambiente con los aspectos necesarios para ejecutar los casos de prueba automatizados.
3. Luego, se iniciará con la creación de los scripts automatizados bajo los dos frameworks de automatización definidos anteriormente. La programación de estos scripts se realizará bajo la metodología ágil denominada Kanban.
4. Posteriormente, se iniciará con la etapa de desarrollo del experimento. El principal objetivo de esta etapa es iniciar con la ejecución diaria de los scripts automatizados con el objetivo de registrar los datos obtenidos por un lapso de 15 días. Esta ejecución se realiza de manera automática en la herramienta Azure DevOps, gracias a la configuración de tareas específicas en los pipelines de pruebas.
5. Una vez registrados los resultados, se pasará a la etapa de análisis en la cual se realizará una recolección de datos el cual consiste en organizar y manipular la información obtenida para que esta pueda ser analizada e interpretada. Y a partir de esto se registrará los resultados obtenidos.
6. Para finalizar, se elaborarán las conclusiones que incluirán las principales prácticas y experiencias sobre el tema derivadas del TFG.

3. Validación de la propuesta

En esta sección se pueden visualizar los resultados obtenidos de la puesta en ejecución del experimento que se encuentra detallado en los capítulos anteriores de este

documento. Como se mencionó anteriormente, para la validación de la propuesta se puso en marcha la etapa de desarrollo del experimento, en la cual se ejecutaron por 15 días los 20 scripts desarrollados bajo los frameworks Record and Playback y Test Library Architecture, esto con el fin de medir los resultados de las variables dependientes que se propusieron.

Si bien es cierto que el problema bajo estudio en este TFG es el generar prácticas y experiencias en automatización de pruebas funcionales de integración en C#, hemos considerado que una forma pragmática de lograr dicho fin, es recabar esas prácticas y experiencias mediante el análisis de los resultados de la manipulación de las variables del experimento.

En los siguientes apartados, se puede visualizar los resultados obtenidos por framework y una comparación de ambos esto con el objetivo de detallar a grosso modo el comportamiento obtenido de la manipulación de variables en el experimento.

3.1. Resultados ejecución de scripts bajo el framework Record and Playback

Al definir la variable independiente como Record and Playback, se logró documentar su impacto en las variables dependientes las cuales son eficiencia en la ejecución, mantenimiento del script y cantidad de errores encontrados.

Entrando más en detalle, para la medición de la variable *eficiencia en la ejecución* se documentaron los tiempos obtenidos de cada corrida en la integración continua esto con el fin de lograr una comparación fundamentada entre frameworks. En la Tabla 4 se pueden observar los tiempos que se obtuvieron de la ejecución de los scripts desarrollados.

Tabla 4 Tiempo ejecución Scripts framework Record and Playback

Día	TE Record&Playback
9/15/2020	0:03:18
9/16/2020	0:02:40
9/17/2020	0:03:06
9/18/2020	0:02:38
9/21/2020	0:02:00

9/22/2020	0:01:00
9/23/2020	0:04:04
9/24/2020	0:01:50
9/25/2020	0:01:45
9/28/2020	0:01:50
9/29/2020	0:01:43
9/30/2020	0:01:34
10/1/2020	0:01:31
10/2/2020	0:01:35
10/5/2020	0:01:40

Fuente: Elaboración propia

En términos de la segunda variable dependiente definida para el experimento *Mantenimiento del Script,* a lo largo de los 15 días que se ejecutaron se realizó mantenimiento correctivo a todos aquellos que fallaban en la ejecución continua por un lapso de 6 días. Este mantenimiento se enfoca en corregir aquellos defectos que se encuentran en el software.

En cuanto a términos de este framework, se puede resaltar que posee una gran dificultad a la hora de dar el mantenimiento. En otras palabras, cada vez que se requiera alguna modificación o corrección en el script incluso cambios menores va a necesitar que toda la prueba sea actualizada o grabada nuevamente. Por lo que, mantener los scripts de este framework puede llegar a tener un costo mayor que el valor inicial de realizarlos. Por otra parte, en lo que concierne a la tercera variable definida como *Cantidad de errores encontrados,* se puede recalcar que los scripts lograron encontrar todos los errores esperados en la aplicación. En la Tabla 5 se pueden observar la cantidad de errores encontrados en la ejecución de los scripts desarrollados bajo el framework record and Playback, por un lapso de 15 días.

Tabla 5 Cantidad de errores encontrados por Record and Playback.

Día	Cantidad de errores encontrados Record and Playback
9/15/2020	4
9/16/2020	4
9/17/2020	4
9/18/2020	4
9/21/2020	4
9/22/2020	4
9/23/2020	4
9/24/2020	4
9/25/2020	4
9/28/2020	4
9/29/2020	4
9/30/2020	4
10/1/2020	4
10/2/2020	4
10/5/2020	4

Fuente: Elaboración propia

3.2. Resultados ejecución de scripts bajo el framework Test Library Architecture

Si se define la variable independiente como Test Library Architecture, como se menciona con anterioridad se logra documentar su impacto en las variables dependientes las cuales son eficiencia en la ejecución, mantenimiento del script y cantidad de errores encontrados.

Para la medición de la variable *eficiencia en la ejecución* se documentaron los tiempos obtenidos de cada corrida en ejecución continua. En la Tabla 6 se pueden observar los tiempos obtenidos de la ejecución de los scripts desarrollados bajo este framework, por un lapso de 15 días.

Tabla 6 Tiempo ejecución Scripts framework Test Library Architecture

Día	TE Test Library Architecture
9/15/2020	0:02:40
9/16/2020	0:03:42
9/17/2020	0:04:18
9/18/2020	0:02:53
9/21/2020	0:02:00
9/22/2020	0:02:00
9/23/2020	0:04:35
9/24/2020	0:02:07
9/25/2020	0:02:20
9/28/2020	0:02:07
9/29/2020	0:02:18
9/30/2020	0:02:04
10/1/2020	0:02:10
10/2/2020	0:02:01
10/5/2020	0:01:59

Fuente: Elaboración propia

Posteriormente se realizó el análisis de la segunda variable dependiente *Mantenimiento del Script*, a lo largo de los 15 días que se ejecutaron se realizó mantenimiento correctivo a todos aquellos que fallaban en la ejecución continua por un lapso de 4 días. Además, estos cambios fueron efectuados para corregir los errores que se presentaron en el software y la estabilidad.

En cuanto términos del framework, cabe resaltar que el mantenimiento es sumamente sencillo esto si se tiene conocimiento en programación. Por tal motivo, se puede resaltar que la dificultad en el mantenimiento no es alta, ya que a diferencia del framework Record and Playback se puede interactuar con el código sin tener que repetir o grabar las pruebas. Por otro lado, en lo que se refiere la tercer variables dependiente denominada *Cantidad de errores encontrados*, se puede determinar que los scripts desarrollados en este framework lograron encontrar los errores esperados del sistema. En la Tabla 7 se pueden observar la cantidad de errores encontrados en la ejecución de los scripts desarrollados bajo el framework record and Playback, por un lapso de 15 días

Tabla 7 Cantidad de errores encontrados por Test Library Architecture.

Día	Cantidad de errores encontrados Test Library Architecture
9/15/2020	4
9/16/2020	4
9/17/2020	4
9/18/2020	4
9/21/2020	4
9/22/2020	4
9/23/2020	4
9/24/2020	4
9/25/2020	4
9/28/2020	4
9/29/2020	4
9/30/2020	4
10/1/2020	4
10/2/2020	4
10/5/2020	4

Fuente: Elaboración propia

3.3. Comparación de frameworks a partir de las variables dependientes

Si se profundiza en los resultados obtenidos de la interacción de las variables durante la ejecución del experimento, es importante realizar una comparación objetiva y argumentada con el fin de plasmar por medio de experiencias cuál framework obtuvo mejores resultados.

En cuanto a términos de la eficiencia en la ejecución de ambos frameworks, si se toman los datos de las tablas 4 y 5 para realizar un promedio de duración por framework se obtienen los siguientes resultados: 0:02:09 para el framework Record and Playback y 0:02:37 para el framework Test library architecture. Demostrando así que el framework Record and Playback es más eficiente en términos de tiempo de ejecución. Asimismo, a

continuación, en la ilustración 2 se puede observar una gráfica del comportamiento de los tiempos de ejecución en la integración continua.

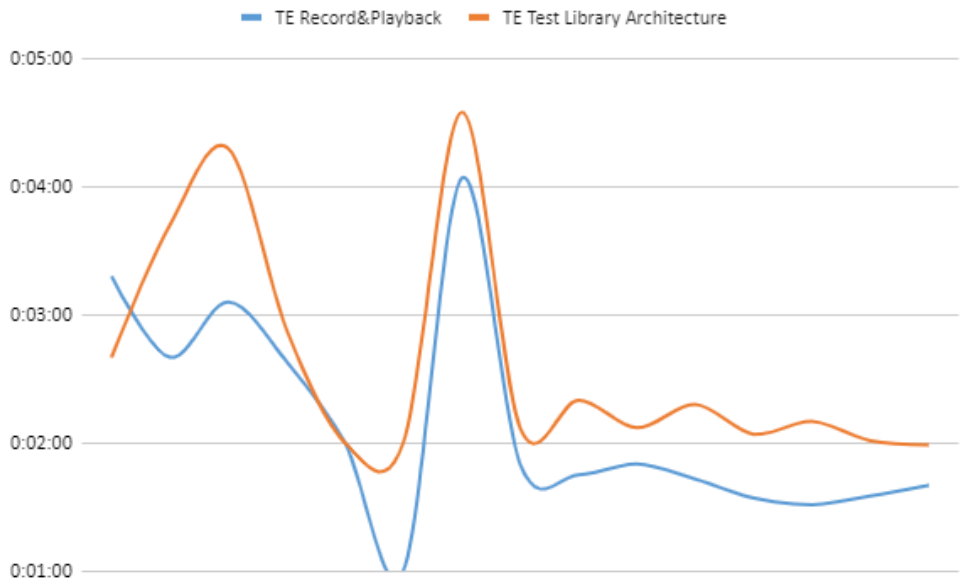

Ilustración 2 Gráfico tiempos de ejecución (Elaboración propia)

Por otro lado, en cuanto a la comparación del mantenimiento de los scripts bajo ambos frameworks, se observa que existe una gran diferencia en términos de dificultad, costo y tiempo. Puesto que, dar mantenimiento a los scripts del framework Record and Playback en la mayoría de los casos significa que se debe realizar la grabación de los scripts desde cero. Por lo que, se puede considerar inviable en proyectos de automatización grandes debido al enorme retrabajo.

Al contrario, el mantenimiento de los scripts del framework Test Library Architecture es mucho más sencillo si se tiene conocimientos previos de programación. Ya que, se pueden solucionar los problemas de una manera más específica sin tener que realizar el script nuevamente. Esto permite tener una reducción en tiempo, costo y esfuerzo que son mucho más altos en el framework Record and Playback.

En términos generales, para el framework Test Library Architecture fue necesario realizar mantenimiento correctivo por un lapso de 4 días, y para el framework Record and Playback por un lapso de 6 días. En la ilustración 3, se puede encontrar un gráfico de

porcentajes en el cual se plasma la comparación de la cantidad de días en los cuales se realizó mantenimiento en ambos frameworks. Dando esto como resultado que el framework que necesito más mantenimiento fue el Record and playback.

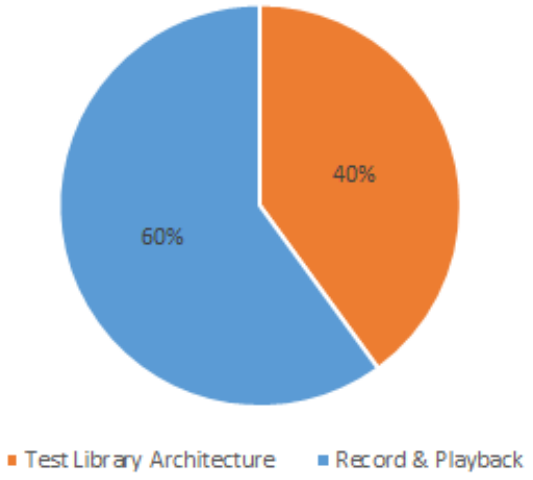

Ilustración 3 Gráfico porcentajes de mantenimiento (Elaboración propia)

Asimismo, si se comparan la cantidad de errores encontrados por ambos frameworks se puede determinar que ambos son igual de eficaces. Ya que, los dos lograron encontrar los errores esperados del sistema. Por ende, esta variable no es un factor determinante a la hora de determinar cuál es el mejor framework. En la ilustración 4 se pueden observar la cantidad de errores encontrados en la ejecución de los scripts desarrollados bajo cada uno de los frameworks utilizados, por un lapso de 15 días.

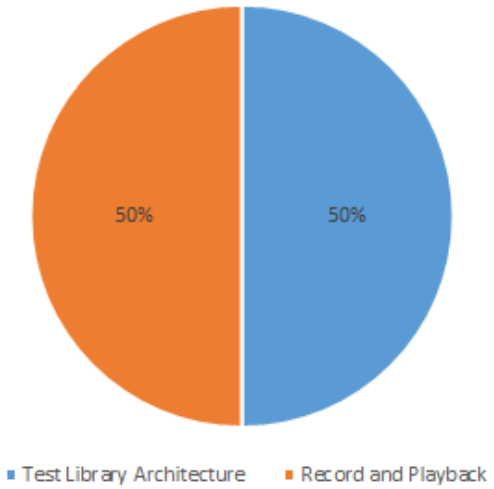

Ilustración 4 Gráfico porcentajes de cantidad de errores encontrados (Elaboración propia)

De este modo, si se analizan las tres variables dependientes se puede determinar a nivel personal por medio de experiencias que el framework Test Library Architecture es el más recomendado si se desean realizar proyectos de automatización. El principal motivo para llegar a esta conclusión fue la variable de mantenimiento, ya que como se menciona anteriormente este framework posee un bajo costo y dificultad en este ámbito. Esto gracias a que, la estructura se basa en la reutilización de código lo que genera grandes ventajas como lo son la reducción de costos, la reducción de tiempos, la reducción de esfuerzos, entre otros.

A pesar de que, en los tiempos de ejecución el framework Record and Playback obtuvo un mejor promedio la diferencia fue mínima. Por lo que, el resultado final de esta variable no es tan determinante en esta decisión. Al igual, que la variable de cantidad de errores encontrados ya que se obtuvieron los mismos resultados.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

1. Conclusiones

La presente conclusión será presentada siguiendo la línea de los objetivos específicos definidos con anterioridad en este documento. A continuación, se procederá a concluir sobre cada uno.

En cuanto al primer objetivo específico, las fuentes bibliográficas utilizadas en este TFG fueron de la más alta calidad. Se utilizaron 28 referencias bibliográficas. El 64% (18) fueron fuentes primarias y el 36% (10) fuentes secundarias. El medio de publicación utilizado fue de revistas en un 11% (3), referencias de libros un 28% (8), tesis reconocidas un 25% (7), consistentes en una referencia por cada uno de estos medios de publicación. El análisis de la antigüedad de estas referencias bibliográficas indica que el 11% (3) corresponden al periodo del año 2000 al 2005, de igual forma un 14% (4) fueron publicados entre el 2006 y el 2010, posteriormente se utilizaron el 25% (7) que comprende de los años 2011 al 2015 y finalmente, para el periodo comprendido entre el 2016 y el 2019, año de publicación del artículo, se utilizaron un 43% (12) de referencias bibliográficas.

Respecto al segundo objetivo específico, una vez analizada la literatura a través de las diversas fuentes mencionadas anteriormente, se procedio al diseñó de un experimento de laboratorio en el cual fueron incluidos los elementos esenciales para el desarrollo de este TFG. Estos aspectos fueron la especificación y programación de 10 scripts por cada uno de los frameworks utilizados y la definición de los protocolos y guías de gestión de datos a utilizar.

Referente al tercer objetivo específico, durante el desarrollo de este TFG, se puso en marcha el proceso de ejecución de los 20 scripts diseñados durante un periodo de 15 días. Esto con el fin de recolectar la información necesaria para comparar los resultados obtenidos entre ambos frameworks, con el fin de establecer y efectuar un análisis que será determinante para este TFG que servirá como puente para la recolección de datos y poder

medir las variables dependientes. Todo esto con el principal objetivo de generar prácticas y experiencias.

En relación al cuarto y último objetivo específico, a partir de la información recolectada durante el desarrollo del experimento se pueden resaltar las siguientes prácticas y experiencias. La detección de errores en las etapas tempranas de desarrollo, puede brindar un gran ahorro en cuestiones de tiempo y dinero. Por lo que, como se menciona anteriormente la implementación de pruebas automatizadas de software son un elemento de gran ayuda para agilizar los procesos en las empresas.

En el ámbito de la automatización existen varios tipos de frameworks, los cuales son estándares y reglas que van a facilitar la creación y ejecución de script automatizados. En el presente experimento se trabajaron dos tipos de framework el Record and Playback y el Test Library Architecture. Después de realizar todo el proceso de diseño, desarrollo y puesta en ejecución de los scripts desarrollados bajo ambos frameworks se pueden resaltar las siguientes ventajas y desventajas de cada uno.

Con respecto al framework Record and Playback, se determinaron las siguientes ventajas: es fácil de utilizar y su curva de aprendizaje es baja o nula, es la manera más rápida para generar scripts automatizados y los scripts pueden ser exportados a varios lenguajes de programación. Sin embargo, posee algunas desventajas ya que tiene un alto costo de mantenimiento, esto debido a que si se debe cambiar un paso del script se debe grabar todo nuevamente. Además, existe una cobertura limitada, ya que comúnmente las aplicaciones dedicadas a grabar casos de prueba solo poseen las funcionalidades básicas de prueba, por lo que los scripts siguen un poco más de la funcionalidad básica de la aplicación. Asimismo, las aplicaciones utilizadas para realizar scripts por lo general tienen funcionalidades limitadas. Por último, a la hora de grabar hay que tener mucho cuidado con las acciones que se realizan para no hacer pasos innecesarios, esto debido a que las herramientas registran toda acción que se realice sin excepción.

En relación al framework Test Library Architecture, se determinaron las siguientes ventajas: existe una alta reutilización de código, lo que permite agilizar la creación de scripts y el mantenimiento de los scripts es fácil de realizar, gracias al alto nivel de modularización.

Sin embargo, se encontraron las siguientes desventajas: se necesita un alto nivel de conocimiento técnico para poder elaborar los scripts, se toma más tiempo la realización de estos scripts, ya que, deben ser elaborados desde cero y al estar los datos incrustados en los scripts si se debe realizar algún cambio en estos se deben modificar los scripts.

Si se realiza una comparación de ambos frameworks basada en la experiencia presentada durante el desarrollo y ejecución del experimento se pueden resaltar varios puntos de interés. Primeramente, en la programación de los scripts del framework Record and Playback se evidenció la facilidad en el uso de la herramienta utilizada en este caso Selenium WebDriver, para la cual no hay que tener ningún conocimiento profundo en automatización de pruebas. Al contrario, en el framework Test Library Architecture se debió realizar un estudio a profundidad con el fin de lograr programar ciertas funciones dentro de los scripts.

Por otra parte, en términos de ejecución se evidenció durante la etapa de aplicación del experimento que existe muy poca diferencia entre los tiempos de ejecución de ambos frameworks, estos tiempos se encuentran plasmados en la sección de Validación de la Propuesta en este documento; la diferencia entre ambos fue de 28 segundos.

Además, en cuanto al mantenimiento de los scripts se evidenciaron grandes diferencias en términos de dificultad, costo y tiempo entre ambos frameworks. Ya que, por un lado, el framework Record and Playback tiene un alto costo de mantenimiento debido a la gran cantidad de retrabajo que se debe realizar si se debe realizar algún cambio o actualización en los scripts.

Pero, por otro lado, en el framework Test Library Architecture si se poseen conocimientos de programación el mantenimiento es sumamente sencillo ya que cualquier cambio solo se realiza a la línea específica de código sin tener que volver a hacer todo desde cero. Además, al utilizar una librería de funcionalidades comunes se pueden hacer cambios directos sin tener que modificar script por script.

En conclusión, si se analizan todas las variables y factores desarrollados y ejecutados durante el experimento se puede recomendar a todos los profesionales del área de calidad

la utilización del framework Test Library Architecture. Ya que, si bien el framework Record and Playback posee cualidades favorables, vale la pena invertir en el desarrollo de scripts automatizados robustos que puedan asegurar por completo la calidad de los productos de software, además, de permitir la reducción de costos y tiempos a un largo plazo.

2. Recomendaciones

Debido al amplio alcance que puede tener el proceso de automatización de pruebas, siempre se desea que exista una mejora continua. Por lo que, es importante dar las siguientes recomendaciones siempre dentro del alcance aprobado con este TFG. Se recomienda realizar un análisis previo de la estructura y casos de prueba de la aplicación, así como identificar que tipo de pruebas se desean realizar. Esto con el objetivo de determinar el alcance y además verificar si la aplicación es automatizable, ya que, como se mencionó con anterioridad no toda prueba se puede automatizar.

Se recomienda realizar un análisis previo de las herramientas existentes para automatización de acuerdo al framework a utilizar. Esto con el objetivo de determinar cúal es la herramienta adecuada para el tipo de pruebas que se desean realizar y así se ajuste de manera adecuada a los objetivos propuestos.

Se recomienda utilizar un agente iterativo, esto para que sea más práctico a la hora de ejecutar las pruebas automatizadas, y que se tenga la posibilidad de visualizar el proceso cuando estos se ejecutan, en cambio si se utiliza un agente no iterativo este se ejecuta en segundo plano, por lo que, no se podría observar lo que se está realizando y tiende hacer más inestable.

Se recomienda utilizar el framework Test Library Architecture ya que como se demuestra a lo largo de este trabajo y los resultados obtenidos este framework es más efectivo para realizar las pruebas automatizadas por su bajo costo de mantenimiento. Por otro lado, si lo que se pretende es tener mayores resultados que los vistos con este experimento se recomienda ampliar la cantidad de frameworks para realizar más comparaciones y obtener diferentes resultados para así llegar a una conclusión más amplia.

3. Limitaciones

Durante la elaboración y ejecución del TFG se presentaron ciertas limitaciones. Una de estas fue la necesidad de utilizar una máquina virtual en Azure para instalar las herramientas que eran necesarias para la elaboración de este proyecto, esto con el fin de hacer uso de la nube para tener un respaldado del proyecto y para poder hacer la ejecución los scripts automatizados. No obstante, al no tener una cuenta activa en Azure se tuvo que optar por utilizar la máquina física, si bien esto no es un problema a la hora de instalar las herramientas, se corre el riesgo que este deje de funcionar y así perder todo el desarrollo del experimento.

Por otro lado, otra limitación fue la forma en que se ejecutan los scripts automatizados. Al inicio del experimento, gracias a la forma que se había configurado la integración continua y los despliegues continuos, se presentaban ciertos errores de intermitencia lo que provocaba que los scripts fallaran continuamente. Esto debido a que se instaló un agente que se ejecutaba en segundo plano y no se podía observar el proceso, ni detallar el problema que se presentaba. Sin embargo, para la resolución de este inconveniente se procedió a cambiar el método de ejecución, por lo que, se instaló un agente iterativo que tiene como funcionalidad principal levantar el navegador web y realizar el proceso de las corridas en vivo resolviendo el error de la intermitencia.

4. Trabajos futuros

Como continuación de este TFG, es importante tener como referente la línea de trabajo que se ha seguido a lo largo de esta investigación, con el fin de dar continuidad al esfuerzo invertido. Por lo que, el principal aspecto a tomar en cuenta para el proseguir de esta investigación son los frameworks de automatización utilizados como variable independiente para la creación de los scripts en el experimento planteado.

Como se menciona a lo largo de este trabajo se utilizaron dos frameworks de automatización, el "*Record and Playback*" y "*Test Library Architecture*". Actualmente, existe una gran variedad de frameworks que pueden ser utilizados con el fin de ampliar los resultados de este experimento. Algunos de estos frameworks son: "Module Based Testing"

el cual se encarga de dividir la aplicación en módulos, “Data Driver” que se basa en datos almacenados en tablas u hojas de cálculos, “Keyboard Driver” basado en palabras claves e “Hybrid Testing” que es una combinación de todos los framework que se mencionan anteriormente.

Por lo que, es importante como trabajo futuro extender la cantidad de frameworks a utilizar como variables independientes, con el objetivo de poder determinar con un mayor respaldo cuál framework es el más efectivo y recomendable para utilizar. Y de esta manera, gracias a la ampliación de la cantidad de frameworks a utilizar se puede generar un análisis más exhaustivo por medio de más experiencias, gracias a los distintos hallazgos y resultados que se pueden obtener.

REFERENCIAS

REFERENCIAS

Aristegui, J. (2010). Los casos de prueba en la prueba de software. *Revista Digital Lámpsakos*, No. 3, pp. 27-34.

Artega, A. (2013). Automatización de conveyors para líneas de producción con apoyo de RS LOGIX 500. Recuperado de http://132.248.9.195/ptd2013/agosto/0699828/0699828.pdf

Cabrera, M. y Lara, F. (2015). Fichas de procedimientos de evaluación educativa UDLA.

Cárdenas, O. (2016). *Automatización de casos de prueba para mejorar el proceso de calidad de software* (Tesis de Licenciatura). Fundación Universitaria Los Libertadores, Colombia. Recuperado de https://repository.libertadores.edu.co/bitstream/handle/11371/738/C%c3%a1rdenasB%c3%a1ezOlgaLucia.pdf?sequence=2&isAllowed=y

Cubas, R. 2017. *Testing y calidad de software. Automatización de pruebas con Selenium WebDriver* (Trabajo Fin de Grado). E.T.S.I. y Sistemas de Telecomunicación (UPM), Madrid. Recuperado de http://oa.upm.es/49320/1/PFC_RAFAEL_CUBAS_MONTENEGRO.pdf

Franco, J. 2010. *Metodología para testing de software basado en componentes* (Trabajo Fin de Grado). Universidad EAFIT, Medellín .Recuperado de https://core.ac.uk/download/pdf/47237302.pdf

Gómez, G. (2015). *Las Pruebas de Integración como Proceso de la Calidad del Software en el Ámbito de las Telecomunicaciones* (Tesis fin de carrera). Universidad Carlos III. Madrid, España.

Grau, X. (2000). Principios Básicos de Usabilidad para Ingenieros Software. JISBD, 39-46.

Hayes, L. (2004). The automated testing handbook. Software Testing Institute.

Kidd, C. (2018). Beginner's Guide to Test Automation Frameworks [Blog]. Recuperado de https://www.bmc.com/blogs/test-automation-frameworks/

Microsoft. (s.f.). ¿Qué es DevOps? Explicación de DevOps | Microsoft Azure. Recuperado de https://azure.microsoft.com/es-es/overview/what-is-devops/

Microsoft. (s.f.). ¿Qué es una máquina virtual?. Recuperado de https://azure.microsoft.com/es-es/overview/what-is-a-virtual-machine/

Minhaz, S. (2016). *University Management System* [Software]. Bangladesh: Noakhali Science & Technology University. Recuperado de https://github.com/smminhaz/University-Management-system

Müller, T et al. (2011) *Probador Certificado del ISTQB: Programa de Estudio de Nivel Básico*. Bélgica: International Software Testing Qualifications Board.

Mubarak, U y Zhanfang, C. (2019). A Study of Automated Software Testing: Automation Tools and Frameworks. International Journal of Computer Science Engineering, 217-225. doi: 10.5281/zenodo.3924795

Muradas, Y. (2018). Conoce las 3 metodologías ágiles más usadas [Blog]. Recuperado de https://openwebinars.net/blog/conoce-las-3-metodologias-agiles-mas-usadas/

Olsen, K et al. (2018) *Probador Certificado del ISTQB: Programa de Estudio de Nivel Básico*. Bélgica: International Software Testing Qualifications Board.

Pantaleo, G. (2011). *Calidad en el desarrollo de software*. Paraguay: Alfaomega Grupo Editor Argentino S.A.

Peres, H. (2018). *Automatización de tests de software con Selenium*. Rio de Janeiro: Simplísimo Livros Ltda.

Pérez, J. (2006). *Clasificación de usuarios basada en la detección de errores usando técnicas de procesadores de lenguaje* (Doctorado). Universidad de Oviedo.

Pressman, R. (2010). *Ingeniería del software* (7th ed.). New York: The McGraw-Hill Companies, Inc. Recuperado de http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

Rivera, C. (2018). *Automatización de pruebas de regresión* (Tesis de maestría). Universidad de Chile. Recuperado de http://repositorio.uchile.cl/bitstream/handle/2250/165608/Automatizaci%C3%B3n-de-pruebas-de-regresi%C3%B3n.pdf?sequence=1&isAllowed=y

Rodríguez, C. (2019). Automatizar pruebas de software: ¿cuándo y por qué? [Blog]. Recuperado de https://cl.abstracta.us/blog/automatizar-pruebas-de-software/

Sajjad, S. (2016). *Basic Guidelines for Research: An Introductory Approach for All Disciplines*. Bangladesh: Book Zone Publication.

Sommerville, I. (2005). *Ingeniería del Software* (7th ed.). Madrid: Pearson Educación.

Toledo, F. (2014). *Introducción a las pruebas de sistemas de información*. Montevideo: Abstracta.

Vinasco, J.; Roca, M. y Pardo, C. (2013). Integración de Pruebas Automáticas para la Optimización de los Procesos de Producción de Software en un Estudio de Caso Real.

Williamson, K., y Johanson, G. (2018). *Research methods: information, systems, and contexts*. (2nd ed.). Chandos Publishing.

ANEXOS

Anexo 1. Formato Lista de Cotejo

Lista de Cotejo
Fecha: (DD/MM/AAAA)

Criterio	Cumple	No cumple	Observaciones
Todos los scripts se ejecutaron sin errores.			
Los scripts permiten fácilmente que una determinada modificación sea implementada.			
Los scripts se encuentran estables, no se necesita mantenimiento.			
Los scripts cumplen con los requisitos y pasos establecidos en la programación.			
Los scripts se ejecutan de manera continua.			
Se recolectó la información requerida después de la ejecución de los scripts.			

Conclusión

Fuente: Elaboración propia

Anexo 2. Formato Diario de Campo

DIARIO DE CAMPO	
Fecha:	
Investigador/Observador	
Objetivo	
Técnica aplicada	
Descripción de actividades	**Hallazgos o resultados**
Observaciones	

Fuente: Elaboración propia

Anexo 3. Casos de prueba

ID: TC01 Crear departamento		**Prerrequisitos:**
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Save Department en el menú principal.	Se abre la página de creación de departamentos.
2	Darle click al botón Save sin llenar los campos del formulario.	El formulario no deja guardar el departamento y marca los campos de código y nombre en rojo.
3	Ingresar el código del departamento. Ej: 01.	El sistema deja ingresar datos en el campo del código. Si es de un carácter no deja salvar el departamento.

4	Ingresar el nombre del departamento. Ej: Ciencias	El sistema deja ingresar datos en el campo del nombre.
5	Click botón save	El departamento se creó exitosamente y un mensaje de validación fue desplegado (Saved Successfully).
6	Darle click a la opción View All Departments en el menú principal.	Se puede visualizar una tabla con todos los departamentos.
7	Verificar que se muestre el departamento recién creado.	

Fuente: Elaboración propia

ID: TC02 Ingresar código de departamento repetido		**Prerrequisitos:**
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Save Department en el menú principal	Se abre la página de creación de departamentos.
2	Ingresar el código del departamento Ej: 02	El sistema deja ingresar datos en el campo del código.
3	Ingresar el nombre del departamento. Ej: Astronomía.	El sistema deja ingresar datos en el campo del nombre.
4	Click botón save	El departamento se creó exitosamente y un mensaje de validación fue desplegado (Saved Successfully).
5	Darle click a la opción Save Department en el menú principal	Se abre la página de creación de departamentos.
6	Ingresar el mismo código del departamento ingresado en el paso 2	El sistema deja ingresar datos en el campo del código. Si es de un carácter no deja salvar el departamento.
7	Ingresar el nombre del departamento. Ej: Ciencias	El sistema deja ingresar datos en el campo del nombre.

8	Click botón save	Se despliega una validación de duplicidad de código (Department Code Already Exists. Department Code be unique). El departamento no se guarda.
9	Darle click a la opción View All Departments en el menú principal.	Se puede visualizar una tabla con todos los departamentos.
10	Verificar que no se muestre el departamento que se trata de crear	

Fuente: Elaboración propia

ID: TC03 Crear curso		**Prerrequisitos:**
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Save Department en el menú principal.	Se abre la página de creación de departamentos.
2	Ingresar el código del departamento Ej: 03.	El sistema deja ingresar datos en el campo del código.
3	Ingresar el nombre del departamento. Ej: Sociología.	El sistema deja ingresar datos en el campo del nombre.
4	Click botón save.	El departamento se creó exitosamente y un mensaje de validación fue desplegado (Saved Successfully).
5	Darle click a la opción Save Course en el menú principal.	Se abre la página de creación de cursos.
6	Darle click al botón Save sin llenar los campos del formulario.	El formulario no deja guardar el curso y marca los campos necesarios con validaciones en rojo.
7	Ingresar el código del curso. Ej: 01.	El sistema deja ingresar datos en el campo del code. Si es de menos de 6 caracteres no deja

		salvar el curso y tira una validación de campo.
8	Ingresar el nombre del curso. Ej: Química.	El sistema deja ingresar datos en el campo Name.
9	Ingresar la cantidad de créditos. Ej: 1.	El sistema deja ingresar datos en el campo Credit.
10	Ingresar la descripción del curso. Ej: Es un curso de química.	El sistema deja ingresar datos en el campo Description.
11	Seleccionar el departamento creado en el paso 1.	El sistema despliega un dropdown con los departamentos existentes. Y deja seleccionar 1 opción.
12	Seleccionar un semestre. Ej: 1st.	El sistema despliega un dropdown con los semestres existentes. Y deja seleccionar 1 opción.
13	Click botón save.	El departamento se creó exitosamente y un mensaje de validación es desplegado (Saved Successfully).
14	Darle click a la opción View All Courses Statics en el menú principal.	
15	Seleccionar departamento.	Se puede visualizar una tabla con todos los cursos.
16	Verificar que se muestre el curso recién creado.	

Fuente: Elaboración propia

ID: TC04: Crear profesor		**Prerrequisitos:**
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Save Department en el menú principal	Se abre la página de creación de departamentos.

2	Ingresar el código del departamento Ej: 04	El sistema deja ingresar datos en el campo del código.
3	Ingresar el nombre del departamento. Ej: Matemática	El sistema deja ingresar datos en el campo del nombre.
4	Click botón save	El departamento se creó exitosamente y un mensaje de validación fue desplegado (Saved Successfully).
5	Darle click a la opción Save Teacher en el menú principal.	Se abre la página de creación de profesores.
6	Darle click al botón Save sin llenar los campos del formulario	El formulario no deja guardar el curso y marca los campos necesarios con validaciones en rojo.
7	Ingresar el nombre del profesor. Ej: Luisa Jimenez	El sistema deja ingresar datos en el campo Name.
8	Ingresar la dirección. Ej: Heredia	El sistema deja ingresar datos en el campo Address.
9	Ingresar el email. Ej: Es un curso de luisa20@gmail.com	El sistema deja ingresar datos en el campo Email. Existe una validación del formato del correo.
10	Ingresar contacto. Ej: 88888888	El sistema deja ingresar datos en el campo Contact.
11	Seleccionar puesto Ej: professor	El sistema despliega un dropdown con los puestos existentes. Y deja seleccionar 1 opción.
12	Seleccionar el departamento creado en el paso 4.	Resultado Esperado: El sistema despliega un dropdown con los departamentos existentes.
13	Ingresar la cantidad de créditos. Ej: 1	El sistema deja ingresar datos en el campo Credit.. Y deja seleccionar 1 opción.
14	Click botón save	El departamento se creó exitosamente y un mensaje de

		validación fue desplegado (Saved Successfully).

Fuente: Elaboración propia

ID: TC05: Asignar curso a profesor		**Prerrequisitos:** Crear departamento, crear profesor,crear cursos.
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Course Assign to Teacher en el menú principal	Se abre la página de asignación de profesores.
2	Darle click al botón Assign sin llenar los campos del formulario	El formulario no deja guardar y marca los campos necesarios con validaciones en rojo.
3	Seleccionar un departamento.	El sistema despliega un dropdown con los departamentos existentes
4	Seleccionar un profesor.	El sistema despliega un dropdown con los profesores existentes. Los campos Credit to be Taken y Remaining Credit son llenados con los datos del profesor seleccionado y no son editables.
5	Seleccionar un curso.	El sistema despliega un dropdown con los códigos de los cursos existentes. Los campos Name y Credit son llenados con los datos del código seleccionado y no son editables.
6	Click botón assign	Resultado Esperado: El curso se asignó exitosamente a un profesor y un mensaje de validación es desplegado (Assigned Successfully).

Fuente: Elaboración propia

ID: TC06 Crear estudiante	**Prerrequisitos:**

Paso	Descripción	Resultado Esperado
1	Darle click a la opción Save Department en el menú principal.	Se abre la página de creación de departamentos.
2	Ingresar el código del departamento Ej: 05.	El sistema deja ingresar datos en el campo del código.
3	Ingresar el nombre del departamento. Ej: Ingeniería.	El sistema deja ingresar datos en el campo del nombre.
4	Click botón save.	El departamento se creó exitosamente y un mensaje de validación es desplegado (Saved Successfully).
5	Darle click a la opción Register Student en el menú principal.	Se abre la página de creación de estudiantes.
6	Darle click al botón Register sin llenar los campos del formulario.	El formulario no deja guardar el estudiante y marca los campos necesarios con validaciones en rojo.
7	Ingresar el nombre del estudiante. Ej: Fabian Jimenez.	El sistema deja ingresar datos en el campo Name.
8	Ingresar el email. Ej: Es un curso de fabianjim20@gmail.com.	El sistema deja ingresar datos en el campo Email. Existe una validación del formato del correo.
9	Ingresar contacto. Ej: 88888888.	El sistema deja ingresar datos en el campo Contact.
10	Click campo date.	El sistema abre el datepicker y permite seleccionar una fecha de creación.
11	Ingresar la dirección. Ej: Heredia	El sistema deja ingresar datos en el campo Address.
12	Seleccionar el departamento creado en el paso 4.	El sistema despliega un dropdown con los departamentos existentes.
13	Click botón register.	El departamento se creó exitosamente y un mensaje de

		validación es desplegado (Saved Successfully+student info).

Fuente: Elaboración propia

ID: TC07: Crear horario de clase		**Prerrequisitos:** TC01: Crear departamento, TC03: Crear curso
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Class Room Allocation>>Allocate Classrooms en el menú principal	Se abre la página de creación de estudiantes.
2	Darle click al botón Save sin llenar los campos del formulario	El formulario no deja guardar el horario y marca los campos necesarios con validaciones en rojo.
3	Seleccionar un departamento.	El sistema despliega un dropdown con los departamentos existentes.
4	Seleccionar un curso.	El sistema despliega un dropdown con los cursos existentes.
5	Seleccionar un room.	Resultado Esperado: El sistema despliega un dropdown con las salas de clase existentes.
6	Seleccionar un día.	El sistema despliega un dropdown con los días de la semana.
7	Seleccionar una hora inicial.	El sistema despliega un timepicker para seleccionar una hora.
8	Seleccionar una hora final.	El sistema despliega un timepicker para seleccionar una hora.

9	Click botón save.	El departamento se creó exitosamente y un mensaje de validación fue desplegado (Saved Successfully).
10	View Class Schedule en el menú principal.	
11	Seleccionar departamento.	Se puede visualizar una tabla con todos los horarios existentes para el departamento seleccionado.
12	Verificar que se muestre el curso recién creado.	

Fuente: Elaboración propia

ID: TC08 Registrar estudiante en un curso		**Prerrequisitos:** Crear departamento, Crear curso, Crear estudiante.
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Student >> Enroll Student in A Course en el menú principal.	Se abre la página de asignación de cursos a estudiantes.
2	Darle click al botón Enroll sin llenar los campos del formulario.	El formulario no deja guardar y marca los campos necesarios con validaciones en rojo.
3	Seleccionar Registration No del estudiante.	El sistema despliega un dropdown con los ids de los estudiantes existentes. Los campos Name, Email y Department se rellenan y no son editables.
4	Seleccionar un curso.	El sistema despliega un dropdown con los cursos existentes
5	Seleccionar un room.	El sistema despliega un dropdown con las salas de clase existentes.

6	Seleccionar una fecha.	El sistema despliega un datepicker con los días de la semana.
7	Click botón enroll.	El departamento se creó exitosamente y un mensaje de validación fue desplegado (Saved Successfully).

Fuente: Elaboración propia

ID: TC09 Añadir notas al estudiante		**Prerrequisitos:** Crear curso.
Paso	**Descripción**	**Resultado Esperado**
1	Darle click a la opción Student>> Save Student Result en el menú principal.	Se abre la página de asignación de nota a curso de estudiantes.
2	Darle click al botón Save sin llenar los campos del formulario.	El formulario no deja guardar y marca los campos necesarios con validaciones en rojo.
3	Seleccionar Student Reg. No del estudiante.	El sistema despliega un dropdown con los ids de los estudiantes existentes. Los campos Name, Email y Department se rellenan y no son editables.
4	Seleccionar un curso.	El sistema despliega un dropdown con los cursos existentes.
5	Seleccionar grade. Ej: A+	El sistema despliega un dropdown con las notas posibles a asignar.
6	Click botón save	La nota se asignó exitosamente y un mensaje de validación fue desplegado (Saved Successfully).

Fuente: Elaboración propia

ID: TC10: Ver Notas del estudiante	**Prerrequisitos:** Crear Estudiante, Añadir notas al estudiante.

Paso	Descripción	Resultado Esperado
1	Darle click a la opción Student>> View Result en el menú principal.	Se abre la página de búsqueda de notas de estudiantes.
2	Seleccionar Student Reg. No del estudiante.	El sistema despliega un dropdown con los ids de los estudiantes existentes. Los campos Name, Email y Department se rellenan y no son editables. Y se muestran los resultados en una tabla.
3	Close.	

Fuente: Elaboración propia

Anexo 4. Scripts automatizados del framework Record and Playback

Para más detalle, el proyecto que contiene los scripts del framework Record and Playback se pueden encontrar en el siguiente link de Github https://github.com/TFGH-F/Record_Playback.

TC01 Crear departamento

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
```

```
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;

[TestFixture]
public class TC01CreardepartamentoTest {
  private IWebDriver driver;
    public IDictionary<string, object> vars {get; private set;}
    private IJavaScriptExecutor js;
    [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC01Creardepartamento() {

       driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
    driver.FindElement(By.LinkText("Department")).Click();
    driver.FindElement(By.LinkText("Save Department")).Click();
    driver.FindElement(By.Id("Code")).Click();
    driver.FindElement(By.Id("Code")).SendKeys("01");
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Ciencias");
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("form > p")).Text,
Is.EqualTo("Saved Successfully"));
    driver.FindElement(By.LinkText("Department")).Click();
```

```
            driver.FindElement(By.LinkText("View All Departments")).Click();
            driver.FindElement(By.LinkText("2")).Click();
            Assert.That(driver.FindElement(By.CssSelector(".grid-row:nth-child(2)
> .grid-cell:nth-child(1)")).Text, Is.EqualTo("01"));
            Assert.That(driver.FindElement(By.CssSelector(".grid-row:nth-child(2)
> .grid-cell:nth-child(2)")).Text, Is.EqualTo("Ciencias"));
            driver.Close();
            }
        }
```

TC02 Ingresar código de departamento repetido

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC02IngresarcdigodedepartamentorepetidoTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC02Ingresarcdigodedepartamentorepetido() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
        driver.FindElement(By.LinkText("Department")).Click();
```

```
    driver.FindElement(By.LinkText("Save Department")).Click();
    driver.FindElement(By.Id("Code")).Click();
    driver.FindElement(By.Id("Code")).SendKeys("02");
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Astronomia");
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("form > p")).Text,
Is.EqualTo("Saved Successfully"));
    driver.FindElement(By.LinkText("Department")).Click();
    driver.FindElement(By.LinkText("Save Department")).Click();
    driver.FindElement(By.Id("Code")).Click();
    driver.FindElement(By.Id("Code")).SendKeys("02");
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Estudios");
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("form > p")).Text,
Is.EqualTo("Department Code Already Exists. Department Codebe be unique"));
    driver.FindElement(By.LinkText("Department")).Click();
    driver.FindElement(By.LinkText("View All Departments")).Click();
    driver.FindElement(By.LinkText("2")).Click();
    Assert.That(driver.FindElement(By.CssSelector(".grid-row:nth-child(3) >
.grid-cell:nth-child(1)")).Text, Is.EqualTo("02"));
    Assert.That(driver.FindElement(By.CssSelector(".grid-row:nth-child(3) >
.grid-cell:nth-child(2)")).Text, Is.EqualTo("Astronomia"));
    driver.Close();
  }
}
```

TC03 Crear curso

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC03CrearcursoTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
```

```
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC03Crearcurso() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
    driver.FindElement(By.LinkText("Department")).Click();
    driver.FindElement(By.LinkText("Save Department")).Click();
    driver.FindElement(By.Id("Code")).Click();
    driver.FindElement(By.Id("Code")).SendKeys("31");
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Química");
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("form > p")).Text,
Is.EqualTo("Saved Successfully"));
    driver.FindElement(By.LinkText("Course")).Click();
    driver.FindElement(By.LinkText("Save Course")).Click();
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.Id("Code-error")).Text, Is.EqualTo("Please
enter a valid Course Code!"));
    Assert.That(driver.FindElement(By.Id("Name-error")).Text, Is.EqualTo("Please
enter the valid Course Name!"));
    Assert.That(driver.FindElement(By.Id("Credit-error")).Text,
Is.EqualTo("Credit is required!"));
    Assert.That(driver.FindElement(By.Id("Description-error")).Text,
Is.EqualTo("Write something about course!"));
    Assert.That(driver.FindElement(By.Id("DepartmentId-error")).Text,
Is.EqualTo("Please select related department!"));
    Assert.That(driver.FindElement(By.Id("SemesterId-error")).Text,
Is.EqualTo("Please select related semester!"));
    driver.FindElement(By.Id("Code")).Click();
    driver.FindElement(By.Id("Code")).SendKeys("564378");
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Quimica avanzado");
    driver.FindElement(By.Id("Credit")).Click();
    driver.FindElement(By.Id("Credit")).SendKeys("4");
    driver.FindElement(By.Id("Description")).Click();
    driver.FindElement(By.Id("Description")).SendKeys("Este curso pertenece a
quimica");
    driver.FindElement(By.Id("DepartmentId")).Click();
```

```
    {
      var dropdown = driver.FindElement(By.Id("DepartmentId"));
      dropdown.FindElement(By.XPath("//option[. = 'Química']")).Click();
    }
    driver.FindElement(By.Id("DepartmentId")).Click();
    driver.FindElement(By.Id("SemesterId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("SemesterId"));
      dropdown.FindElement(By.XPath("//option[. = '1st']")).Click();
    }
    driver.FindElement(By.Id("SemesterId")).Click();
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("p:nth-child(4)")).Text,
Is.EqualTo("Saved Successfully"));
    driver.FindElement(By.LinkText("Course")).Click();
    driver.FindElement(By.LinkText("View Course Statics")).Click();
    driver.FindElement(By.Id("DepartmentId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DepartmentId"));
      dropdown.FindElement(By.XPath("//option[. = 'Química']")).Click();
    }
    driver.FindElement(By.Id("DepartmentId")).Click();
    Assert.That(driver.FindElement(By.CssSelector("td:nth-child(1)")).Text,
Is.EqualTo("564378"));
    Assert.That(driver.FindElement(By.CssSelector("td:nth-child(2)")).Text,
Is.EqualTo("Quimica avanzado"));
    Assert.That(driver.FindElement(By.CssSelector("td:nth-child(3)")).Text,
Is.EqualTo("1st"));
    Assert.That(driver.FindElement(By.CssSelector("td:nth-child(4)")).Text,
Is.EqualTo("Not Assigned yet"));
    driver.Close();
  }
}
```

TC04: Crear profesor

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
```

```
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC04CrearprofesorTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC04Crearprofesor() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
    driver.FindElement(By.LinkText("Department")).Click();
    driver.FindElement(By.LinkText("Save Department")).Click();
    driver.FindElement(By.Id("Code")).Click();
    driver.FindElement(By.Id("Code")).SendKeys("04");
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Matematica");
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("form > p")).Text,
Is.EqualTo("Saved Successfully"));
    driver.FindElement(By.LinkText("Teacher")).Click();
    driver.FindElement(By.LinkText("Save Teacher")).Click();
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.Id("Name-error")).Text, Is.EqualTo("Please
enter the Name"));
    Assert.That(driver.FindElement(By.Id("Address-error")).Text,
Is.EqualTo("Address is required!"));
    Assert.That(driver.FindElement(By.Id("Email-error")).Text,
Is.EqualTo("Please email address is required"));
    Assert.That(driver.FindElement(By.Id("Contact-error")).Text,
Is.EqualTo("Please enter the contact"));
    Assert.That(driver.FindElement(By.Id("DesignationId-error")).Text,
Is.EqualTo("Designation is required!"));
    Assert.That(driver.FindElement(By.Id("DepartmentId-error")).Text,
Is.EqualTo("Please select department"));
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Luisa Jimenez");
    driver.FindElement(By.Id("Address")).Click();
```

```
    driver.FindElement(By.Id("Address")).SendKeys("Heredia");
    driver.FindElement(By.Id("Email")).Click();
    driver.FindElement(By.Id("Email")).SendKeys("luisa20@gmail.com");
    driver.FindElement(By.Id("Contact")).Click();
    driver.FindElement(By.Id("Contact")).SendKeys("88888888");
    driver.FindElement(By.Id("DesignationId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DesignationId"));
      dropdown.FindElement(By.XPath("//option[. = 'Professor']")).Click();
    }
    driver.FindElement(By.Id("DepartmentId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DepartmentId"));
      dropdown.FindElement(By.XPath("//option[. = 'Matematica']")).Click();
    }
    driver.FindElement(By.Id("CreditTobeTaken")).Click();
    driver.FindElement(By.Id("CreditTobeTaken")).SendKeys("1");
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("p:nth-child(4)")).Text,
Is.EqualTo("Saved Sucessfully"));
    driver.Close();
  }
}
```

TC05: Asignar curso a profesor

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC05AsignarcursoaprofesorTest {
    private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
```

```
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC05Asignarcursoaprofesor() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
    driver.FindElement(By.LinkText("Teacher")).Click();
    driver.FindElement(By.LinkText("Assign Course To Teacher")).Click();
    driver.FindElement(By.Id("submitButton")).Click();
    {
      var element = driver.FindElement(By.Id("submitButton"));
      Actions builder = new Actions(driver);
      builder.MoveToElement(element).Perform();
    }
    {
      var element = driver.FindElement(By.TagName("body"));
      Actions builder = new Actions(driver);
      builder.MoveToElement(element, 0, 0).Perform();
    }
    Assert.That(driver.FindElement(By.Id("DepartmentId-error")).Text,
Is.EqualTo("Please select department!"));
    Assert.That(driver.FindElement(By.Id("TeacherId-error")).Text,
Is.EqualTo("Please select the teacher Name!"));
    Assert.That(driver.FindElement(By.Id("CourseId-error")).Text,
Is.EqualTo("Please select semester!"));
    driver.FindElement(By.Id("DepartmentId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DepartmentId"));
      dropdown.FindElement(By.XPath("//option[. = 'Patologia']")).Click();
    }
    driver.FindElement(By.Id("DepartmentId")).Click();
    driver.FindElement(By.Id("TeacherId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("TeacherId"));
      dropdown.FindElement(By.XPath("//option[. = 'Karla Guevara']")).Click();
    }
    driver.FindElement(By.Id("TeacherId")).Click();
    {
      string value =
driver.FindElement(By.Id("CreditTobeTaken")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("4"));
    }
    {
```

```
      string value =
driver.FindElement(By.Id("CreditTaken")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("4"));
    }
    driver.FindElement(By.Id("CourseId")).Click();
    {
      var element = driver.FindElement(By.Id("CreditTobeTaken"));
      Boolean isEditable = element.Enabled && element.GetAttribute("readonly")
== null;
      Assert.False(isEditable);
    }
    {
      var element = driver.FindElement(By.Id("CreditTaken"));
      Boolean isEditable = element.Enabled && element.GetAttribute("readonly")
== null;
      Assert.False(isEditable);
    }
    driver.FindElement(By.Id("CourseId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("CourseId"));
      dropdown.FindElement(By.XPath("//option[. = '567890']")).Click();
    }
    driver.FindElement(By.Id("CourseId")).Click();
    {
      string value = driver.FindElement(By.Id("Name")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("Patologia 1"));
    }
    {
      var element = driver.FindElement(By.Id("Name"));
      Boolean isEditable = element.Enabled && element.GetAttribute("readonly")
== null;
      Assert.False(isEditable);
    }
    {
      string value = driver.FindElement(By.Id("Credit")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("4"));
    }
    {
      var element = driver.FindElement(By.Id("Credit"));
      Boolean isEditable = element.Enabled && element.GetAttribute("readonly")
== null;
      Assert.False(isEditable);
    }
    driver.FindElement(By.Id("submitButton")).Click();
    Assert.That(driver.FindElement(By.CssSelector("p:nth-child(4)")).Text,
Is.EqualTo("Assigned successfully"));
    driver.Close();
```

```
  }
}
```

TC06 Crear estudiante

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC06CrearestudianteTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC06Crearestudiante() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
    driver.FindElement(By.LinkText("Department")).Click();
    driver.FindElement(By.LinkText("Save Department")).Click();
    driver.FindElement(By.Id("Code")).Click();
    driver.FindElement(By.Id("Code")).SendKeys("05");
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Ingieneria");
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("form > p")).Text,
Is.EqualTo("Saved Successfully"));
    driver.FindElement(By.LinkText("Student")).Click();
    driver.FindElement(By.LinkText("Register Student")).Click();
```

```
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.Id("Name-error")).Text, Is.EqualTo("Please
enter the student Name"));
    Assert.That(driver.FindElement(By.Id("Email-error")).Text,
Is.EqualTo("Please email address is required"));
    Assert.That(driver.FindElement(By.Id("Contact-error")).Text,
Is.EqualTo("Please enter the student contact"));
    Assert.That(driver.FindElement(By.Id("Address-error")).Text,
Is.EqualTo("Address is required!"));
    Assert.That(driver.FindElement(By.Id("DepartmentId-error")).Text,
Is.EqualTo("Please select department"));
    driver.FindElement(By.Id("Name")).Click();
    driver.FindElement(By.Id("Name")).SendKeys("Fabian Gutierrez");
    driver.FindElement(By.Id("Email")).Click();
    driver.FindElement(By.Id("Email")).SendKeys("fabianjim20@gmail.com");
    driver.FindElement(By.Id("Contact")).Click();
    driver.FindElement(By.Id("Contact")).SendKeys("88888888");
    driver.FindElement(By.Id("RegDate")).Click();
    driver.FindElement(By.Id("RegDate")).SendKeys("12/8/1997");
    driver.FindElement(By.LinkText("1")).Click();
    driver.FindElement(By.Id("Address")).Click();
    driver.FindElement(By.Id("Address")).SendKeys("Heredia");
    driver.FindElement(By.Id("DepartmentId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DepartmentId"));
      dropdown.FindElement(By.XPath("//option[. = 'Ingieneria']")).Click();
    }
    driver.FindElement(By.Id("DepartmentId")).Click();
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector(".body-content:nth-
child(5)")).Text, Is.EqualTo("Saved Successfully!\r\nRegistration No:05-2020-
001\r\nName:Fabian Gutierrez\r\nEmail:fabianjim20@gmail.com\r\nContact
Number:88888888"));
    driver.Close();
  }
}
```

TC07: Crear horario de clase

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
```

```
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC07CrearhorariodeclaseTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC07Crearhorariodeclase() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
    driver.FindElement(By.LinkText("Class Room Allocation")).Click();
    driver.FindElement(By.LinkText("Allocate Classroom")).Click();
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.Id("DepartmentId-error")).Text,
Is.EqualTo("Please select department!"));
    Assert.That(driver.FindElement(By.Id("CourseId-error")).Text,
Is.EqualTo("Please Select Course!"));
    Assert.That(driver.FindElement(By.Id("RoomId-error")).Text,
Is.EqualTo("Please Select room!"));
    Assert.That(driver.FindElement(By.Id("DayId-error")).Text,
Is.EqualTo("Please Select day!"));
    Assert.That(driver.FindElement(By.Id("StartTime-error")).Text,
Is.EqualTo("Select Start time!"));
    Assert.That(driver.FindElement(By.Id("Endtime-error")).Text,
Is.EqualTo("Please select End time!"));
    driver.FindElement(By.CssSelector(".body-content")).Click();
    driver.FindElement(By.Id("DepartmentId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DepartmentId"));
      dropdown.FindElement(By.XPath("//option[. = 'Artesanía']")).Click();
    }
    driver.FindElement(By.Id("DepartmentId")).Click();
    driver.FindElement(By.Id("CourseId")).Click();
    {
```

```
      var dropdown = driver.FindElement(By.Id("CourseId"));
      dropdown.FindElement(By.XPath("//option[. = '453214']")).Click();
    }
    driver.FindElement(By.Id("CourseId")).Click();
    driver.FindElement(By.Id("RoomId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("RoomId"));
      dropdown.FindElement(By.XPath("//option[. = 'Room No:101']")).Click();
    }
    driver.FindElement(By.Id("RoomId")).Click();
    driver.FindElement(By.Id("DayId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DayId"));
      dropdown.FindElement(By.XPath("//option[. = 'Monday']")).Click();
    }
    driver.FindElement(By.Id("DayId")).Click();
    driver.FindElement(By.CssSelector(".form-group:nth-child(5) > .control-
label")).Click();
    driver.FindElement(By.CssSelector(".dww1 > .dwwbm > span")).Click();
    driver.FindElement(By.CssSelector(".dww1 > .dwwbm > span")).Click();
    {
      var element = driver.FindElement(By.CssSelector(".dww1 > .dwwbm >
span"));
      Actions builder = new Actions(driver);
      builder.DoubleClick(element).Perform();
    }
    driver.FindElement(By.CssSelector(".dww1 > .dwwbm > span")).Click();
    driver.FindElement(By.CssSelector(".dww2 > .dwwbm > span")).Click();
    driver.FindElement(By.LinkText("Set")).Click();
    driver.FindElement(By.CssSelector(".form-group:nth-child(6) > .control-
label")).Click();
    driver.FindElement(By.CssSelector(".dww0 > .dwwbp > span")).Click();
    driver.FindElement(By.CssSelector(".dww0 > .dwwbp > span")).Click();
    {
      var element = driver.FindElement(By.CssSelector(".dww0 > .dwwbp >
span"));
      Actions builder = new Actions(driver);
      builder.DoubleClick(element).Perform();
    }
    driver.FindElement(By.CssSelector(".dww1 > .dwwbp > span")).Click();
    driver.FindElement(By.CssSelector(".dww1 > .dwwbp > span")).Click();
    {
      var element = driver.FindElement(By.CssSelector(".dww1 > .dwwbp >
span"));
      Actions builder = new Actions(driver);
      builder.DoubleClick(element).Perform();
    }
```

```
    driver.FindElement(By.LinkText("Set")).Click();
    driver.FindElement(By.CssSelector(".btn")).Click();
    Assert.That(driver.FindElement(By.CssSelector("p:nth-child(5)")).Text,
Is.EqualTo("Saved Successfully !"));
    driver.FindElement(By.LinkText("Class Room Allocation")).Click();
    driver.FindElement(By.LinkText("View Class Schedule")).Click();
    driver.FindElement(By.Id("DepartmentId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("DepartmentId"));
      dropdown.FindElement(By.XPath("//option[. = 'Artesanía']")).Click();
    }
        driver.FindElement(By.Id("DepartmentId")).Click();
        Assert.That(driver.FindElement(By.CssSelector("tbody:nth-child(2)
td:nth-child(1)")).Text, Is.EqualTo("453214"));
        Assert.That(driver.FindElement(By.CssSelector("tbody:nth-child(2)
td:nth-child(2)")).Text, Is.EqualTo("Moldaje"));
        driver.Close();
  }
}
```

TC08 Registrar estudiante en un curso

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC08RegistrarestudianteenuncursoTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
```

```
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC08Registrarestudianteenuncurso() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
        driver.FindElement(By.LinkText("Student")).Click();
        driver.FindElement(By.LinkText("Enroll Student In A Course")).Click();
        driver.FindElement(By.CssSelector(".btn")).Click();
        Assert.That(driver.FindElement(By.Id("StudentId-error")).Text,
Is.EqualTo("Please select student Registration No"));
        Assert.That(driver.FindElement(By.Id("CourseId-error")).Text,
Is.EqualTo("Please select Course"));
        driver.FindElement(By.Id("StudentId")).Click();
        {
            var dropdown = driver.FindElement(By.Id("StudentId"));
            dropdown.FindElement(By.XPath("//option[. = '19-2020-
001']")).Click();
        }
        driver.FindElement(By.Id("StudentId")).Click();
        driver.FindElement(By.Id("CourseId")).Click();
        driver.FindElement(By.Id("CourseId")).Click();
        {
            string value =
driver.FindElement(By.Id("Name")).GetAttribute("value");
            Assert.That(value, Is.EqualTo("Fabiana Salas"));
        }
        {
            string value =
driver.FindElement(By.Id("Email")).GetAttribute("value");
            Assert.That(value, Is.EqualTo("fab20@gmail.com"));
        }
        {
            string value =
driver.FindElement(By.Id("DepartmentId")).GetAttribute("value");
            Assert.That(value, Is.EqualTo("Artes Plasticas"));
        }
        {
            var element = driver.FindElement(By.Id("Name"));
            Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
            Assert.False(isEditable);
        }
        {
            var element = driver.FindElement(By.Id("Email"));
            Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
```

```
            Assert.False(isEditable);
        }
        {
            var element = driver.FindElement(By.Id("DepartmentId"));
            Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
            Assert.False(isEditable);
        }
        driver.FindElement(By.CssSelector(".btn")).Click();
        Assert.That(driver.FindElement(By.CssSelector("p:nth-child(4)")).Text,
Is.EqualTo("Saved Successfully!"));
        driver.Close();
    }
}
```

TC09 Añadir notas al estudiante

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC09AadirnotasalestudianteTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC09Aadirnotasalestudiante() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
```

```
        driver.FindElement(By.LinkText("Student")).Click();
        driver.FindElement(By.LinkText("Save Student Result")).Click();
        driver.FindElement(By.CssSelector(".btn")).Click();
        Assert.That(driver.FindElement(By.Id("StudentId-error")).Text,
Is.EqualTo("Please select the student reg no"));
        Assert.That(driver.FindElement(By.Id("CourseId-error")).Text,
Is.EqualTo("Please select a course"));
        Assert.That(driver.FindElement(By.Id("Grade-error")).Text,
Is.EqualTo("Please select grade"));
        driver.FindElement(By.Id("StudentId")).Click();
        {
            var dropdown = driver.FindElement(By.Id("StudentId"));
            dropdown.FindElement(By.XPath("//option[. = '16-1997-
001']")).Click();
        }
        driver.FindElement(By.Id("StudentId")).Click();
        driver.FindElement(By.Id("CourseId")).Click();
        {
            var dropdown = driver.FindElement(By.Id("CourseId"));
            dropdown.FindElement(By.XPath("//option[. = 'Natación']")).Click();
        }
        driver.FindElement(By.Id("CourseId")).Click();
        driver.FindElement(By.Id("Grade")).Click();
        {
            var dropdown = driver.FindElement(By.Id("Grade"));
            dropdown.FindElement(By.XPath("//option[. = 'A+']")).Click();
        }
        driver.FindElement(By.Id("Grade")).Click();
        {
            string value =
driver.FindElement(By.Id("Name")).GetAttribute("value");
            Assert.That(value, Is.EqualTo("Martha Rodriguez"));
        }
        {
            string value =
driver.FindElement(By.Id("Email")).GetAttribute("value");
            Assert.That(value, Is.EqualTo("maro@gmail.com"));
        }
        {
            string value =
driver.FindElement(By.Id("Department")).GetAttribute("value");
            Assert.That(value, Is.EqualTo("Deporte"));
        }
        {
            var element = driver.FindElement(By.Id("Name"));
            Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
```

```
            Assert.False(isEditable);
        }
        {
            var element = driver.FindElement(By.Id("Email"));
            Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
            Assert.False(isEditable);
        }
        {
            var element = driver.FindElement(By.Id("Department"));
            Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
            Assert.False(isEditable);
        }
        driver.FindElement(By.CssSelector(".btn")).Click();
        Assert.That(driver.FindElement(By.CssSelector("#saveStudentResult >
p")).Text, Is.EqualTo("Saved sucessfull!"));
        driver.Close();
  }
}
```

TC10: Ver Notas del estudiante

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class TC10VerNotasdelestudianteTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
```

```
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void tC10VerNotasdelestudiante() {
    driver.Navigate().GoToUrl("http://25.76.110.243:8030/");
    driver.FindElement(By.LinkText("Student")).Click();
    driver.FindElement(By.LinkText("View Result")).Click();
    driver.FindElement(By.Id("StudentId")).Click();
    {
      var dropdown = driver.FindElement(By.Id("StudentId"));
      dropdown.FindElement(By.XPath("//option[. = '22-2020-001']")).Click();
    }
    driver.FindElement(By.Id("StudentId")).Click();
    {
      string value = driver.FindElement(By.Id("Name")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("Hugo Jimenez"));
    }
    {
      string value = driver.FindElement(By.Id("Email")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("jim22@gmail.com"));
    }
    {
      string value =
driver.FindElement(By.Id("Department")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("Salud"));
    }
    {
      string value =
driver.FindElement(By.Id("Department")).GetAttribute("value");
      Assert.That(value, Is.EqualTo("Salud"));
    }
    {
      var element = driver.FindElement(By.Id("Name"));
      Boolean isEditable = element.Enabled && element.GetAttribute("readonly")
== null;
      Assert.False(isEditable);
    }
    {
      var element = driver.FindElement(By.Id("Email"));
      Boolean isEditable = element.Enabled && element.GetAttribute("readonly")
== null;
      Assert.False(isEditable);
    }
    {
      var element = driver.FindElement(By.Id("Department"));
```

```
      Boolean isEditable = element.Enabled && element.GetAttribute("readonly")
== null;
      Assert.False(isEditable);
    }
    Assert.That(driver.FindElement(By.CssSelector("#myData td:nth-
child(1)")).Text, Is.EqualTo("1166995"));
    Assert.That(driver.FindElement(By.CssSelector("#myData td:nth-
child(2)")).Text, Is.EqualTo("Enfermeria"));
    Assert.That(driver.FindElement(By.CssSelector("#myData td:nth-
child(3)")).Text, Is.EqualTo("A+"));
  }
}
```

Anexo 5. Scripts automatizados del framework Test Library Architecture.

Para más detalle, el proyecto que contiene los scripts del framework Test Library Architecture se pueden encontrar en el siguiente link de Github https://github.com/TFGH-F/Test_Library_Architecture.

TC01 Crear departamento

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;

namespace PruebaSelenium
{
    [TestClass]
    public class TC01CrearDepartamento
    {
        [TestMethod,]
        public void TC01Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                //Iniciar Página Principal
```

```
                Libreria test = new Libreria();
                //Resultado Esperado: Se abre la página de creación de
departamentos
                test.AbrirPaginaPrincipal(wdriver);
                //Creación de departamento
                test.CrearDepartamento(wdriver, "01", "Ciencias");
                //Darle click a la opción View All Departments en el menú
principal
                wdriver.FindElement(By.LinkText("Department")).Click();
                wdriver.FindElement(By.LinkText("View All
Departments")).Click();
                //Verificación de datos
                wdriver.FindElement(By.LinkText("2")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector(".grid-row:nth-
child(2) > .grid-cell:nth-child(1)")).Text, Is.EqualTo("01"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector(".grid-row:nth-
child(2) > .grid-cell:nth-child(2)")).Text, Is.EqualTo("Ciencias"));
                wdriver.Close();
            }
        }
    }
}
```

TC02 Ingresar código de departamento repetido

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;

namespace PruebaSelenium
{
    [TestClass]
    public class TC02IngresarCodRepetido
    {
        [TestMethod,]
        public void TC02Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
```

```
                test.AbrirPaginaPrincipal(wdriver);
                // Abrimos la página web de inicio se sesión
                //Crear Departamento
                test.CrearDepartamento(wdriver, "02", "Astronomia");
                test.BuscarMenu(wdriver, "Department", "Save Department");
                wdriver.FindElement(By.Id("Code")).Click();
                //Validar departamento repetido
                wdriver.FindElement(By.Id("Code")).SendKeys("02");
                wdriver.FindElement(By.Id("Name")).Click();
                wdriver.FindElement(By.Id("Name")).SendKeys("Estudios");
                wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("form >
p")).Text, Is.EqualTo("Department Code Already Exists. Department Codebe be
unique"));
                //Darle click a la opción View All Departments en el menú
principal
                test.BuscarMenu(wdriver, "Department", "View All Departments");
                //Validar que el departamento creado  se encuentre dentro de
View All Departments en el menú principal
                wdriver.FindElement(By.LinkText("2")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector(".grid-row:nth-
child(3) > .grid-cell:nth-child(1)")).Text, Is.EqualTo("02"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector(".grid-row:nth-
child(3) > .grid-cell:nth-child(2)")).Text, Is.EqualTo("Astronomia"));
                wdriver.Close();
            }
        }
    }
}
```

TC03 Crear curso

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;

namespace PruebaSelenium
{
    [TestClass]
    public class TC03CrearCurso
```

```
    {
        [TestMethod,]
        public void TC03Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
                test.AbrirPaginaPrincipal(wdriver);
                //Crear Departamento
                test.CrearDepartamento(wdriver, "31", "Química");
                test.BuscarMenu(wdriver, "Course", "Save Course");
                wdriver.FindElement(By.CssSelector(".btn")).Click();
                //Validación de campos vacíos
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Code-
error")).Text, Is.EqualTo("Please enter a valid Course Code!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Name-
error")).Text, Is.EqualTo("Please enter the valid Course Name!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Credit-
error")).Text, Is.EqualTo("Credit is required!"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Description-
error")).Text, Is.EqualTo("Write something about course!"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("DepartmentId-
error")).Text, Is.EqualTo("Please select related department!"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("SemesterId-error")).Text,
Is.EqualTo("Please select related semester!"));
                //Se ingresan los campos del curso
                wdriver.FindElement(By.Id("Code")).Click();
                wdriver.FindElement(By.Id("Code")).SendKeys("564378");
                wdriver.FindElement(By.Id("Name")).Click();
                wdriver.FindElement(By.Id("Name")).SendKeys("Quimica avanzado");
                wdriver.FindElement(By.Id("Credit")).Click();
                wdriver.FindElement(By.Id("Credit")).SendKeys("4");
                wdriver.FindElement(By.Id("Description")).Click();
                wdriver.FindElement(By.Id("Description")).SendKeys("Este curso
pertenece a quimica");
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("DepartmentId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Química']")).Click();
                }
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                wdriver.FindElement(By.Id("SemesterId")).Click();
```

```
                {
                    var dropdown = wdriver.FindElement(By.Id("SemesterId"));
                    dropdown.FindElement(By.XPath("//option[. =
'1st']")).Click();
                }
                wdriver.FindElement(By.Id("SemesterId")).Click();
                wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("p:nth-
child(4)")).Text, Is.EqualTo("Saved Successfully"));
                //Se valida que este el curso fue creado
                test.BuscarMenu(wdriver, "Course", "View Course Statics");
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("DepartmentId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Química']")).Click();
                }
                wdriver.FindElement(By.Id("DepartmentId")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("td:nth-
child(1)")).Text, Is.EqualTo("564378"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("td:nth-
child(2)")).Text, Is.EqualTo("Quimica avanzado"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("td:nth-
child(3)")).Text, Is.EqualTo("1st"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("td:nth-
child(4)")).Text, Is.EqualTo("Not Assigned yet"));
                wdriver.Close();


            }
        }
    }
}
```

TC04: Crear profesor

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
```

```
using NUnit.Framework;

namespace PruebaSelenium
{
    [TestClass]
    public class TC04CrearProfesor
    {
        [TestMethod,]
        public void TC04Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
                test.AbrirPaginaPrincipal(wdriver);
                //Se crea un departamento
                test.CrearDepartamento(wdriver, "04", "Matematica");
                //Se ingresa al menu del profesor
                test.BuscarMenu(wdriver, "Teacher", "Save Teacher");
                wdriver.FindElement(By.CssSelector(".btn")).Click();
                //Se valida los campos Vacíos
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Name-
error")).Text, Is.EqualTo("Please enter the Name"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Address-
error")).Text, Is.EqualTo("Address is required!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Email-
error")).Text, Is.EqualTo("Please email address is required"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Contact-
error")).Text, Is.EqualTo("Please enter the contact"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("DesignationId-
error")).Text, Is.EqualTo("Designation is required!"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("DepartmentId-
error")).Text, Is.EqualTo("Please select department"));
                //Se ingresan los datos en los campos de texto
                wdriver.FindElement(By.Id("Name")).Click();
                wdriver.FindElement(By.Id("Name")).SendKeys("Luisa Jimenez");
                wdriver.FindElement(By.Id("Address")).Click();
                wdriver.FindElement(By.Id("Address")).SendKeys("Heredia");
                wdriver.FindElement(By.Id("Email")).Click();

wdriver.FindElement(By.Id("Email")).SendKeys("luisa20@gmail.com");
                wdriver.FindElement(By.Id("Contact")).Click();
                wdriver.FindElement(By.Id("Contact")).SendKeys("88888888");
                wdriver.FindElement(By.Id("DesignationId")).Click();
                {
```

```
                    var dropdown = wdriver.FindElement(By.Id("DesignationId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Professor']")).Click();
                }
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("DepartmentId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Matematica']")).Click();
                }
                //Se valida que el profesor se creara exitosamente
                wdriver.FindElement(By.Id("CreditTobeTaken")).Click();
                wdriver.FindElement(By.Id("CreditTobeTaken")).SendKeys("1");
                wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("p:nth-
child(4)")).Text, Is.EqualTo("Saved Sucessfully"));
                wdriver.Close();

            }
        }
    }
}
```

TC05: Asignar curso a profesor

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;
using OpenQA.Selenium.Interactions;

namespace PruebaSelenium
{
    [TestClass]
    public class TC05AsigCursoProfesor
    {
        [TestMethod,]
        public void TC05Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
```

```
                test.AbrirPaginaPrincipal(wdriver);
               //Se ingresa al menu de asignación de profesor
                test.BuscarMenu(wdriver, "Teacher", "Assign Course To Teacher");
                //Se asigna al profesor a un curso
                wdriver.FindElement(By.Id("submitButton")).Click();
                {
                    var element = wdriver.FindElement(By.Id("submitButton"));
                    Actions builder = new Actions(wdriver);
                    builder.MoveToElement(element).Perform();
                }
                {
                    var element = wdriver.FindElement(By.TagName("body"));
                    Actions builder = new Actions(wdriver);
                    builder.MoveToElement(element, 0, 0).Perform();
                }
                //Se valida los campos de texto vacíos

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("DepartmentId-
error")).Text, Is.EqualTo("Please select department!"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("TeacherId-error")).Text,
Is.EqualTo("Please select the teacher Name!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("CourseId-
error")).Text, Is.EqualTo("Please select semester!"));
                //Se llenan los datos
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("DepartmentId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Patologia']")).Click();
                }
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                wdriver.FindElement(By.Id("TeacherId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("TeacherId"));
                    dropdown.FindElement(By.XPath("//option[. = 'Karla
Guevara']")).Click();
                }
                wdriver.FindElement(By.Id("TeacherId")).Click();
                {
                    string value =
wdriver.FindElement(By.Id("CreditTobeTaken")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("4"));
                }
                {
                    string value =
wdriver.FindElement(By.Id("CreditTaken")).GetAttribute("value");
```

```
                    NUnit.Framework.Assert.That(value, Is.EqualTo("4"));
                }
                wdriver.FindElement(By.Id("CourseId")).Click();
                {
                    var element = wdriver.FindElement(By.Id("CreditTobeTaken"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    var element = wdriver.FindElement(By.Id("CreditTaken"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                wdriver.FindElement(By.Id("CourseId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("CourseId"));
                    dropdown.FindElement(By.XPath("//option[. =
'567890']")).Click();
                }
                wdriver.FindElement(By.Id("CourseId")).Click();
                {
                    string value =
wdriver.FindElement(By.Id("Name")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Patologia
1"));
                }
                {
                    var element = wdriver.FindElement(By.Id("Name"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    string value =
wdriver.FindElement(By.Id("Credit")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("4"));
                }
                {
                    var element = wdriver.FindElement(By.Id("Credit"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                //Se valida que la asignación fue correcta
                wdriver.FindElement(By.Id("submitButton")).Click();
```

```
NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("p:nth-
child(4)")).Text, Is.EqualTo("Assigned successfully"));
                wdriver.Close();

            }
        }
    }
}
```

TC06 Crear estudiante

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;

namespace PruebaSelenium
{
    [TestClass]
    public class TC06CrearEstudiante
    {
        [TestMethod,]
        public void TC06Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
                test.AbrirPaginaPrincipal(wdriver);
                //Se crea departamento
                test.CrearDepartamento(wdriver, "05", "Ingieneria");
                //Se registra un estudiante
                test.BuscarMenu(wdriver, "Student", "Register Student");
                wdriver.FindElement(By.CssSelector(".btn")).Click();
                //Se valida los campos vacíos
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Name-
error")).Text, Is.EqualTo("Please enter the student Name"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Email-
error")).Text, Is.EqualTo("Please email address is required"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Contact-
error")).Text, Is.EqualTo("Please enter the student contact"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Address-
error")).Text, Is.EqualTo("Address is required!"));
```

```
NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("DepartmentId-
error")).Text, Is.EqualTo("Please select department"));
                //Se registra los datos en los campos de texto
                 wdriver.FindElement(By.Id("Name")).Click();
                 wdriver.FindElement(By.Id("Name")).SendKeys("Fabian Gutierrez");
                 wdriver.FindElement(By.Id("Email")).Click();

wdriver.FindElement(By.Id("Email")).SendKeys("fabianjim20@gmail.com");
                 wdriver.FindElement(By.Id("Contact")).Click();
                 wdriver.FindElement(By.Id("Contact")).SendKeys("88888888");
                 wdriver.FindElement(By.Id("RegDate")).Click();
                 wdriver.FindElement(By.Id("RegDate")).SendKeys("12/8/1997");
                 wdriver.FindElement(By.LinkText("1")).Click();
                 wdriver.FindElement(By.Id("Address")).Click();
                 wdriver.FindElement(By.Id("Address")).SendKeys("Heredia");
                 wdriver.FindElement(By.Id("DepartmentId")).Click();
                 {
                     var dropdown = wdriver.FindElement(By.Id("DepartmentId"));
                     dropdown.FindElement(By.XPath("//option[. =
'Ingieneria']")).Click();
                 }
                 //Se valida que se registro el estudiante exitosamente
                 wdriver.FindElement(By.Id("DepartmentId")).Click();
                 wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector(".body-
content:nth-child(5)")).Text, Is.EqualTo("Saved Successfully!\r\nRegistration
No:05-2020-001\r\nName:Fabian
Gutierrez\r\nEmail:fabianjim20@gmail.com\r\nContact Number:88888888"));
                 wdriver.Close();
             }
         }
     }
}
```

TC07: Crear horario de clase

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;
using OpenQA.Selenium.Interactions;
```

```
namespace PruebaSelenium
{
    [TestClass]
    public class TC07CrearHorararioClase
    {
        [TestMethod,]
        public void TC07Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
                test.AbrirPaginaPrincipal(wdriver);
                // Abrimos la página web de inicio se sesión
                //Se ingresa al menu de asociar clase
                test.BuscarMenu(wdriver, "Class Room Allocation", "Allocate
Classroom");
                wdriver.FindElement(By.CssSelector(".btn")).Click();
                //Se valida los Campos de texto Vacíos

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("DepartmentId-
error")).Text, Is.EqualTo("Please select department!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("CourseId-
error")).Text, Is.EqualTo("Please Select Course!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("RoomId-
error")).Text, Is.EqualTo("Please Select room!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("DayId-
error")).Text, Is.EqualTo("Please Select day!"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("StartTime-error")).Text,
Is.EqualTo("Select Start time!"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Endtime-
error")).Text, Is.EqualTo("Please select End time!"));
                wdriver.FindElement(By.CssSelector(".body-content")).Click();
                //Se registra los datos
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("DepartmentId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Artesanía']")).Click();
                }
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                wdriver.FindElement(By.Id("CourseId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("CourseId"));
                    dropdown.FindElement(By.XPath("//option[. =
'453214']")).Click();
```

```
                }
                wdriver.FindElement(By.Id("CourseId")).Click();
                wdriver.FindElement(By.Id("RoomId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("RoomId"));
                    dropdown.FindElement(By.XPath("//option[. = 'Room
No:101']")).Click();
                }
                wdriver.FindElement(By.Id("RoomId")).Click();
                wdriver.FindElement(By.Id("DayId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("DayId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Monday']")).Click();
                }
                wdriver.FindElement(By.Id("DayId")).Click();
                wdriver.FindElement(By.CssSelector(".form-group:nth-child(5) >
.control-label")).Click();
                wdriver.FindElement(By.CssSelector(".dww11 > .dwwbm >
span")).Click();
                wdriver.FindElement(By.CssSelector(".dww11 > .dwwbm >
span")).Click();
                {
                    var element = wdriver.FindElement(By.CssSelector(".dww11 >
.dwwbm > span"));
                    Actions builder = new Actions(wdriver);
                    builder.DoubleClick(element).Perform();
                }
                wdriver.FindElement(By.CssSelector(".dww11 > .dwwbm >
span")).Click();
                wdriver.FindElement(By.CssSelector(".dww12 > .dwwbm >
span")).Click();
                wdriver.FindElement(By.LinkText("Set")).Click();
                wdriver.FindElement(By.CssSelector(".form-group:nth-child(6) >
.control-label")).Click();
                wdriver.FindElement(By.CssSelector(".dww10 > .dwwbp >
span")).Click();
                wdriver.FindElement(By.CssSelector(".dww10 > .dwwbp >
span")).Click();
                {
                    var element = wdriver.FindElement(By.CssSelector(".dww10 >
.dwwbp > span"));
                    Actions builder = new Actions(wdriver);
                    builder.DoubleClick(element).Perform();
                }
                wdriver.FindElement(By.CssSelector(".dww11 > .dwwbp >
span")).Click();
```

```
                wdriver.FindElement(By.CssSelector(".dww11 > .dwwbp >
span")).Click();
                {
                    var element = wdriver.FindElement(By.CssSelector(".dww11 >
.dwwbp > span"));
                    Actions builder = new Actions(wdriver);
                    builder.DoubleClick(element).Perform();
                }
                //Se valida que la clase fue registrada correctamente
                wdriver.FindElement(By.LinkText("Set")).Click();
                wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("p:nth-
child(5)")).Text, Is.EqualTo("Saved Successfully !"));
                //Se valida que la clase esta registrada
                test.BuscarMenu(wdriver, "Class Room Allocation", "View Class
Schedule");
                wdriver.FindElement(By.Id("DepartmentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("DepartmentId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Artesanía']")).Click();
                }
                wdriver.FindElement(By.Id("DepartmentId")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("tbody:nth-
child(2) td:nth-child(1)")).Text, Is.EqualTo("453214"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("tbody:nth-
child(2) td:nth-child(2)")).Text, Is.EqualTo("Moldaje"));
                wdriver.Close();
            }
        }
    }
}
```

TC08 Registrar estudiante en un curso

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;
using OpenQA.Selenium.Interactions;
```

```
namespace PruebaSelenium
{
    [TestClass]
    public class TC08Registrarestudiante
    {
        [TestMethod,]
        public void TC08Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
                test.AbrirPaginaPrincipal(wdriver);
                //Se ingresa al menu de estudiante en Enroll Student In A Course
                test.BuscarMenu(wdriver, "Student", "Enroll Student In A
Course");
                wdriver.FindElement(By.CssSelector(".btn")).Click();
                //Se valida los campos de texto Vacíos

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("StudentId-error")).Text,
Is.EqualTo("Please select student Registration No"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("CourseId-
error")).Text, Is.EqualTo("Please select Course"));
                wdriver.FindElement(By.Id("StudentId")).Click();
                //Se ingresan los datos del Estudiante
                {
                    var dropdown = wdriver.FindElement(By.Id("StudentId"));
                    dropdown.FindElement(By.XPath("//option[. = '19-2020-
001']")).Click();
                }
                wdriver.FindElement(By.Id("StudentId")).Click();
                {
                    string value =
wdriver.FindElement(By.Id("Name")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Fabiana
Salas"));
                }
                {
                    string value =
wdriver.FindElement(By.Id("Email")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value,
Is.EqualTo("fab20@gmail.com"));
                }
                {
                    string value =
wdriver.FindElement(By.Id("DepartmentId")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Artes
```

```
Plasticas"));
                }
                {
                    var element = wdriver.FindElement(By.Id("Name"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    var element = wdriver.FindElement(By.Id("Email"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    var element = wdriver.FindElement(By.Id("DepartmentId"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                //Se valida que los datos se registraron correctamente
                wdriver.FindElement(By.Id("CourseId")).Click();
                wdriver.FindElement(By.Id("CourseId")).Click();
                wdriver.FindElement(By.Id("EnrollDate")).Click();
                wdriver.FindElement(By.LinkText("8")).Click();
                wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("p:nth-
child(4)")).Text, Is.EqualTo("Saved Successfully!"));
                wdriver.Close();
            }
        }
    }
}
```

TC09 Añadir notas al estudiante

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;

namespace PruebaSelenium
{
```

```
    [TestClass]
    public class TC09AñadirNotas
    {
        [TestMethod,]
        public void TC09Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
                test.AbrirPaginaPrincipal(wdriver);
                // Abrimos la página web de inicio se sesión
                //Se Ingresa al menu de estudiante
                test.BuscarMenu(wdriver, "Student", "Save Student Result");
                wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("StudentId-error")).Text,
Is.EqualTo("Please select the student reg no"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("CourseId-
error")).Text, Is.EqualTo("Please select a course"));
                NUnit.Framework.Assert.That(wdriver.FindElement(By.Id("Grade-
error")).Text, Is.EqualTo("Please select grade"));
                wdriver.FindElement(By.Id("StudentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("StudentId"));
                    dropdown.FindElement(By.XPath("//option[. = '16-1997-
001']")).Click();
                }
                wdriver.FindElement(By.Id("StudentId")).Click();
                wdriver.FindElement(By.Id("CourseId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("CourseId"));
                    dropdown.FindElement(By.XPath("//option[. =
'Natación']")).Click();
                }
                wdriver.FindElement(By.Id("CourseId")).Click();
                wdriver.FindElement(By.Id("Grade")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("Grade"));
                    dropdown.FindElement(By.XPath("//option[. =
'A+']")).Click();
                }
                wdriver.FindElement(By.Id("Grade")).Click();
                {
                    string value =
wdriver.FindElement(By.Id("Name")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Martha
```

```
Rodriguez"));
                }
                {
                    string value =
wdriver.FindElement(By.Id("Email")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value,
Is.EqualTo("maro@gmail.com"));
                }
                {
                    string value =
wdriver.FindElement(By.Id("Department")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Deporte"));
                }
                {
                    var element = wdriver.FindElement(By.Id("Name"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    var element = wdriver.FindElement(By.Id("Email"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    var element = wdriver.FindElement(By.Id("Department"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                wdriver.FindElement(By.CssSelector(".btn")).Click();

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("#saveStudentResu
lt > p")).Text, Is.EqualTo("Saved sucessfull!"));
                wdriver.Close();
            }
        }
    }
}
```

TC10: Ver Notas del estudiante

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System.Threading;
```

```
using LibreriaMaestra;
using NUnit;
using NUnit.Framework;
using OpenQA.Selenium.Interactions;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;


namespace PruebaSelenium
{
    [TestClass]
    public class TC10VerNotasEst
    {
        [TestMethod,]
        public void TC10Principal()
        {
            using (IWebDriver wdriver = new ChromeDriver())
            {
                Libreria test = new Libreria();
                //Iniciar Página Principal
                test.AbrirPaginaPrincipal(wdriver);
                //Se ingresa al menu de estudiantes
                test.BuscarMenu(wdriver, "Student", "View Result");
               //Se visualizan los datos del estudiante
                wdriver.FindElement(By.Id("StudentId")).Click();
                {
                    var dropdown = wdriver.FindElement(By.Id("StudentId"));
                    dropdown.FindElement(By.XPath("//option[. = '22-2020-
001']")).Click();
                }
                wdriver.FindElement(By.Id("StudentId")).Click();
                {
                    string value =
wdriver.FindElement(By.Id("Name")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Hugo
Jimenez"));
                }
                {
                    string value =
wdriver.FindElement(By.Id("Email")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value,
Is.EqualTo("jim22@gmail.com"));
                }
                {
```

```
                    string value =
wdriver.FindElement(By.Id("Department")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Salud"));
                }
                {
                    string value =
wdriver.FindElement(By.Id("Department")).GetAttribute("value");
                    NUnit.Framework.Assert.That(value, Is.EqualTo("Salud"));
                }
                {
                    var element = wdriver.FindElement(By.Id("Name"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    var element = wdriver.FindElement(By.Id("Email"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                {
                    var element = wdriver.FindElement(By.Id("Department"));
                    Boolean isEditable = element.Enabled &&
element.GetAttribute("readonly") == null;
                    NUnit.Framework.Assert.False(isEditable);
                }
                //Se valida que la información sea correcta

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("#myData td:nth-
child(1)")).Text, Is.EqualTo("1166995"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("#myData td:nth-
child(2)")).Text, Is.EqualTo("Enfermeria"));

NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("#myData td:nth-
child(3)")).Text, Is.EqualTo("A+"));
                wdriver.Close();
            }
        }
    }
}
```

Librería de Clase

```
using System;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
```

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Threading;
using NUnit.Framework;
using OpenQA.Selenium.Support.UI;

namespace LibreriaMaestra
{
    public class Libreria
    {
          [TestMethod,]

        public void AbrirPaginaPrincipal(IWebDriver wdriver)
        {
            // Abrimos la página web
            wdriver.Navigate().GoToUrl("http://25.76.110.243:8010/");
            //Maximizamos la ventana
            wdriver.Manage().Window.Maximize();
        }

        public void CrearDepartamento(IWebDriver wdriver, string codigo, string
nombre) {

            BuscarMenu(wdriver, "Department", "Save Department");
            wdriver.FindElement(By.Id("Code")).Click();
            wdriver.FindElement(By.Id("Code")).SendKeys(codigo);
            wdriver.FindElement(By.Id("Name")).Click();
            wdriver.FindElement(By.Id("Name")).SendKeys(nombre);
            wdriver.FindElement(By.CssSelector(".btn")).Click();
            NUnit.Framework.Assert.That(wdriver.FindElement(By.CssSelector("form
> p")).Text, Is.EqualTo("Saved Successfully"));
        }
        public void BuscarMenu(IWebDriver wdriver, string opcion1, string
opcion2)
        {
            wdriver.FindElement(By.LinkText(opcion1)).Click();
            wdriver.FindElement(By.LinkText(opcion2)).Click();

        }

    }
}
```