UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo e implementación de un arnés de pruebas de automatización para validar el funcionamiento de un producto

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Mecatrónica

P R E S E N T A

Rolando Conteras Vargas

ASESOR DE INFORME

M.A. Luis Yair Bautista Blanco

Ciudad Universitaria, Cd. Mx., 2022

Contenido

1. Chamberlain Group

Chamberlain Group es una compañía estadounidense fundada en el año 1954 por Richard L. Duchossois, actualmente Grupo Chamberlain es el líder global en soluciones de acceso residencial y comercial. Hoy en día la empresa abarca marcas como LiftMaster, myQ, Controlled Products System Group, Systems, Merlin y Grifco, cada una de ellas enfocadas en el desarrollo, mejora e implementación de software y equipo para controles de acceso.[1]

La compañía cuenta con diversas sedes alrededor del mundo en países como Canadá, Alemania, Australia, Nueva Zelanda, China, Hong Kong, Taiwán y por supuesto México. La planta ubicada en Nogales, Sonora, es la planta de manufactura más importante de la compañía, debido a que en ésta se producen gran parte de los productos que son distribuidos a todo el mundo. Del mismo modo, la sede en México también es la encargada de brindar soporte con base en distintas ramas de la ingeniería como lo son el área de software, mecánica y eléctrica electrónica.

Dentro de los productos y servicios que brinda la compañía se encuentran un par de mercados principales, uno de ellos es el residencial, en éste se incluyen abridores de garaje, sistemas de control y apertura, servicios de apertura remota y elementos de seguridad. Otro de los mercados meta más importantes de Chamberlain es el apartado comercial, en este caso se desarrollan abridores de puertas a nivel industrial, controles perimetrales, sistemas de acceso inteligentes, además de contar con servicios especiales en caso de que el cliente necesite requerimientos específicos para sus controles de accesos.

CHAMBERLAIN GROUP

Ilustración 1 - Logo de la empresa

1.1 Contexto actual y objetivos de la empresa

Grupo Chamberlain fue adquirida en el a 2022 por la empresa Blackstone por la cantidad de cinco mil millones de dólares, esto representa un evento importante en la compañía debido a que desde su fundación la empresa había pertenecido siempre a la familia Duchossois; con ello se esperan cambios dentro de los niveles directivos, así como un flujo de capital mayor para el desarrollo y producción de nuevos productos.

La empresa ha evolucionado a lo largo de los años, adaptándose a los requerimientos y necesidades del mercado, por lo que su misión está enfocada a que no importa donde se encuentren las personas que cuentan con el soporte de la empresa, ésta brindará la seguridad de que lo que más valoran está protegido y siempre a su alcance.

Del mismo modo, la visión de la empresa se concentra y se define por la siguiente oración “Dar el poder del acceso y la información”. Las soluciones de la empresa están enfocadas a conectar a las personas con su entorno y del mismo modo mantenerlas en movimiento, cada una de las marcas de la empresa cuenta con una historia de desarrollo amplia y con ello soluciones robustas ante cada una de las necesidades de los clientes. Finalmente, gracias a la conectividad y al rastreo de datos, cada una de las soluciones proporcionan un aseguramiento en tiempo real y como consecuencia la tranquilidad de los clientes.

1.2 Servicios que la empresa proporciona

1.2.1 Propietarios de viviendas

Este es el servicio más conocido y tradicional de la empresa, en este apartado la empresa genera productos enfocados en el mercado residencial pues incluye abridores de garaje y operadores de puerta, los cuales han ido evolucionando con el paso de los años de acuerdo con las necesidades de los clientes. Estos productos brindan al usuario la capacidad de abrir y cerrar de manera automatizada casi cualquier tipo de garaje o puerta, los productos tienen ciertas flexibilidades para dar una solución robusta y completa.

Dentro de esta misma sección se encuentran los sistemas de control de acceso, así como accesorios que incluyen paneles de control, cámaras, chapas, controles remotos, entre otros productos. Todos estos elementos tienen la finalidad de brindar a los usuarios la posibilidad de tener acceso a su residencia según sus necesidades y cualidades de su sistema de acceso.

Ilustración 2 - Abridor de puerta con accesorios.

1.2.2 Servicios comerciales e industriales

Esta sección de productos está orientada a brindar soluciones comerciales para determinados sectores, en los que se incluyen productos como garajes industriales o puertas de almacenes, bodegas, centros comerciales entre otros. Este tipo de elementos están diseñados para soportar cargas de trabajo más altas, de esta manera se asegura la funcionalidad de acceso, pero en este caso a nivel industrial. Del mismo modo, existen productos encargados del control de entrada y salida, así como accesorios perimetrales, de repuesto y complementarios.

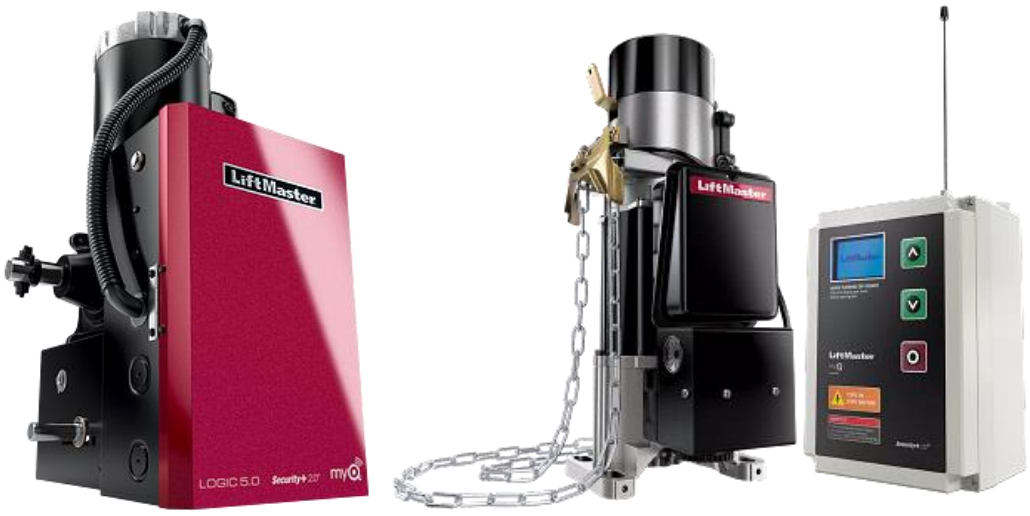

Ilustración 3 - Abridores de puertas de uso industrial.

1.2.3 Servicios Adicionales

En los últimos años Chamberlain ha desarrollado una plataforma para mantener conectados a los usuarios con su propiedad en todo momento, *myQ* es un sistema que entre sus funcionalidades permite establecer horarios de acceso; interactuar con residentes de una unidad habitacional; generar códigos de acceso y abrir o cerrar de manera remota un garaje. Este servicio cuenta con una página web, así como una aplicación móvil.

Finalmente, otro de los servicios que proporciona la empresa es la asistencia y soporte técnico de los equipos que ya se encuentran en mercado y de los proyectos que pretenden utilizar alguno de los productos. Como parte de estos servicios, el departamento adecuado puede brindar asistencia personalizada ante algún fallo o mejora de nuestros servicios.[2]

Ilustración 4 - Productos de acceso y comunicación inteligentes.

1.3 Cronograma

El funcionamiento de la empresa, hablando concretamente del acomodo organizacional, se basa tal y como la muestra la Ilustración 5. De manera central tenemos a la cabeza al CEO de la organización, dentro de sus funciones principales se encuentran mantener relaciones con los inversores y accionistas de la empresa, así como identificar cuáles son las prioridades de la empresa dentro de las distintas áreas que abarca. En este caso en particular el CEO mantiene relación constante con los vicepresidentes y presidentes de cada una de las secciones de la empresa.

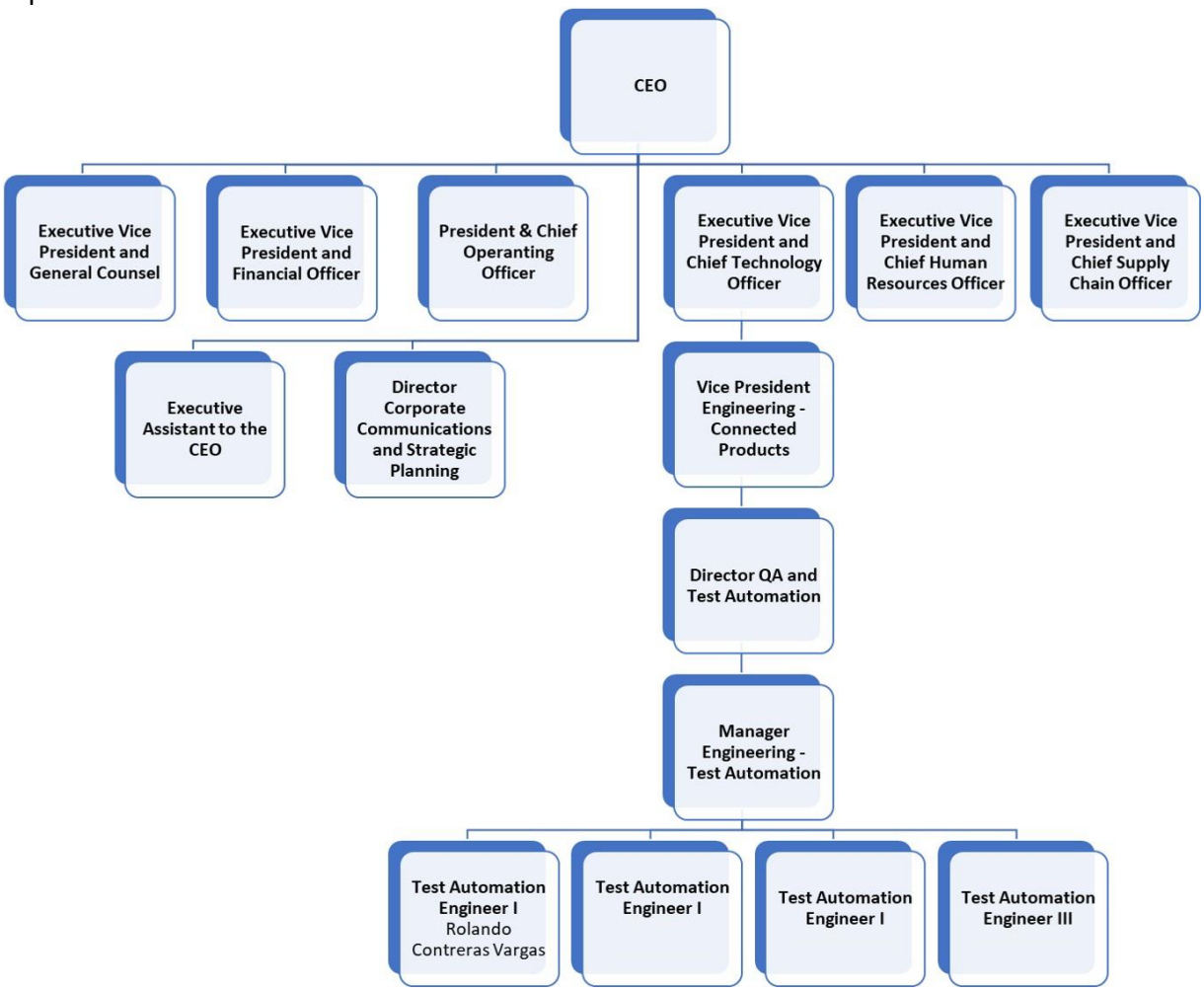

Ilustración 5 - Organigrama de la empresa.

1.4 Funciones

Mi primer contacto con Grupo Chamberlain fue a través de un programa interno llamado *"Design Engineering Apprentice Program".* Este programa tiene una duración de aproximadamente seis meses y tiene como objetivo conocer y dar solución a un problema dentro de la empresa, de manera particular fue el área de programación.

Inicialmente tomé un par de cursos sobre "Software Testing", dentro de ellos aprendí a utilizar herramientas como *Selenium*, *Appium*, protocolos de comunicación serial, entre otras. Este tipo

de herramientas están enfocadas a la manipulación automatizada de elementos web, así como aplicaciones móviles, todo ello con el objetivo de validar el funcionamiento unitario o integral de un script o elemento de un sitio web o una aplicación.

En el organigrama se muestra que el departamento encargado de la automatización de pruebas es *QA and Test Automation,* este departamento se encarga de validar el funcionamiento de manera integral de cada uno de los productos que la empresa desarrolla. De manera específica mi jefe directo es el Gerente de Pruebas Automatizadas (*Manager Engineering – Test Atomation*).

Al igual que los compañeros que estamos sobre la misma línea jerárquica, nuestra función es desarrollar soluciones con base en los requerimientos de un plan de pruebas. Es importante menciona que el funcionamiento del equipo es integral pues permite solicitar apoyo sobre algún tema en particular, aunque no se encuentren directamente relacionados con los proyectos en curso. El equipo de trabajo cuenta con elementos de nuestro país, pero también de Estados Unidos, un dato importante pues permite relacionarnos con personas que tienen como lengua nativa el idioma inglés, sin duda una habilidad indispensable para mantener una comunicación asertiva entre los elementos del equipo.

A partir de mi desempeño y de mis objetivos cumplidos dentro los proyectos, me brindaron un ascenso y obtuvé un contrato indefinido como “Test Automation Engineer I”. Dentro de este nuevo puesto seguí enriqueciendo mis conocimientos sobre la automatización de pruebas, con un par de certificaciones y cursos que han mejorado mis habilidades dentro de este campo. Dentro de estos cursos se incluye un programa impartido por la Universidad de Minnesota sobre *Software Testing,* del mismo modo tomé cursos sobre metodologías ágiles que me permitieron conocer el proceso para desarrollar proyectos con rapidez y flexibilidad para adecuarse a la necesidad de los requerimientos.

A continuación, se enlistan la serie de actividades principales dentro de esta posición:

- Evaluación de proyectos enfocados a verificar el funcionamiento del equipo de desarrollo, así como el software del sistema.
- Desarrollo de escenarios de prueba a través de scripts de programación en lenguaje Java o C#.
- Generar reportes automatizados de cada una de las pruebas realizadas a los productos.
- Documentar cada uno de los escenarios de prueba en la plataforma de la empresa, así como actualizar la plataforma Test Rail con cada uno de los pasos para cada prueba.
- Realizar reportes quincenales con el avance realizado durante este periodo haciendo énfasis en la cobertura de las pruebas y el porcentaje de avance.
- Diseñar *features* electrónicos/mecánicos para el desarrollo de pruebas.

Inicialmente mis funciones estaban orientadas a la programación directa de los escenarios de prueba, en este apartado aprendí sobre técnicas de programación eficientes y orientadas a proyectos de *Software Testing*. En este punto en específico conocí el *IDE Eclipse* con el cual desarrollo los escenarios de prueba en el lenguaje Java, este IDE es una herramienta que permite

conocer el entorno de trabajo, así como las herramientas de programación.

Para llevar un control adecuado del código de programación generado se utilizan repositorios con control de versiones, en este caso el proveedor del servicio es *Bit Bucket* y el programa que muestra la interfaz de trabajo es *Source Tree*. Con base en este sistema aprendí un par de procesos y comandos para tener en orden el código generado, así como acceder a versiones anteriores de manera segura.

Manejar un control de versiones adecuado permite al equipo tener organizado cada uno de los cambios o actualizaciones dentro del proyecto; este tipo de estructura ayuda a evitar la perdida de información, un ámbito de suma importancia debido a lo que implica dentro de una organización el tiempo y trabajo empleado por cada uno de los desarrolladores involucrados.

Es necesario añadir que cuando un integrante nuevo del equipo se incorpora es más sencillo el proceso de aprendizaje debido a que todo se encuentra documentado dentro del repositorio, del mismo modo si un compañero de trabajo deja o cambia de puesto, toda la información que desarrolló se encuentra respaldada dentro del repositorio.

Cada proyecto en específico suele requerir dependencias adicionales, para ello utilicé la arquitectura de proyectos *Maven,* la cual funciona para importar librerías en específico al proyecto. Para este apartado utilicé dependencias para manejar correos electrónicos, protocolos de comunicación serial, manejo de imágenes, video, audio, entre otras. Por mencionar un par de ejemplos para el manejo de comunicación serial realicé scripts para enviar y leer información para los protocolos de comunicación Serial y Telnet, del mismo modo, creé algunas funciones para verificar la calidad de imágenes.

Las librerías de comunicación serial que creé mientras desarrollaba el proyecto tienen la capacidad de ser utilizadas en proyectos posteriores, debido a que son una clase de funciones genéricas cuyo fin es ser la base para crear funciones especificas con base en los requerimientos de cada una de las pruebas.

Para las siguientes pruebas del mismo tipo de productos es posible utilizar los métodos que realicé, ya que se encuentran en el lenguaje de programación que utiliza el departamento para los distintos tipos de pruebas, además de contar con toda la documentación necesaria para su implementación.

Dentro de mi desarrollo en Grupo Chamberlain conocí los procesos necesarios para el trabajo colaborativo, así como el seguimiento de cada uno de los proyectos. El equipo de *Test Automation* tiene una serie de procesos específicos para llevar a cabo el seguimiento de los proyectos, generalmente se asigna un *Scrum Master*, quien se encarga de revisar el avance de trabajo, así como dar solución a los agentes externos que puedan interrumpir o entorpecer el desarrollo de actividades del equipo.

Las metodologías ágiles tienen como objetivo reducir y flexibilizar los tiempos de entrega durante el desarrollo de proyectos, de manera específica para este proyecto el *Scrum Master* jugó un papel importante para definir de manera estandarizada los entregables según las tareas asignadas a cada integrante del equipo, esto permitió llevar un control de las tareas y en caso de existir algún problema durante su desarrollo, reasignar las tareas o resolver los problemas específicos dentro del tiempo planeado.

2. Automatización de Pruebas

Las pruebas automatizadas son un método de pruebas de software que aprovecha las herramientas de automatización para controlar la ejecución de una serie de instrucciones o pasos sin utilizar intervención humana durante el proceso. Posteriormente, estas realizan una comparación entre las entradas y salidas del sistema según los resultados esperados. Las pruebas automatizadas ofrecen resultados con mayor eficiencia y reducen el tiempo previo a la comercialización de un producto. [3]

Existen distintas formas de clasificar las pruebas automatizadas, gran parte de los escenarios de prueba parten de pruebas que se realizaban de manera manual y que debido a sus condiciones es posible llevarla a cabo por un programa. De manera general existen cuatro apartados principales en los cuales se pueden definir de manera precisa la clasificación de las pruebas automatizadas.

2.1 Pruebas Unitarias – Unit Testing

Este es el primer nivel de pruebas que generalmente es utilizado durante la etapa de desarrollo del software que está destinado a ser probado. Las pruebas unitarias consisten en aislar una sola parte de la aplicación del resto del software y probar su funcionamiento, al ser una prueba directa el sistema generalmente ignora API´s externas, bases de datos, o cualquier otro elemento de software que interfiera con el sistema que se quiere identificar. El objetivo principal de las pruebas unitarias es observar cómo funcionará cada componente del sistema que está siendo probado, sin interferir con el resto de este.

2.2 Pruebas de Integración – Integration Testing

El siguiente paso una vez que se ha validado y verificado el funcionamiento de los elementos unitarios del sistema, se procede a observar su funcionamiento en conjunto. Es por ello que las unidades de prueba son integradas de manera lógica de tal manera que trabajen en grupo tal y como se espera. El propósito principal de estas pruebas es evaluar cómo los módulos unitarios se comunican y trabajan juntos dentro del sistema.

2.3 Pruebas de humo – Smoke Testing

La estabilidad de un sistema dentro las pruebas automatizadas es un identificador que permite medir cuantitativamente que tan eficiente es un equipo o elemento de prueba, para definir qué tan estable es debe responder directamente a los criterios de falla o aprobación basados en los requerimientos de cada prueba.
Las pruebas de humo se realizan para examinar si la construcción del sistema es estable o no. El objetivo de este apartado es observar si las principales funciones del sistema funcionan correctamente y dar pie a pruebas específicas una vez validado la estabilidad del sistema.

2.4 Pruebas de regresión – Regression Testing

Las pruebas de regresión verifican que un cambio reciente dentro del sistema, generalmente de programación, no afecta a ninguna característica de este. Es decir, verifica los cambios realizados para validar que la funcionalidad del sistema permanece después de estos.

3. Automatización de Pruebas dentro de la empresa

3.1 Plan de prueba

Un plan de pruebas para validar el funcionamiento de un sistema electrónico o de software tiene como fin definir qué elementos y componentes van a ser sometidos a prueba, para que con ello el equipo pueda realizar el proceso de validación y verificación de los requerimientos del sistema. A su vez, el plan de pruebas permite llevar un control sistemático de la trazabilidad de los requerimientos, de esta manera es posible que el equipo identifique el porcentaje de avance con respecto a todos los elementos a cumplir.

Durante el desarrollo del plan de pruebas es posible obtener información sobre los errores, defectos o fallas que puede contener el sistema y con ello, es posible tomar decisiones con respecto a las correcciones pertinentes, así como los cambios que pudiera sufrir el sistema, de esta manera se asegura la calidad del producto que se está entregando.

En el caso particular de los proyectos en los que trabajé, el objetivo principal era validar el funcionamiento del equipo (sistema) en distintos escenarios. Es decir, validar las versiones de firmware de cada equipo y a partir de ello, generar reportes con la información adecuada que permita al equipo de desarrollo hacer las correcciones pertinentes en caso de ser necesarias.

3.1.1Escenarios de prueba

Los escenarios de prueba son una parte fundamental de cualquier plan de prueba, en cada uno de estos escenarios se definen los pasos cuya función es cumplir con las metas de cobertura. Es importante tomar en cuenta los recursos que se emplearán para cada uno de los escenarios debido a que implican tiempo y recursos económicos para su ejecución.

Cada uno de los escenarios se concentra de manera de manera particular en cada uno de los requerimientos y necesidades planteados en el inicio del plan de prueba. Es en este punto donde el equipo concentra esfuerzos en determinar los caminos de ejecución; flujo de cada prueba; así como, la determinación de escenarios para encontrar la mayoría de los errores.

En el caso de los productos con los que trabajamos es muy común que busquemos simular el comportamiento del usuario dentro de cada uno de los equipos, es por ello que de manera particular realizo las pruebas tal y como uno de nuestros clientes interactuaría con el servicio y/o producto. Esto me permitió conocer el flujo de los pasos a seguir y con base en ello determinar los elementos necesarios para dar cobertura a los requerimientos y necesidades.

3.1.2 Objeto de prueba

Es esencial para el desarrollo de los escenarios conocer el equipo o software el cual será sometido a las pruebas, esto implica conocer las características y elementos que conforman el sistema que se pondrá a prueba.

En el caso de pruebas dirigidas a software es necesario analizar las características del programa, así como las herramientas necesarias para realizar las pruebas, del mismo modo es necesario analizar con equipo físico si son necesarios elementos adicionales para establecer la

comunicación necesaria entre el dispositivo y el equipo de cómputo con los escenarios de prueba.

3.2 Software utilizado

Existen distintas herramientas dentro del mundo de la automatización que ayudan a desarrollar la ejecución de los distintos tipos de pruebas, estos softwares se encuentran en distintos lenguajes de programación y generalmente requieren código adicional y librerías para completar la lógica y en su caso, adquirir información adicional.

3.2.1 Selenium

Selenium es una herramienta de código abierto que automatiza los navegadores web. Ofrece una interfaz única que permite escribir scripts de prueba en lenguajes de programación como Ruby, Java, NodeJS, PHP, Perl, Python y C#, entre otros. Un controlador del navegador ejecuta estos scripts en una instancia del navegador en su dispositivo.[4]

Selenium es uno de los drivers más utilizados para el control de interfaces web, cuenta con una variedad de librerías de apoyo que potencian su funcionamiento dentro del mundo de la automatización de pruebas.

Dentro de la variedad de librerías que incluye Selenium se encuentran aquellas destinadas a la localización de elementos web dentro de las interfaces de los sitios de internet. Del mismo modo es posible intervenir de manera directa con dichos elementos, por ejemplo, agregando texto, seleccionado cajas de contenido y por su puesto dar clic sobre botones o elementos en los cuales se puede interactuar.

Con Selenium es posible generar reportes de cada una de las pruebas realizadas, esta liberaría es capaz de añadir contenido que explícitamente queramos mostrar en cada uno de los pasos del escenario, además de capturas de pantalla del dispositivo de prueba. Al final se muestra un archivo HTML con dicha información.

En mi caso, utilicé Selenium como la herramienta principal para diseñar scripts de prueba dentro de la interfaz gráfica de gran parte de las unidades físicas con las que he trabajado, del mismo modo utilicé esta herramienta para ejecutar pruebas dentro de la plataforma web donde se aloja el dashboard de los equipos de la compañía, es así cómo fue posible simular todo el proceso que el usuario realiza.

Para diseñar los scripts primero realizaba la secuencia de pasos manual, es decir que desarrollaba cada uno de los pasos del escenario dentro de mi navegador. Una vez que realizaba dichos pasos y según el plan de pruebas determinaba el criterio de falla o aceptación de dicha prueba, con base en ello obtenía los identificadores xpath de cada uno de los elementos necesarios para cada paso. En ciertas ocasiones, con base en el estado de elementos web como botones o etiquetas mostradas en pantalla, podía determinar si el script estaba listo para continuar con el siguiente paso y así evitar que fallará por motivos ajenos a lo que buscaba la prueba.

Finalmente, añadía la secuencia de pasos a la clase de Java dentro de la definición de cada uno,

una vez programados todos y cada uno de los pasos venia la etapa de depuración, donde se comprueba si alguno de los pasos no funciona correctamente o si existe un error de programación que evite que el script funcione correctamente. Para dicho proceso, es necesario añadir mensajes dentro de la consola del sistema, así se pueden levantar banderas de estado para la validación de cada una de las funciones programadas.

3.2.2 Appium

Appium es una herramienta de código abierto para automatizar aplicaciones nativas, web móviles e híbridas en plataformas móviles iOS, móviles Android y de escritorio Windows. Las aplicaciones nativas son aquellas escritas con los SDK de iOS, Android o Windows. Las aplicaciones web móviles son aplicaciones web a las que se accede mediante un navegador móvil (Appium es compatible con Safari en iOS y Chrome o con la aplicación "Browser" integrada en Android). Las aplicaciones híbridas tienen una envoltura alrededor de una "vista web", un control nativo que permite la interacción con el contenido web.[5]

Al igual que Selenium, esta herramienta permite tomar control de manera automatizada de un dispositivo móvil con ello es posible realizar escenarios de prueba que incluyan la manipulación de un dispositivo Android o iOS. La compañía tiene un par de aplicaciones en las que el usuario puede intervenir, por ello que el uso de esta herramienta es necesario para dar cobertura a distintos escenarios de prueba.

Esta gama de productos en las que el usuario interviene directamente contiene un gran número de escenarios de prueba, no solo para verificar que el equipo funciona correctamente, sino que en distintas ocasiones el cliente encuentra errores de funcionamiento. Cuando el usuario es quien directamente encuentra fallas, es necesario simular el error y con base en ello generar los escenarios de prueba específicos con la ventaja de que se pueden reciclar pasos de otras pruebas para el mismo proyecto.

3.3 Proyecto de desarrollo

Después del entrenamiento inicial empecé a trabajar con mi primer proyecto, éste estaba orientado a generar un arnés de pruebas de un producto específico de la empresa. El *Test Harness* o Arnés de Prueba es el término para definir cada uno de los elementos del proyecto, en este caso incluye un repositorio con el código que se está desarrollando; también incluye una página dentro de la base de datos de la empresa con toda la documentación. Esta página incluye los requerimientos y especificaciones del equipo al que se le están realizando las pruebas, información sobre las versiones del producto, así como todos los escenarios de prueba a realizar.

3.4 Paquetes utilizados

Una vez que revisé toda la documentación necesaria para desarrollar el proyecto, el primer paso es generar el repositorio, en este caso trabajé con lenguaje Java y utilicé *Eclipse IDE*. Creé un proyecto maven y añadí algunos paquetes que describiré a continuación.

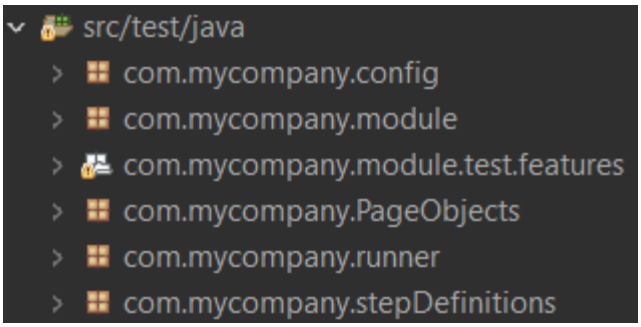

Ilustración 6 - Estructura de los paquetes principales de un proyecto maven.

3.4.1 Configuración

El primer paquete llamado *config* contiene una clase Java con toda la información para la configuración inicial de las pruebas. En esencia, este archivo contiene variables estáticas y globales que pueden ser utilizadas en otros scripts como por ejemplo la dirección IP del dispositivo de prueba, credenciales de inicio de sesión, sitios o páginas web, definición de tiempos de espera, puertos de comunicación serial, etc.

Este archivo es un elemento fundamental para mantener buenas prácticas de programación, pues permite declarar e inicializar variables globales dentro de todo el proyecto. Un ejemplo claro es evitar *magic numbers* durante el desarrollo del proyecto, esta mala práctica de programación hace referencia específicamente a utilizar valores numéricos como parámetro dentro de alguna función u operación aritmética. Dentro de esta clase es posible declarar este tipo de valores como una constante, resulta útil cuando necesitamos cambiar de producto y podemos reasignar los valores de las variables, todo dentro del mismo archivo.

3.4.2 Módulos

El paquete *module* contiene clases Java con módulos auxiliares para nuestras pruebas, en la siguiente lista se exponen los archivos en los que trabaje directamente. Estos archivos son fundamentales para el uso de métodos específicos según los escenarios a desarrollar, en este apartado describiré sus funcionalidades dentro del proyecto.

- EmailHelper.java
- ExtentReporter.java
- CLIHandler.java
- TelnetHandler.java

En esta lista sobresalen módulos como el que gestiona correos electrónicos (EmailHelper.java), entre los distintos métodos que tiene esta clase se encuentran aquellos que ayudan a obtener información específica de un correo electrónico; obtener los asuntos de la lista de correo electrónicos; verificación de correos recibidos, etc. Otro módulo importante es el de generación de reportes (ExtentReporter.java) ya que cada uno de los escenarios prueba contienen y describen los pasos de cada prueba, estos métodos pueden añadir información relevante para interpretar los resultados de las pruebas, del mismo modo, generan un archivo HTML que puede ser abierto en cualquier navegador.

Dos de los módulos que utilicé en el proyecto y que desarrollé de manera nativa son los encargados de un par de protocolos de comunicación, serial y telnet (CLIHandler.java y telnetHandler.java). Los métodos desarrollados en estos dos módulos tienen como objetivo enviar y recibir información a través de los protocolos de comunicación descritos, la utilidad de ello radica en obtener datos que los dispositivos de prueba.

Para desarrollar las clases anteriores investigué acerca de protocolos de comunicación serial, para ello fue necesario entender los requerimientos de dichos métodos de comunicación. Durante mi desarrollo académico utilice este tipo de protocolos para establecer comunicación con microcontroladores. A partir de lo anterior creé la función para inicializar la conexión, para ello la función recibe como parámetros el nombre del puerto asignado por la computadora y la velocidad de transmisión de datos.

Como funciones indispensables también desarrolle funciones para leer, escribir y cerrar la sesión. Con estos métodos es posible establecer comunicación con el equipo de pruebas de manera adecuada, durante el proceso de implementación añadí una serie de *logs* para conocer paso a paso el estado de la conexión y evitar problemas de desbordamiento de memoria.

3.4.3 Features

El siguiente paquete tiene que ver con la definición de todos los escenarios de prueba, la sección de *features* contiene las distintas categorías de los escenarios que van a ser realizados. Es en estos archivos con extensión ".feature" donde se enlistan todos los pasos de cada uno de los escenarios. Se utiliza lenguaje coloquial que en términos de Software Testing es llamado *"Behavior Driven Development"*. El objetivo de utilizar este tipo de escritura es evitar el vocabulario técnico y que cualquier persona que desee acceder a la prueba pueda entender el desarrollo de cada uno de los pasos. Cada una de las oraciones describe los pasos, sin embargo, hacen referencia a una función en Java.

Se puede realizar la analogía entre los *features* como el *front-end* del sistema, en cambio los *stepsDefinition* contienen toda la estructura de programación, es decir el *back-end* del sistema. En este proyecto en específico desarrollé escenarios de prueba con las siguientes categorías:

- AdminMode.feature
- AdminPulse.feature
- Credentials.feature
- DoorTest.feature
- EntryCode.feature
- EmailNotifications.feature
- UI.feature
- OTAU.feature
- Schedules.feature
- Videocall.feature

- AppFunctions.feature

Todos estos archivos contienen los escenarios de prueba para dar cobertura total a los requerimientos y especificaciones del proyecto. De manera general, estos escenarios buscan validar el funcionamiento en cada uno de los aspectos descritos por los requerimientos.

3.4.4 Objetos de páginas Web

El paquete *pageObjects* es un archivo indispensable para mantener un orden adecuado al identificar y clasificar los elementos de una página web. Este conjunto de archivos contiene todos los elementos con los cuales extraemos o enviamos información, este tipo de archivos de clase Java contienen los identificadores de los sitios web necesarios para los escenarios de prueba.

Para identificar los elementos de una página web existen distintas herramientas de apoyo, sin embargo, el concepto clave se encuentra en entender la arquitectura de los sitios. Los navegadores se encargan de interpretar códigos de extensión HTML o HTM, con base en ello es posible identificar elementos dentro de un sitio web.

Existen múltiples formas de identificar de forma única un elemento web dentro de la página, por ejemplo: *ID, Name, Class Name, Link Text, Partial Link Text, Tag Name and XPATH.* Un requisito indispensable a la hora de localizar un elemento dentro de un sitio web es que el identificador sea único para evitar que el programa encuentre múltiples elementos. Esto podría generar fallas sobre el funcionamiento de los scripts y evitar que la prueba pueda continuar, o en el peor de los casos, que ésta opere con errores.

```
@FindBy(id="submit_button")
      public WebElement verifyBtn;

@FindBy(css=".mat-pseudo-checkbox")
      public WebElement selectDoor;

@FindBy(xpath="//div[@id='sidebar-wrapper']//li[5]")
      public WebElement people;
```

Código 1 – Definición de elementos web a través de lenguaje xpath.

En las líneas de código se observan tres ejemplos para identificar elementos de una página web, en este caso los elementos son localizados por *ID, Css* y *Xpath.* En este paquete se encuentran definidos todos los elementos necesarios para la prueba y que serán utilizados por los *stepsDefinition*.

El paquete *runner* es la clase encargada de gestionar que escenarios van a ser ejecutados, este archivo permite configurar etiquetas que gestionan el orden de ejecución, así como las clases Java a compilar durante la ejecución del programa general.

3.5 Definición de pasos

Finalmente añadí el paquete *stepsDefinitions*, el cual contiene todas las funciones Java de cada uno de los pasos de las pruebas. Es decir, cada una de las oraciones BDD definidas en los archivos *features* tienen origen en estos archivos, toda la lógica de programación se encuentra en estas funciones. Estos archivos utilizan todos los recursos del resto de los paquetes, las distintas funciones pueden hacer uso de los elementos web, de los módulos, así como de las variables globales del archivo de configuración.

3.6 Elaboración de escenarios de prueba

Para definir los escenarios teóricamente, es necesario conocer el proceso de las pruebas, así como la cobertura necesaria para cada uno de ellos. Las pruebas que desarrollé podría definirlas en dos apartados; el hecho de que este tipo de pruebas lo defina así no quiere decir que no tengan relación o sean mutuamente excluyentes, sin embargo, con base en los recursos utilizados, el diseño de los pasos es diferente, pero al final son complemento de cada uno de los escenarios.

Por un lado, tenemos todos los pasos que tienen relación con algún dispositivo físico, es decir, aquellos que interactúan directamente con el hardware del equipo. Por ejemplo, desarrollé algunos scripts cuyo objetivo era establecer comunicación serial entre el dispositivo y mi computadora, de esta manera era posible enviar y recibir información del equipo sometido a pruebas.

El otro tipo de pasos son relativos a todos aquellos que van enfocados al control del navegador web, es decir, todos los scripts que interactúan con las interfaces HTML de los dispositivos, así como la página web donde se puede cambiar la configuración de los dispositivos.

3.6.1 Ejemplo - Verificación de interfaz de usuario

Para este escenario de prueba mostraré un ejemplo práctico a partir de la cobertura necesaria para un escenario en específico, es preciso aclarar que la imagen de la interfaz gráfica que mostraré no corresponde a la interfaz real del equipo y sólo fungirá como referencia para el ejemplo.

Escenario: Verificar que los elementos de la pantalla de usuario se despliegan correctamente.
Requerimientos: Los elementos de la pantalla de usuario deben desplegarse correctamente y se deben de encontrar habilitados. Los elementos de pantalla incluyen el reloj, mensaje de bienvenida, botón de código de acceso, botón de llamada y fondo de pantalla.

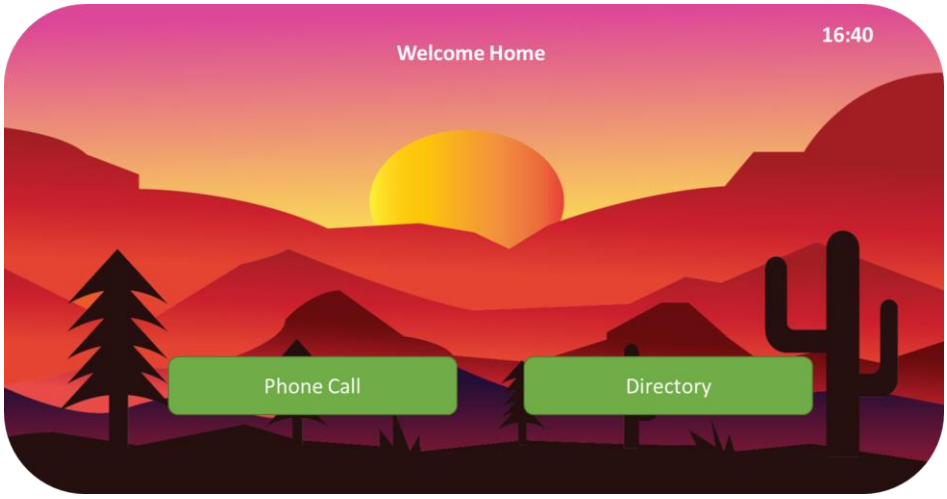

Ilustración 7 - Interfaz gráfica de una pantalla de usuario.

Criterio aceptación: Cada elemento debe ser visible y en el caso de los botones de código de acceso y de directorio deberán estar habilitados.

Para diseñar este escenario de prueba es necesario conocer la arquitectura de los documentos HTML, ya que la pantalla de usuario está basada en este tipo de lenguaje de programación. Este lenguaje utiliza etiquetas para definir los elementos que se muestran en pantalla, a su vez se apoya de atributos que permiten identificarlos con respecto al resto de todos los elementos.

Una vez que identifiqué la arquitectura de la interfaz gráfica, utilicé el navegador Google Chrome para distinguir los elementos de la pantalla. Hay distintas formas de localizar elementos dentro de una página basada en HTML.

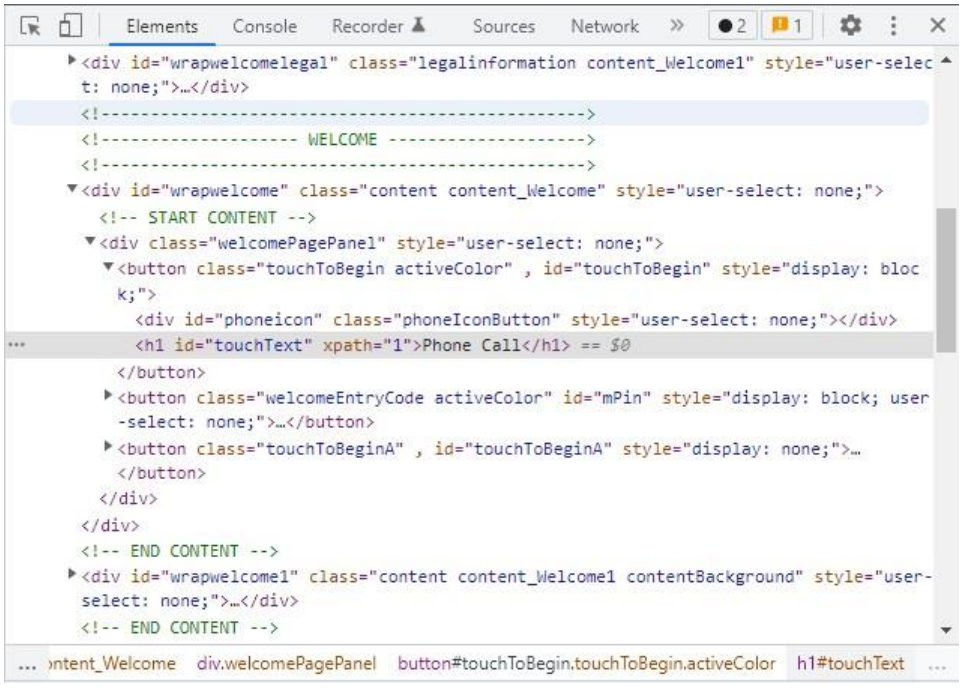

Ilustración 8 - Estructura HTML de un elemento web.

En la ilustración 8 se muestra un fragmento del código de la interfaz gráfica, el texto que se encuentra marcado con la franja gris corresponde al botón “Phone Call”. Como se puede notar, la etiqueta *<h1>* hace referencia al texto del botón mostrado en pantalla, sin embargo, dicho elemento se encuentra definido por más elementos, es en este punto donde se debe utilizar un identificador que sea único dentro de la interfaz.

El lenguaje *Xpath* sirve para localizar de manera específica uno o más elementos XML, tiene una sintaxis sencilla, Xpath se basa en un modelo de datos que interpreta el documento XML como una secuencia de elementos ordenados en una estructura de árbol. Esta estructura del modelo de datos Xpath es comparable a la del Modelo de Objetos del Documento (DOM), que opera como interfaz entre HTML y JavaScript en el buscador web.[6]

Con base en lo anterior y utilizando la extensión de Google Chrome llamada *ChroPath* identifiqué los elementos necesarios para la prueba, a continuación, enlisto los identificadores de los cuatro elementos en lenguaje Xpath:

- Botón de llamada: `//h1[@id='touchText']`
- Botón de código de acceso: `//h1[@id='pinText']`
- Reloj: `//h2[@id='clock']`
- Mensaje de bienvenida: `//h2[@id='greet']`

Ya con los elementos identificados dentro de la pantalla, inicié con la programación del script para validación del escenario, con base en los requerimientos de la prueba utilicé los siguientes pasos. (Los pasos se muestran en inglés siguiendo las reglas del BDD)

```
Scenario: Validate main screen appearance
  1. Given serial port is open
  2. When I sign in with credentials
  3. Given I navigate to standard page
  4. When I validate main screen appearance
  5. Then I get logs
  6. Then I close the port
```

El primer paso consiste en establecer conexión con la unidad de pruebas con la computadora donde se ejecutará el script, por protocolo es posible obtener información de la unidad que pueda complementar la prueba y con ello interferir sobre el criterio de falla o aprobación. En este caso, el firmware del equipo de pruebas puede enviar información del sistema en tiempo real a través de comunicación serial.

El segundo paso va de la mano con el primero, su única función es iniciar sesión dentro del producto sometido a pruebas, esto debido a que en este caso en particular, el dispositivo es una computadora con un sistema operativo derivado de Linux, por lo que debe iniciar sesión dentro de la consola.

El siguiente paso es uno bastante genérico para arnés de pruebas, ya que gran parte de los escenarios suelen utilizar la ventana principal de usuario, es por ello que se encuentra como un paso en particular, lo único que hace el script en este caso es cargar dentro del navegador la interfaz.

El paso principal es el número cuatro pues en éste se realizará la verificación que solicitan los requerimientos de la prueba, a continuación, mostraré el código implicado para este paso y posteriormente describiré a detalle el script. La función funge para validar que los elementos en pantalla se encuentran desplegados y con las características que marcan los requerimientos, es una serie de condiciones para cada uno de los elementos.

Se utiliza un par de métodos del driver selenium, la primera es para validar es si está desplegado un elemento en pantalla, este método regresa un valor booleano positivo en caso de que la condición se cumpla y un valor falso en caso de que el elemento no este desplegado.

A partir de dicho método es posible que con base en el código descrito pueda cumplirse o no la condición y con ello añadir a mi reporte de pruebas si el elemento se desplegó correctamente, del mismo modo empleo el método para validar si el elemento web este vacío, de manera análoga verifica si el elemento desplegado contiene una cadena de texto.

El paso número cinco es una función diseñada para obtener información de la unidad durante el proceso de prueba, tiene como finalidad obtener información en caso de que se presente una falla en el escenario de prueba, estos mensajes que se guardan en un archivo de texto pueden ayudar a depurar el código en caso de falla. El paso final solo es una pequeña función para cerrar comunicación serial entre la computadora y el dispositivo de prueba.

3.7 Ejecución de escenarios de prueba

Para la ejecución de los escenarios de prueba generalmente se utiliza un calendario establecido durante la planeación de las pruebas, este incluye información específica sobre la ejecución de pruebas y en qué condiciones serán inicializadas.

De manera particular y para este proyecto en específico, se cuenta con un sistema llamado *CI Server (Servidor de Integración Continua).* Este sistema tiene la capacidad de correr una serie de escenarios según se elijan, en esencia, es una interfaz que puede inicializar las pruebas realizadas y que se encuentran en el repositorio del proyecto.

Con base en lo anterior, primero ejecuté los escenarios unitarios debido a que son fundamentales para el resto de las pruebas. El argumento de lo anterior se basa a que gran parte de los escenarios comparten pasos en común, es por ello esencial que las pruebas unitarias sean correctamente ejecutadas.

Una vez que las pruebas unitarias han sido validadas según los requerimientos, procedía a ejecutar los escenarios de pruebas integrales, en este caso fue posible validar los primeros

escenarios de principio a fin. Con ello es posible analizar de manera inicial los resultados y con base en ellos verificar que los *scripts* de prueba están funcionando correctamente y si no es así, hacer las correcciones pertinentes. Para el caso específico de este proyecto realicé modificaciones menores al código para realizar un par de validaciones adicionales que servían para conocer el estado de las pruebas que tenían un tiempo de ejecución prologando.

Para la fase más importante del proyecto, las pruebas de humo (*Smoke Test*) ya tenía validadas las fases anteriores, por ello, procedí a ejecutar de manera gradual cada uno de los escenarios propuestos en el plan de pruebas según la entrega de resultados calendarizada. En ciertas ocasiones fue necesario volver a correr ciertos escenarios por fallos ajenos al sistema, es común que los servidores de los productos que se encuentren en etapa de desarrollo sean inestables y por ende no funcionen de manera adecuada.

Finalmente, y de manera usual es necesario realizar una prueba de regresión cuando se necesita una validación adicional o actualizada de las pruebas. Para este caso en particular, se ha realizado una serie de regresiones con base en las nuevas versiones del firmware del equipo.

4. Resultados

Un plan de pruebas de manera fundamental debe incluir los resultados de manera detallada no solo del proyecto sino de cada uno de los escenarios de prueba que fueron descritos y ejecutados. Para cada escenario generé un archivo HTML con la descripción de cada uno de los pasos y la información que valida o no cada uno de los requerimientos para cada prueba.

Todos estos archivos se concentran en una base de datos para consulta, estos incluyen datos de identificación como la versión de firmware en la que fue desarrollado, fecha y responsable de la ejecución. De igual modo existe una plataforma llamada TestRail en la cual subí todos los escenarios de manera nativa y cuál es el objetivo de cada uno de los pasos para cada escenario.

En cuanto los objetivos del proyecto se cumplieron en tiempo y forma, sin embargo, se encontraron ciertas atenuantes durante el desarrollo de las pruebas. Dentro de los problemas más comunes se encuentran la estabilidad del sistema de middleware, ya que en ocasiones los servicios se pausaban o detenían de manera inesperada, esto generaba que las pruebas fallaran automáticamente pues el sitio web dejaba de funcionar.

El proyecto también incluía pruebas de fiabilidad que corrían durante horas e incluso días. Este tipo de pruebas solían ser propensas a falla debido al tiempo de ejecución y la inestabilidad del sistema. En ciertas ocasiones fue necesario realizar funciones adicionales para extraer la información aún cuando la prueba fallara, de esta manera fue bastante útil conservar información y utilizarla para reportarla adecuadamente.

Como oportunidades de mejora, yo propondría solicitar un sistema de pruebas más estable sobre todo cuando se tenga que intervenir de manera directa con el sitio web donde se realiza la configuración del equipo de prueba, ya que suele entorpecer y sobre todo desperdiciar tiempo por las pruebas que se tienen que repetir.

5. Conclusiones

Desde el comienzo de mi desarrollo profesional dentro de la empresa he fortalecido mis habilidades como profesional de la ingeniería, sin duda alguna, adentrarse dentro del campo laboral brinda un panorama general sobre los conocimientos adquiridos durante el desarrollo académico y lo que el mercado está solicitando. A partir de esto pude constatar que conocer las necesidades de las empresas, hablando específicamente de las habilidades de sus empleados, ayuda a mi formación como estudiante para definir y tomar las decisiones adecuadas para insertame de manera satisfactoria en el campo laboral.

Como estudiante de Ingeniería Mecatrónica, el plan de estudios me permitió adquirir habilidades en distintas ramas de oportunidad, de manera principal y con base en mi vocación, la programación fue fundamental para mi primer contacto profesional. Sin embargo, el trabajo que desempeñé incluía tener conocimientos en diseño mecatrónico y manejo adecuado de equipo electrónico, incluidos microcontroladores. Sin duda, una gran ventaja del plan curricular de Mecatrónica es su visión integral, ya que brinda al alumno las herramientas necesarias para interactuar con distintas ramas de la ingeniería de manera satisfactoria y sobre todo útil para el mercado laboral.

Es oportuno mencionar que el desarrollo integral del alumnado puede mejorar e incluso enriquecer sus habilidades para las distintas áreas que la carrera ofrece. Desde mi punto de vista, el diseño de las materias optativas podría incluir temas y proyectos relacionados con la demanda laboral del mercado. Incluso, el hecho de brindar cursos especializados a los alumnos en ciertas áreas de interés; como ejemplo claro tenemos las materias de programación, que sin duda brindan las habilidades para conocer la arquitectura y lógica de programación, sin embargo, en escasas ocasiones es llevada al siguiente nivel, donde el alumno podría generar un producto entregable que podría incluir el desarrollo de *Back-end*, *Front-end, Fullstack* inclusive Arneses de prueba.

Como profesionales de la Mecatrónica podemos desarrollar sistemas completos que no solo incluyan desarrollo de software sino contacto directo con elementos electrónicos o diseño de elementos mecánicos. Como ejemplo claro de mi premisa, se encuentra el hecho de que para realizar pruebas a un producto de mercado es necesario conocer cómo funciona y con base en ello generar las pruebas necesarias, un Ingeniero Mecatrónico tiene una amplia visión para crear un arnés de pruebas con base en su formación académica.

Finalmente, y con base en lo anterior, mi formación académica me brindó las bases para desarrollarme de manera satisfactoria dentro de mi primer contacto profesional. Es preciso mencionar que de cualquier forma es necesario seguir adquiriendo habilidades técnicas, personales y sociales para mejorar dentro del puesto de trabajo. El crecimiento personal y profesional van de la mano con el enriquecimiento intelectual y de las habilidades como profesionales, es por ello que es fundamental conocer nuestras aptitudes para seguir creciendo de manera positiva y de esta manera llegar a un nivel de autorrealización satisfactorio, del mismo modo es necesario contribuir de manera funcional dentro del campo laboral y eso se refleja directamente con la productividad que como profesional de área necesitamos aportar.

6. Referencias

[1] *Home - Chamberlain Group*. (2019, December 19). Chamberlain Group LLC. https://chamberlaingroup.com/

[2] *About Us - Chamberlain Group*. (2019, December 18). Chamberlain Group LLC. https://chamberlaingroup.com/about-us/

[3] Atlassian. (n.d.). *Pruebas de software automatizadas para la entrega continua*. Atlassian. Retrieved May 21, 2022, from https://www.atlassian.com/es/continuous-delivery/software-testing/automated-testing#:~:text=Las%20pruebas%20automatizadas%20consisten%20en,lleva%20a%20cabo%20una%20persona.

[4] *Selenium: Definition, How it works and Why you need it*. (n.d.). BrowserStack. Retrieved May 21, 2022, from https://www.browserstack.com/selenium

[5] *Introduction*. (n.d.). Appium. Retrieved May 21, 2022, from https://appium.io/docs/en/about-appium/intro/

[6] *Tutorial de Xpath para principiantes*. (n.d.). IONOS Digitalguide. Retrieved May 21, 2022, from https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/tutorial-de-xpath/