UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Rediseño de plataforma de pruebas de productos IoT e instalación del servidor

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Axel Castillo Sánchez

ASESOR DE INFORME

Dr. Iván Vladimir Meza Ruiz

Ciudad Universitaria, Cd. Mx., 2023

Índice

1. Introducción

En las siguientes páginas describo el desarrollo e implementación del rediseño de la plataforma de control pruebas en productos de internet de las cosas (*IOT*, por sus siglas en inglés) así como de la instalación del servidor que interactuará con dichos productos. Este proyecto me fue asignado durante los meses de enero a junio de 2022, en el programa de aprendices de la empresa *Chamberlain Group (CGI)* en Nogales, Sonora, México.

Esta plataforma web surgió a partir del incremento del desarrollo de nuevos productos de manera simultánea en CGI. Nuevos productos requieren pruebas extensivas para evitar lo más posible errores en campo. Por lo tanto, esta demanda obligó a las y los ingenieros a delegar pruebas estables a esta plataforma y así, enfocarse en el desarrollo de nuevas pruebas u ofrecer mantenimiento de otras. De la misma manera, surgió la necesidad de mejorar y expandir la capacidad de servidores de pruebas para abarcar y probar más productos al mismo tiempo.

Ahora bien, es importante destacar que la empresa me ha solicitado mantener confidenciales ciertos elementos de información inherentes al desarrollo de este proyecto. Por lo tanto, en algunos capítulos no se indagará a fondo en ciertas características, desarrollos y procesos. A pesar de ello, ofreceré una clara y coherente descripción de mis actividades y resultados.

En el primer capítulo fijo los objetivos de este trabajo escrito y, además, introduzco el nicho de negocio que tiene *CGI* en Estados Unidos, así como la necesidad de traer recién egresados y egresadas de distintas partes de México para complementar su misión de ser una empresa líder en su área.

En el segundo, establezco las condiciones que llevaron a la necesidad de crear este proyecto y puntos específicos de mejora que se pretenden implementar. Termino con la detallada descripción de la metodología ágil, usada ampliamente en las empresas de desarrollo de tecnología, que utilicé y me permitió terminar en tiempo el proyecto de forma estructurada.

En el cuarto, hago una extensa descripción del comportamiento entrada/salida de las interacciones del usuario con la plataforma de pruebas en sus distintas interfaces de usuario, así como el almacenaje de la información y la interacción entre plataformas por medio de una API.

En el quinto, hago mención de la experiencia adquirida en el trabajo, así como la que recibí como estudiante de la Facultad de Ingeniería y las oportunidades actuales que tengo.

Con el Apéndice A, complemento el informe de actividades profesionales con el trabajo final que realicé para la materia de *Temas Selectos de Ingeniería de Diseño* en donde hice la propuesta y caso de análisis para mejorar la plataforma de pruebas de software por medio de la metodología TRIZ, la cual se caracteriza por ser implementada en proyectos tangibles.

Por último, que este trabajo quede como referencia para alumnos y alumnas de últimos semestres de la Facultad de Ingeniería, así como de sus profesores, profesoras y autoridades, para demostrar lo efectivos que son los resultados que ofrece la relación Universidad Pública/Sector Privado en sus futuros egresados. Así como, la importancia de la incentivación del inglés como segundo idioma en las carreras de ingeniería.

1.1. Objetivos

En este reporte tengo fijé dos objetivos principales:

- Explicar el proceso de diseño desde el entendimiento del problema, propuesta de soluciones y cómo estas se ven afectadas en la implementación por las decisiones/necesidades del negocio en cuestión.
- Experimentar con las capacidades de la metodología TRIZ en un proyecto “no físico” como lo es este, que está principalmente orientado a software y comparar resultados teóricos con reales.

1.2. Empresa Chamberlain Group

Chamberlain Group es una compañía estadounidense que ofrece soluciones inteligentes para acceso residencial y comercial. Se encarga desde el diseño hasta la manufactura de sus productos, los cuales consisten principalmente en abridores[1] de puerta para garaje (Figura 1), abridores de puerta comerciales (Figura 2) e intercomunicadores de entrada inteligentes (Figura 3). Estos productos se encuentran distribuidos en el mercado, principalmente estadounidense, bajo las marcas *LiftMaster*, *Chamberlain*, *Merlin* y *Grifco* [1].

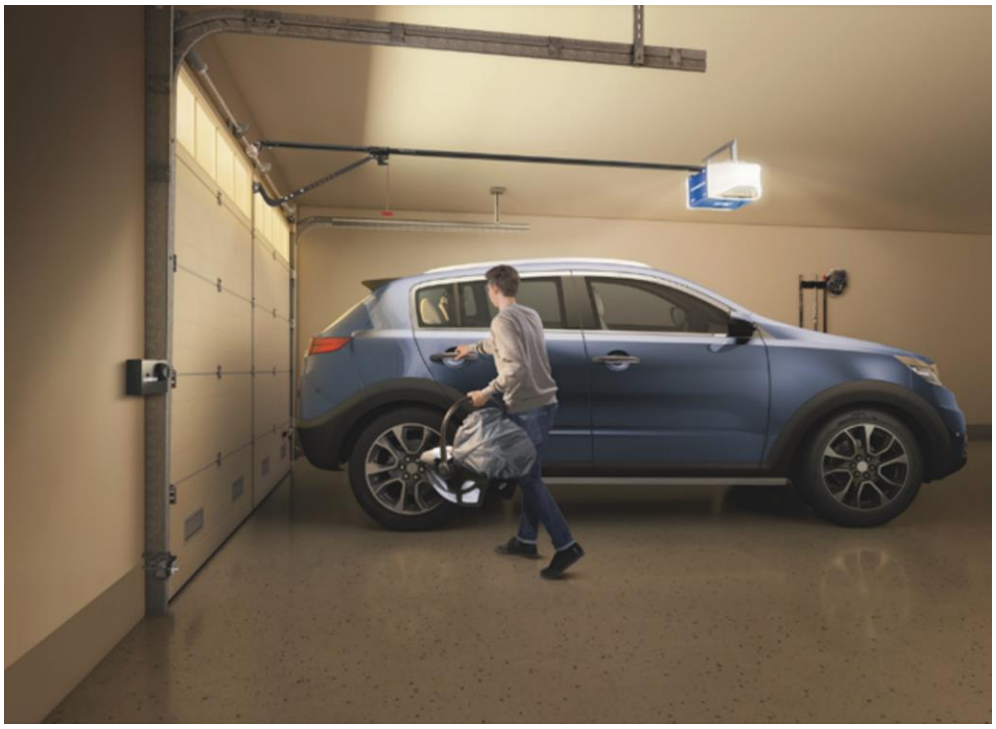

Figura 1. Abridor de puerta de garaje B2405 controlado por bandas [Fotografía], por Electronic House, 2018 (Tomado de https://www.electronichouse.com/home-security/chamberlain-increases-garage-security-with-the-ultimate-security-bundle-garage-door-opener/)

[1] Un abridor de puertas de garaje es un dispositivo mecatrónico en el cual coexisten elementos mecánicos, eléctricos y microcontroladores que controlan la apertura y cierre de una puerta.

Figura 2. Abridor de puerta comercial DDO8900W controlado controlado por cadenas [Fotografía], por Cision, 2019 (Tomado de https://www.prnewswire.com/news-releases/liftmaster-introduces-dock-door-operator-with-cloud-based-access-and-dock-management-capabilities-300897113.html)

Figura 3. Intercomunicador con video para acceso residencial *CAPXL* [Fotografía], por *LiftMaster*, S/F (Tomado de https://www.liftmaster.com/smart-video-intercom-l/p/CAPXLVMC).

Desde hace algunos años sus productos han tomado una nueva dirección, y han apostado por el control y monitoreo remoto, ya que ahora cuentan con conexión a internet y cámaras que les permite enlazarse a la aplicación web y móvil *myQ*. Con *myQ* los usuarios y usuarias pueden recibir videollamadas iniciadas desde el intercomunicador; controlar la apertura y cierre de sus puertas; recibir notificaciones de apertura/cierre y detección de movimiento (personas, carros); administrar sus productos y generar pines temporales para visitas.

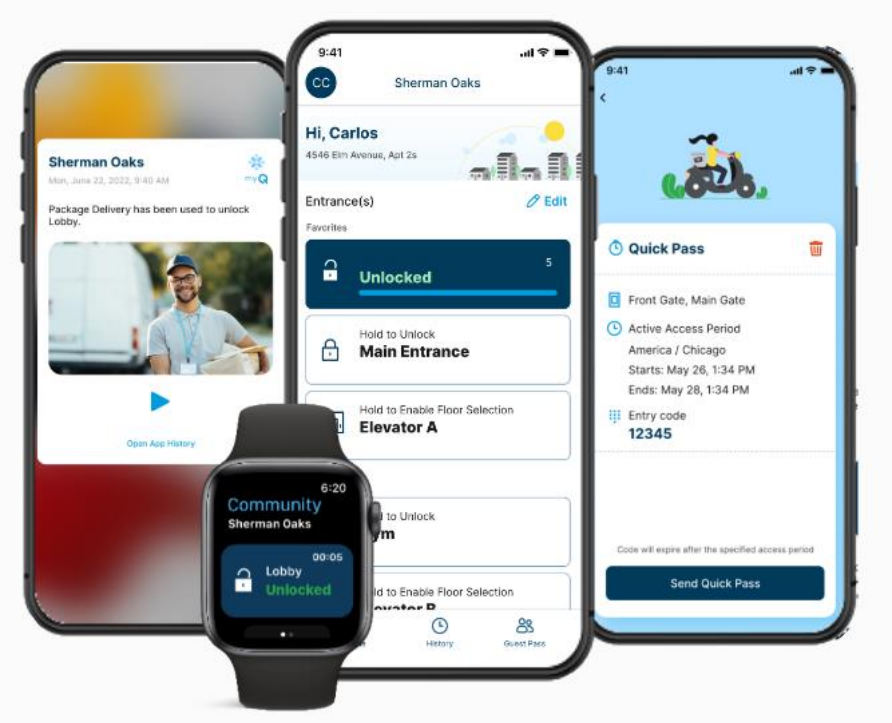

Figura 4. *myQ app* no solo permite la apertura y cierre de puertas de garaje de manera remota, sino que también permite controlar varias puertas en distintas ubicaciones, asignar pines de entrada temporales para invitados y para repartidores de paquetería. Todo desde un dispositivo móvil. [Fotografía], por *myQ*, S/F (Tomado de https://www.myq.com/community/community-manager).

Con estas interacciones remotas, *CGI* ha podido hacer colaboraciones con otras empresas como Amazon y Walmart creando los servicios *Amazon Key In-Garage Delivery* y *Walmart+ In-Garage Delivery,* para facilitar la entrega de paquetes y/o despensa en cientos de hogares estadounidenses por medio de la generación de pines de un solo uso.

Su sede corporativa se encuentra en Illinois, Estados Unidos y también cuenta otras sedes en Canadá, Australia, Nueva Zelanda, China, Hong Kong y Taiwan[1] (Chamberlain Group, s.f.). Una

parte significativa de las operaciones de producción se encuentran en México, en Nogales, Sonora, desde 1973.

1.2.1. Design Engineering Apprentice Program

Desde 2017, *Chamberlain Group* cuenta con un programa de aprendices que plantea una transición entre la universidad y el mundo laboral en ingeniería. A los aprendices se les asigna un proyecto, basado en las necesidades de la empresa, que desarrollarán a lo largo de cinco meses mientras se les ofrece capacitación y asesorías por parte de los miembros del equipo en el que son contratados. Durante este periodo los y las aprendices interactúan en un ambiente interdisciplinario e internacional. Al final del programa, deben realizar una presentación en inglés donde expliquen los impactos y resultados de su proyecto, así como su experiencia personal, a los gerentes de las áreas y los altos mandos de la empresa.

Los estudiantes o recién egresados son principalmente de las áreas de ingeniería como la mecánica, la mecatrónica y la eléctrica-electrónica y provienen de distintas universidades tanto públicas como privadas de México. La Facultad de Ingeniería de la UNAM ha tenido una estrecha relación con CGI desde el inicio del programa y varios de sus egresados laboran actualmente en la empresa. Las áreas que acogen a los aprendices son *Sustaining Electrical*, *Sustaining Mechanical*, *Firmware*, *Packaging* y *Test Automation*.

De esta forma, gracias a este vínculo entre la empresa y la facultad, apliqué al proceso de reclutamiento que se realizó de manera remota el otoño de 2021. De enero a junio de 2022 formé parte del programa en la posición de *Test Automation – Intern* en el equipo de *Test Automation Group (TAG)* donde se me asignó el proyecto que describo en los siguientes capítulos. Al final del periodo, presenté ante los ejecutivos de la empresa el proyecto (capítulo 3) y sus resultados (capítulo 4). Además, debido a mi buen desempeño se me ofreció una oferta de trabajo para la posición de *Test Automation – Engineer I*, la cual continúo ejerciendo actualmente.

2. Antecedentes / Situación actual del problema.

Test Automation Group se enfoca en proveer servicios de automatización confiables y repetibles que ofrezcan una rápida retroalimentación a los equipos encargados del diseño de firmware (el programa que es almacenado permanentemente en un dispositivo de hardware para controlarlo (Merriam-Webster, s.f.)), y software (las aplicaciones o programas que se ejecutan en un dispositivo (Rosencrance, 2021)), de los productos de *Chamberlain Group*.

Los servicios eran generados a través de un servidor instalado en Estados Unidos. Este servidor aloja los ambientes de desarrollo integrado (IDE, por sus siglas en inglés) *Eclipse* y *Visual Studio,* usados por los ingenieros de TAG para desarrollar los escenarios de prueba. Estos escenarios son almacenados en repositorios a través de un sistema de control de versiones (*git*); por lo tanto, la edición de código se realiza remotamente y los cambios efectivos se recuperan a través de git.

El servidor también aloja a *TeamCity,* el cual es una herramienta de integración y entrega continuas *(Continuous Integration/Continuous Delivery),* que ayuda a los equipos de desarrollo a administrar y automatizar el proceso de construcción, prueba y entrega de software de manera eficiente y confiable.

TAG utiliza el ambiente de *TeamCity* para enlazar los repositorios con los escenarios de prueba de los distintos productos para así, delegar el trabajo de ejecución a una computadora que no sea la personal, y la ejecución de pruebas no detenga el desarrollo de otras. Finalmente, el servidor usa comunicación serial como puente de interacción entre las pruebas y el control y monitoreo automático de los productos.

Era comúnmente necesario modificar/ajustar parámetros o ejecutar solo algunos escenarios de pruebas, lo cual implicaba hacer modificaciones internas en el código, subirlo al repositorio y ejecutarlo. Sin embargo, *TeamCity* ofrece, a través de su *REST API* (más información en el capítulo de Herramientas Digitales), la rápida modificación de parámetros y la ejecución de pruebas específicas por medio de etiquetas. Es así como, a forma de intermediario, los ingenieros de TAG desarrollaron e introdujeron una plataforma web que facilitara estas actividades.

En esta página web era posible hacer las modificaciones en los parámetros, por medio de formularios que mandaban la información directo a *TeamCity* y que no se guardaba en ningún lado. Por cuestiones de practicidad, esta solución resultó ser suficiente debido a que eran pocas las personas que ejecutaban las pruebas, había poca demanda de ejecución de pruebas y no eran muchos los productos que TAG abarcaba para probar.

Entonces, esta plataforma se mantuvo de esta forma hasta que resultó ser inestable y contraproducente para el equipo. Después de varias reuniones y casos de prueba y error, se planteó la reestructuración de este sistema.

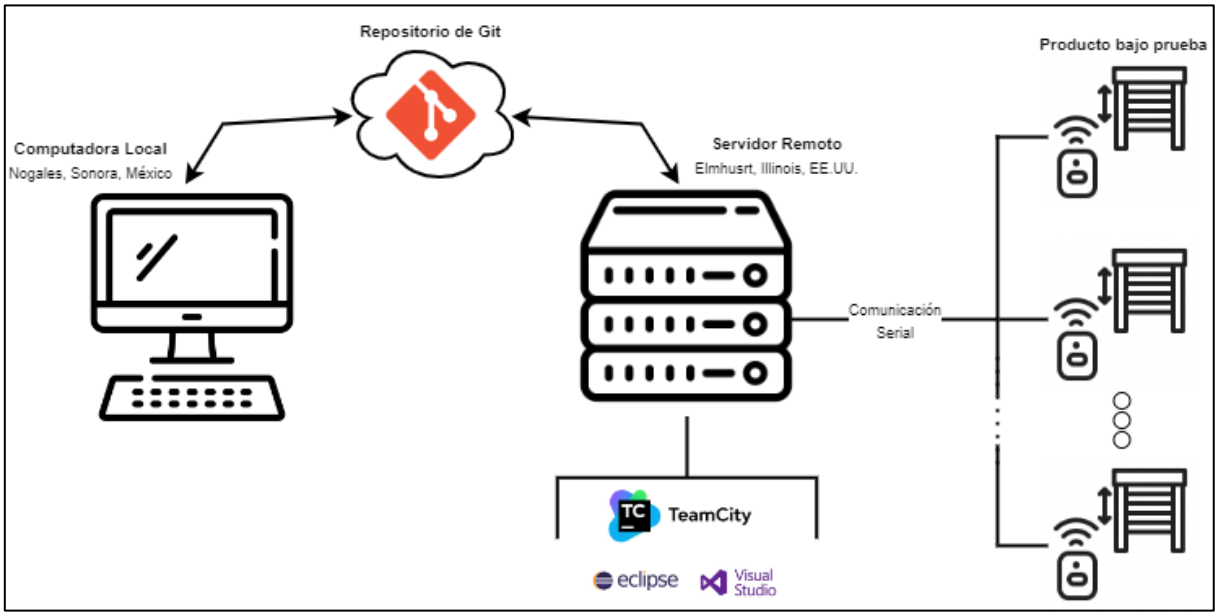

Diagrama 1. Flujo de trabajo previo a la plataforma web inicial.

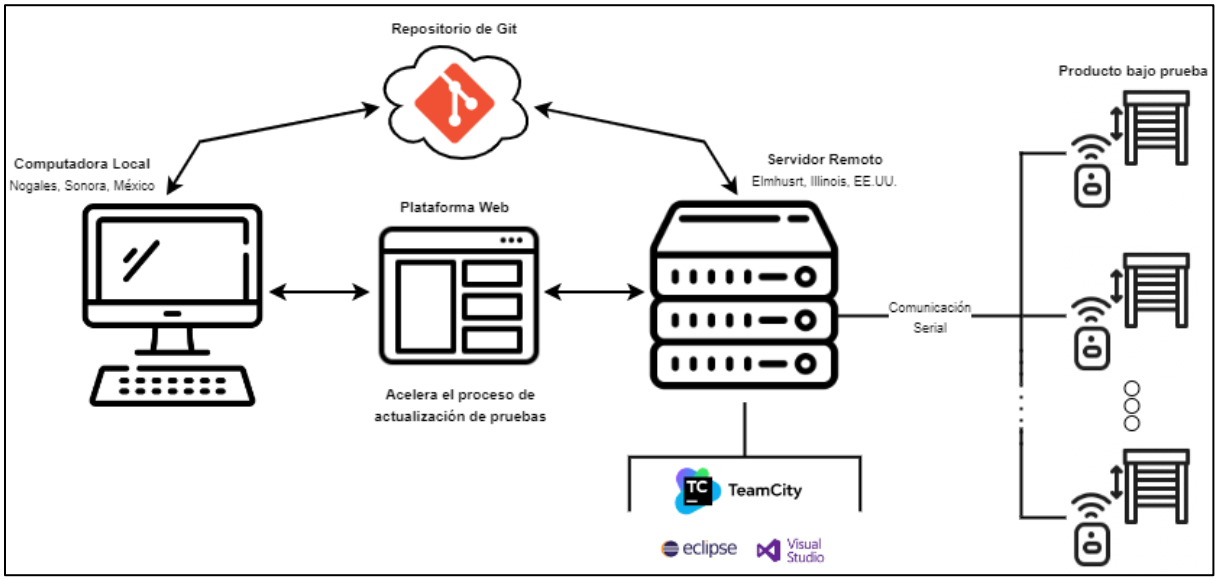

Diagrama 2. Flujo de trabajo con la implementación de la primera plataforma web.

2.1. Propuesta de mejora de plataforma de pruebas

A continuación, se enlistan los principales puntos de mejora detectados en la plataforma inicial:

- La plataforma carece de *backend*, solamente existía la interfaz de interacción con el usuario. No tenía un almacenaje ni procesamiento de la información, lo cual dificultaba la adición de escenarios de prueba de nuevos productos.
- Los reportes de resultados generados por los casos de prueba son alojados manualmente en la página web, es decir, se tiene que acceder remotamente al servidor para recuperar los archivos generados, descargarlos de manera local, crear manualmente la entrada en una tabla de la plataforma web y hacer referencia a las direcciones donde estaban almacenados.
- La página no cuenta con medidas de seguridad como un inicio de sesión. Cualquier persona que obtenga en enlace donde está alojada la plataforma web podrá acceder a todos los datos de las pruebas y ejecutarlas.
- La actual demanda de pruebas satura al servidor, alargando los tiempos de espera de los equipos que requieren la información de los resultados.

Con base en esta información se plantearon los siguientes requerimientos:

1. La plataforma será capaz de almacenar y procesar información internamente.
2. La plataforma contará con un sistema de inicio de sesión.
3. La plataforma podrá ser compartida con personas ajenas a *TAG*.
4. La plataforma almacenará automáticamente los reportes y los *logs* (registro de los eventos y actividades que ocurren en un producto) de cada prueba.
5. La plataforma podrá crear, editar, leer y borrar información de servidores, usuarios, unidades bajo prueba y escenarios de prueba.

Además, para lidiar con la saturación, se planteó hacer una réplica del servidor de Estados Unidos en la planta de manufactura de Nogales.

Entonces, con estos requerimientos se establece el camino para un software interno del TAG cuyos usuarios finales serán los miembros del equipo con el objetivo de administrar la ejecución de pruebas de mantenimiento y almacenar los respectivos productos reportes de los productos que están actualmente en el mercado como el *CAPXL* y *CAPXM*; y dar cobertura de pruebas durante el desarrollo de nuevos productos. También se contempla que usuarios externos o invitados interactúen con el sistema, no para ejecutar pruebas, sino para observar los resultados.

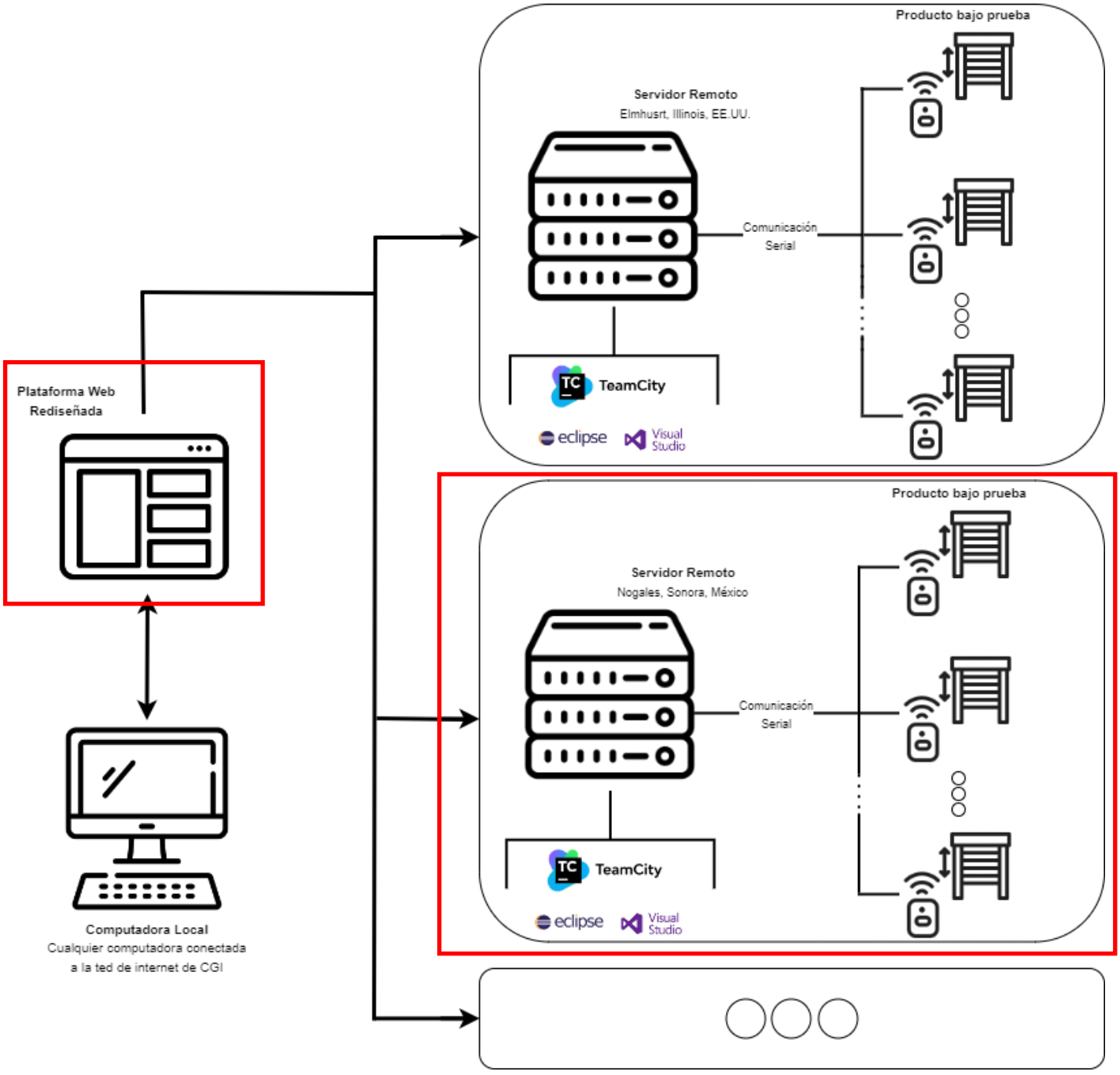

Diagrama 3. Nuevo flujo de trabajo propuesto. En rojo se muestran los módulos en los que proyecto tuvo impacto.

2.2. Metodología ágil: SCRUM

Para la finalización y entrega oportuna de mi proyecto, el equipo estructuró su desarrollo utilizando la metodología SCRUM que consiste en un monitoreo constante del progreso. por medio de las siguientes juntas:

1. Junta de planeación del Sprint: Un sprint es un periodo definido (dos semanas en este caso). Esta junta se realiza para plantear qué objetivos y actividades deberán cumplirse durante el sprint.

2. Junta *Standup*: Son juntas que se realizan dos veces por semana para revisar el avance de las actividades planteadas. Estas juntas permiten detectar bloqueos de forma temprana.

3. Junta de Demostración: Esta junta se realiza al final del sprint y se enfoca en presentar los avances obtenidos.

4. Junta de Refinación y Retrospectiva: Se realiza después de la junta anterior para discutir qué podría mejorarse, quitarse o mantener en cuestiones del desarrollo del proyecto.

En total se realizaron 9 *sprints* a lo largo de 18 semanas, donde asistí a 9 juntas de Planeación, Demostración y Refinación y Retrospectiva y 36 juntas de *Standup*. De esta manera, pude completar el trabajo de forma oportuna y antes de lo establecido.

2.3. Herramientas digitales

Una de las partes esenciales a desarrollar en este proyecto fue reestructurar el sistema existente para que pudiera ser escalable en el futuro. En consecuencia, se decidió implementar el patrón de diseño Modelo-Vista-Controlador (MVC) en la página web. El patrón separa las diferentes responsabilidades de la aplicación en capas claramente definidas y se describen a continuación: modelo (capa encargada de la representación y manipulación de datos de la aplicación), vista (capa encargada de la presentación de los datos al usuario) y el controlador (capa encargada de coordinar la interacción entre las capas anteriores, son operaciones que manejan las solicitudes del usuario) (*MDN Web Docs*, 2022).

Se decidió implementar este patrón debido a que permite modificar las diferentes capas de la aplicación de forma independiente. Además, resultó práctico ajustarlo a lo ya existente y construir las capas de modelo y controlador alrededor de la capa de vista preexistente.

En los siguientes párrafos se describirán con más a detalle los componentes de cada capa, así como las herramientas que se usaron para desarrollarlos, basándonos en la analogía de que la capa modelo es el *Backend*, la capa vista es el *Frontend* y la capa controlador es la comunicación entre las copas anteriores por protocolos *HTTP*.

El *Backend* es la parte de una aplicación web o software que no es visible o accesible para los usuarios del sistema y se encarga de almacenar y recuperar datos, así como de procesar la lógica del sistema (*Merriam-Webster*, 2023). Generalmente se ejecuta en un servidor y se comunica por medio de protocolos cliente-servidor (como el de HTTP) con el *Frontend (Herramientas Web, s.f.),* el cual, es la parte que se muestra al usuario y con la cual puede interactuar en el navegador web (*Merriam-Webster*, 2023).

Para este proyecto se utilizó el gestor de bases de datos *MySQL* para almacenar información de usuarios, servidores, pruebas e información de unidades bajo prueba. Estas bases de datos eran administradas a través de una conexión entre *PHP* y *MySQL* con la cual se podían realizar las cuatro operaciones básicas *CRUD* (acrónimo en inglés de *Create, Read, Update, Delete*): crear, leer, actualizar y eliminar registros.

Los procesos individuales de CRUD involucran la interacción del usuario en los formularios (llámese formulario a la serie de cajas de campos de entrada, *checkboxes,* selecciones, botones, etc.) del Frontend. Es en esta parte, donde hay una nueva interacción entre JavaScript y PHP.

JavaScript es un lenguaje del alto nivel y orientado a objetos que se utiliza en el desarrollo web; entre muchas de sus utilidades, se emplea para la creación de efectos visuales interactivos, manipulación de documentos *HTML* y *CSS*, validación de formularios y procesamiento de datos del lado del cliente (*FutureLearn*, 2022).

Por lo tanto, *JavaScript* se utilizó para validar la información ingresada por el usuario y enviarla o recuperarla del *backend* (*PHP*) a través de solicitudes *HTTP* (como *POST*, para enviar la información, o *GET*, para recuperarla). Esta información es enviada y recibida a través del formato *JSON*, el cual se caracteriza por ser un formato de intercambio de datos ligero y fácil de leer y escribir (Priyanjalee, 2020).

Además, *JavaScript* permite la interacción fundamental con *TeamCity* (*TC*), por medio de una Interfaz de Programación de Aplicaciones (*API*, por sus siglas en inglés). *Amazon Web Services* (*AWS*) define a una *API* como: los "mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos" (s.f.). Existen distintos tipos de *API*, pero la que ofrece *TC* es una *API* de *REST*, la cual es la más popular y flexible actualmente. *REST* significa transferencia de estado representacional y *AWS* menciona, además, que este tipo de API "define un conjunto de funciones como *GET*, *PUT*, *DELETE*, etc. que los clientes pueden utilizar para acceder a los datos del servidor. Los clientes y los servidores intercambian datos mediante *HTTP*". El cuál es el mismo protocolo que mencionó anteriormente. De esta manera el usuario podrá ejecutar pruebas y obtener resultados sin la necesidad de acceder o siquiera tener una cuenta de *TC*.

Por otro lado, para el rediseño y la generación de nuevas interfaces para el usuario en el *Frontend*, se utilizó *Bootstrap*, el cual es un *framework* de diseño web de código abierto que proporciona herramientas y componentes predefinidos para facilitar la creación de interfaces responsivas y atractivas (Jordana, 2022). Algunos de los elementos que proporciona son botones, formularios, tablas, alertas, menús de navegación, iconos, entre muchos otros (Fizgerald, s.f.).

Finalmente*, Bootstrap* y *JavaScript* interactúan con *HTML* y *CSS*. *HTML* es un lenguaje de marcado que se utiliza para definir la estructura y el contenido de una página web. Permite definir elementos como encabezados, párrafos, listas, enlaces, imágenes, formularios, etc. (Astari, 2023). Por otro lado, *CSS* es un lenguaje de estilo que se utiliza para definir el aspecto y la presentación visual de la pagina web. Permite definir el color, la fuente, el tamaño, la posición, el diseño y otros aspectos visuales de los elementos *HTML (*Mdn web docs, 2023*).*

3. Desarrollo

El rediseño se dividió en tres etapas para el desarrollo de software y una cuarta etapa para hardware. En orden fueron: 1. Administración de Usuarios y Servidores, 2. Ejecución de Pruebas, 3. Unidades en Prueba y Reportes de Pruebas y 4. Diseño e Instalación de Servidor de Pruebas.

En la tabla 1 se muestran las estimaciones y el tiempo efectivo que llevó el desarrollo de cada etapa. Se nota un desfase considerable en los tiempos efectivos de las etapas 1 y 2. Esto se debe al periodo de familiarización que tuve con las herramientas durante la primera etapa y, para la segunda, resultó se producida por contratiempos externos relacionados con la conexión al servidor en Estados Unidos con el que se hacían las pruebas.

Etapa	Estimación [*sprints* – 2 semanas]	Tiempo Efectivo [*sprints* – 2 semanas]
1	1	2
2	3	4
3	2	2
4	3	1

Tabla 1. Estimaciones contra tiempo efectivo de desarrollo.

Cabe destacar que antes de que comenzara con mi participación en el rediseño, mi supervisor ya se había encargado del rediseño completo de la base de datos y me proveyó con la información lista para implementarse. Esta base de datos consta de cuatro tablas esenciales: servidores, usuarios, unidades bajo prueba y reportes. A grandes rasgos, tanto unidades bajo prueba como reportes estaban asociadas a un servidor. A su vez cada reporte tenía asociado al usuario que ejecutó la prueba.

3.1. Administración de roles de usuarios y servidores.

La plataforma busca que distintas personas de la empresa interactúen con ella. Por lo tanto, se plantearon los siguientes tres tipos de roles en los usuarios donde cada uno tiene diferentes privilegios o restricciones de acceso a ciertas características del sistema:

- **Rol de Invitado**: Únicamente podrán leer y descargar los reportes de resultados generados por las pruebas. Este tipo de cuenta será principalmente otorgada a las personas que solicitan las pruebas e interesados.
- **Rol de Administrador**: Además de la lectura de los reportes, podrán ejecutar pruebas. De manera inicial, se contempla que estas cuentas sean operadas por ingenieros de TAG, pero, en etapas posteriores del proyecto, se contempla que sean otorgadas a miembros de los equipos de *Firmware*. Esto con el objetivo de acelerar el proceso de retroalimentación.
- **Rol de Super Administrador**: Enfocado únicamente para miembros de TAG, esta cuenta es para administración de la plataforma. Se podrá administrar los privilegios de las cuentas antes mencionadas, enlazar servidores, administrar pruebas y unidades bajo prueba.

		Rol		
		Invitado	Administrador	Super Administrador
Privilegios	Leer y descargar reportes	Sí	Sí	Sí
	Ejecutar pruebas	No	Sí	Sí
	Administración de Usuarios, Servidores, Pruebas y Unidades bajo prueba.	No	No	Sí

Tabla 2. Privilegios de cada tipo de usuario.

La administración de los usuarios se encuentra en la opción *Users* del menú lateral izquierdo y es visible únicamente para cuentas *Super Admin*. Al hacerle clic, desplegará la tabla de la figura 5, la cual muestra un resumen de la información de todas las cuentas agregadas. En las últimas dos columnas del lado derecho están los botones de Edición (azul) y Eliminar (rojo). Interactuando con estos botones se desplegará un formulario o una alerta, según sea el caso.

.

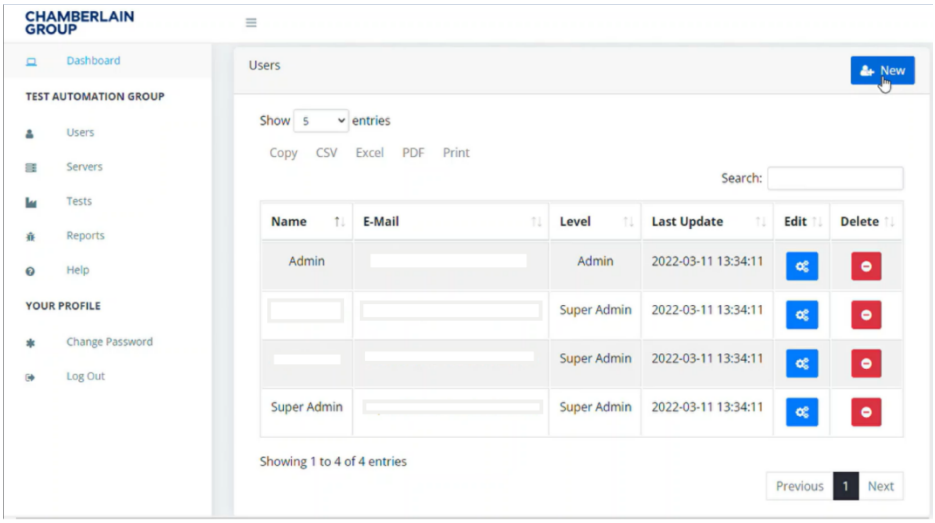

Figura 5. Interfaz de Usuarios. En la tabla contiene la información más relevante de los usuarios.

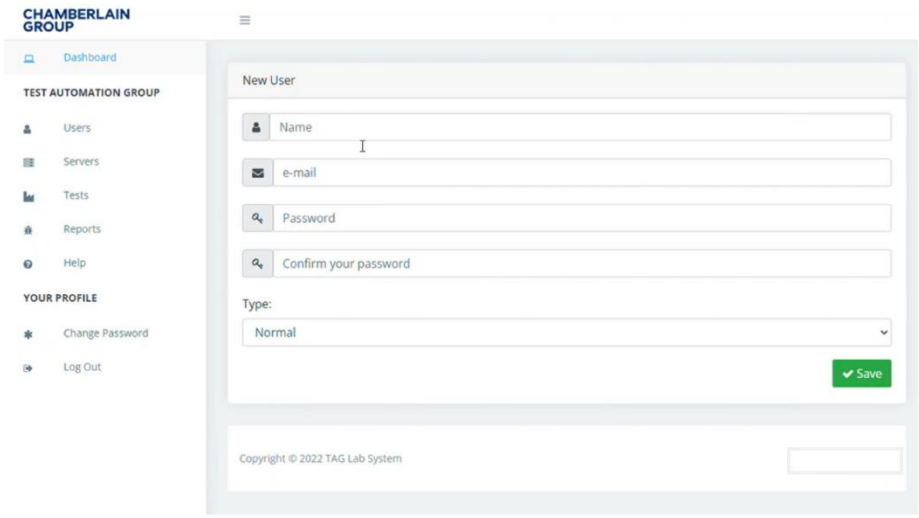

Figura 6. Formulario para la creación de nuevos usuarios.

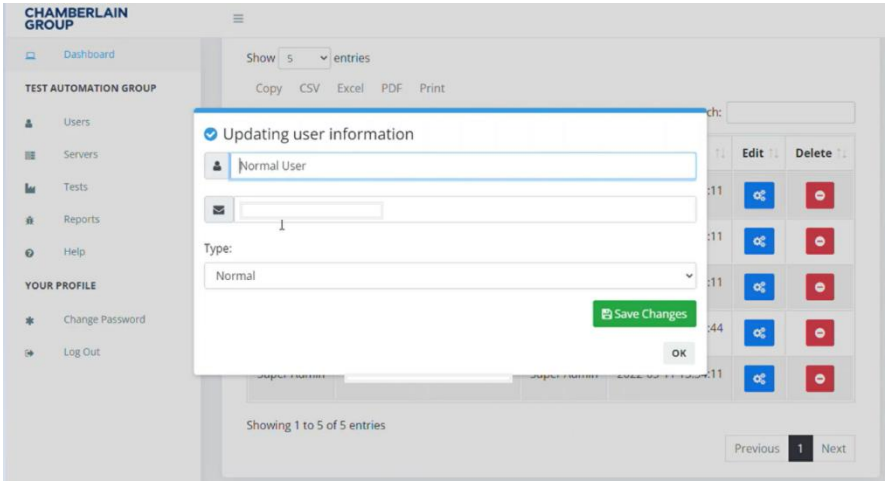

Figura 7. Edición/Actualización de nombres, correos y tipos de cuentas de Usuarios.

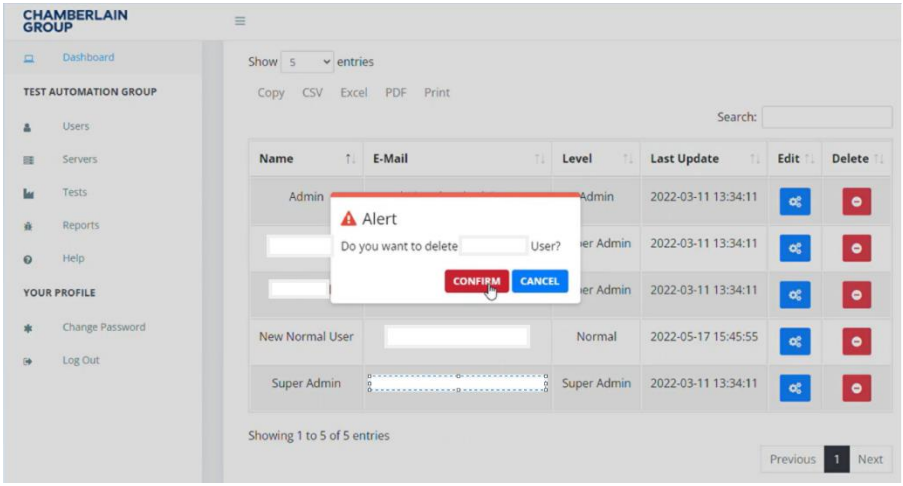

Figura 8. Alerta para confirmar la eliminación de una cuenta.

La interfaz de Administración de Servidores mantiene el mismo patrón al desplegar una tabla con el resumen de información (Figura 9), el botón de edición (Figura 10) y el de borrado (Figura 12). La principal diferencia recae en el botón verde en la columna “Test Connection” (Figura 13). Al presionarlo, por medio de la REST API de *TeamCity*, se hace una petición tipo GET para obtener los nombres de los proyectos/repositorios alojados. Si la petición es exitosa, se lanzará una ventana de alerta indicando: “Conexión Exitosa”; de lo contrario, mostrará el mensaje de conexión fallida.

Team City está alojado en una dirección IP única. Por lo tanto, cada ambiente de pruebas/servidor está asociado a esa IP. De esta forma, se pueden agregar al sistema distintos servidores/ambientes de *TeamCity* a la plataforma y ejecutar pruebas simultáneamente.

Finalmente, los usuarios con privilegios permitidos pueden agregar esta dirección IP, asignarle un Alias y para enlazar la comunicación entre la página web y el ambiente de pruebas es necesario un *Token* (una cadena de caracteres de creación única que funciona como una contraseña, pero con una fecha de expiración definida por el usuario que la creo, es generalmente usada en procesos de *REST API)* (Figura 11). Finalmente, una vez agregados los datos, es posible probar y verificar la conexión con el ambiente de pruebas.

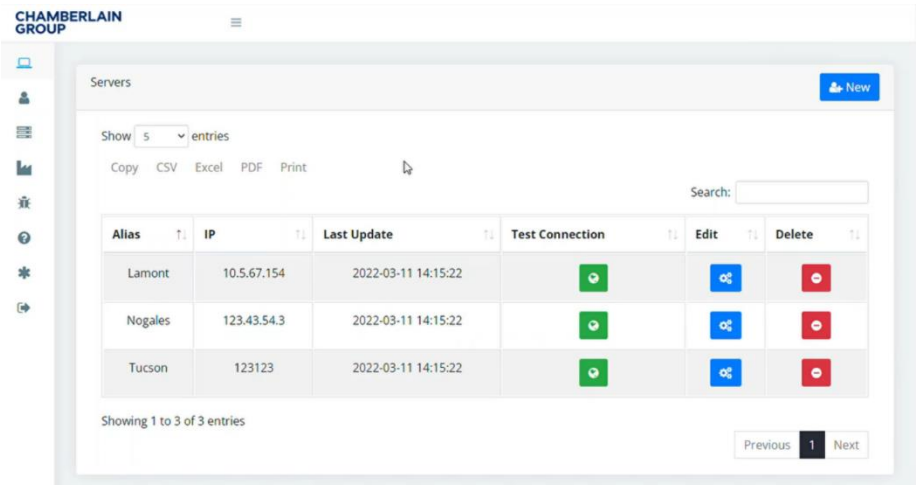

Figura 9. Información más relevante de los servidores.

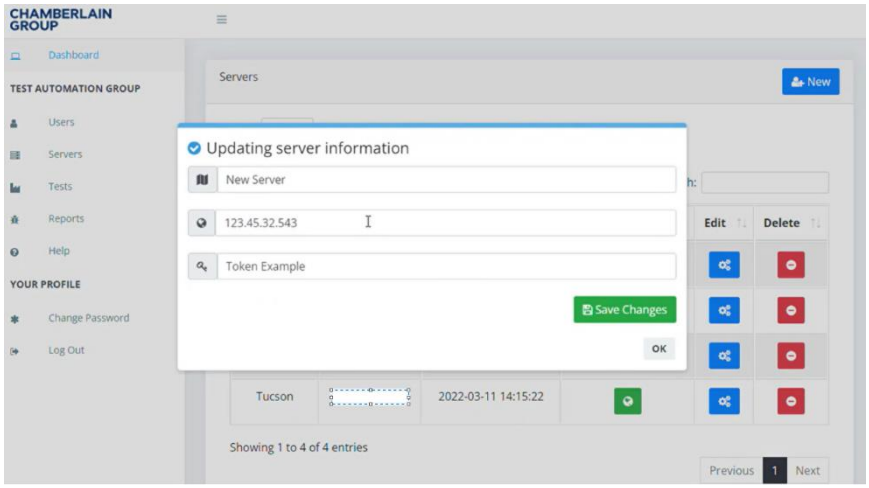

Figura 10. Formulario desplegado al dar clic en el botón *Edit* asignado a un servidor.

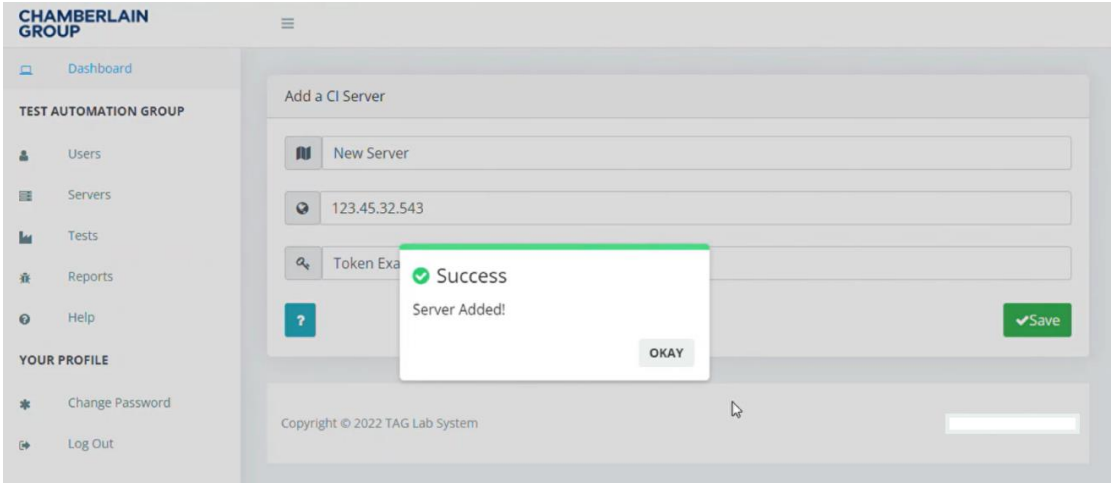

Figura 11. Alerta de confirmación después de añadir un servidor exitosamente.

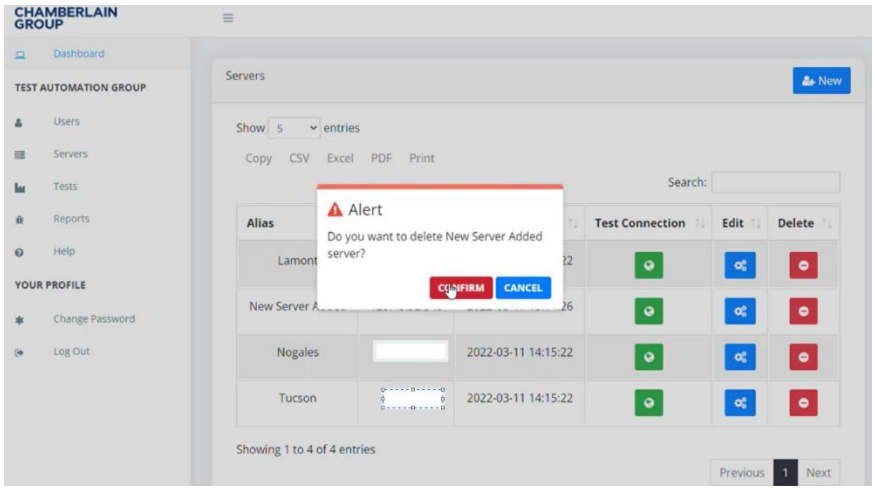

Figura 12. Formulario desplegado al dar clic en el botón *Delete*.

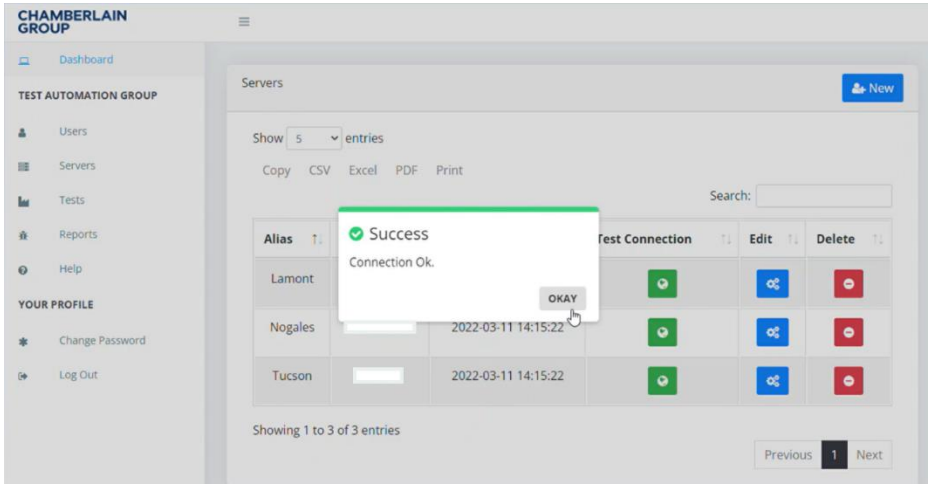

Figura 13. Alerta de confirmación de que la conexión entre el sitio web y el servidor fue exitosa.

3.2. Ejecución de pruebas

El flujo en esta interfaz comienza desde el panel de control. En él se muestran tres menús desplegables (*drop-down menus*) donde se podrá seleccionar el servidor remoto, el proyecto/producto que se probará y la rama del repositorio que contiene los escenarios de prueba (Figura 14). La información de los últimos dos menús se obtiene por medio de la REST API.

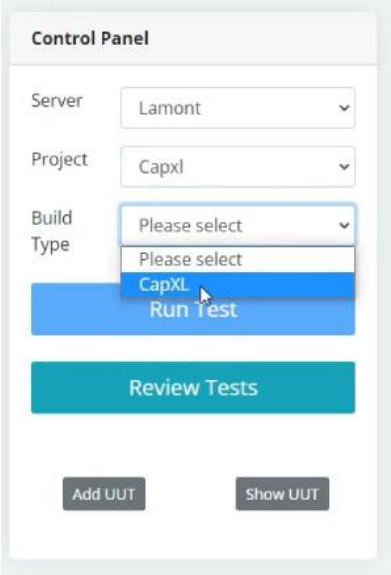

Figura 14. Panel de Control de Pruebas

Cada proyecto y rama tiene asociada una serie de escenarios de prueba únicos. Estos son recuperados de TC y se despliegan en el panel de *Test Suite,* para que el usuario pueda seleccionar uno, algunos o todos.

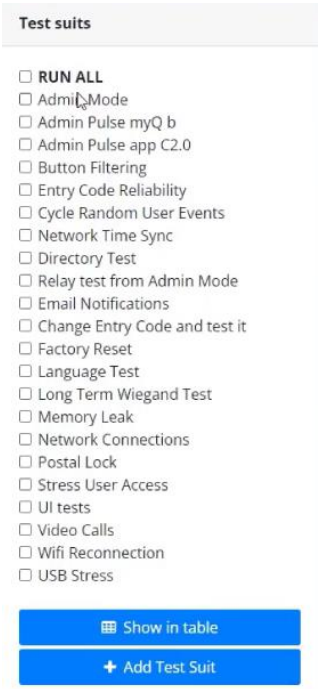

Figura 15. Panel de *Test Suites*.

Si el escenario de prueba tiene parámetros de entrada, desplegará un panel con parámetros por defecto que pueden ser editados para modificar su comportamiento.

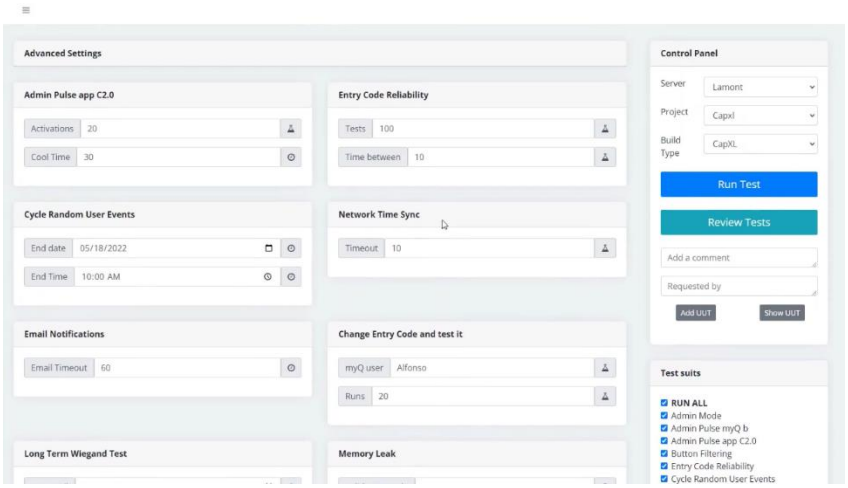

Figura 16. Parámetros de entrada de cada prueba.

En la parte superior de esta interfaz se encuentra el panel de *Configuraciones Avanzadas* (Advanced Settings), donde se pueden modificar parámetros específicos de la configuración de los productos (estos pueden ser los puertos seriales, la direcciones IP a las que hacen referencia, el número de puertas que el producto controla, el tiempo límite de espera para que el dispositivo responda, etc.) y las configuraciones propias de los celulares usados para probar la interacción entre los productos y las aplicaciones móviles, el ambiente de pruebas (producción, preproducción, desarrollo). Estos parámetros tienen también tienen valores asignados por defecto y pueden ser modificados según sea la necesidad (Figura 17).

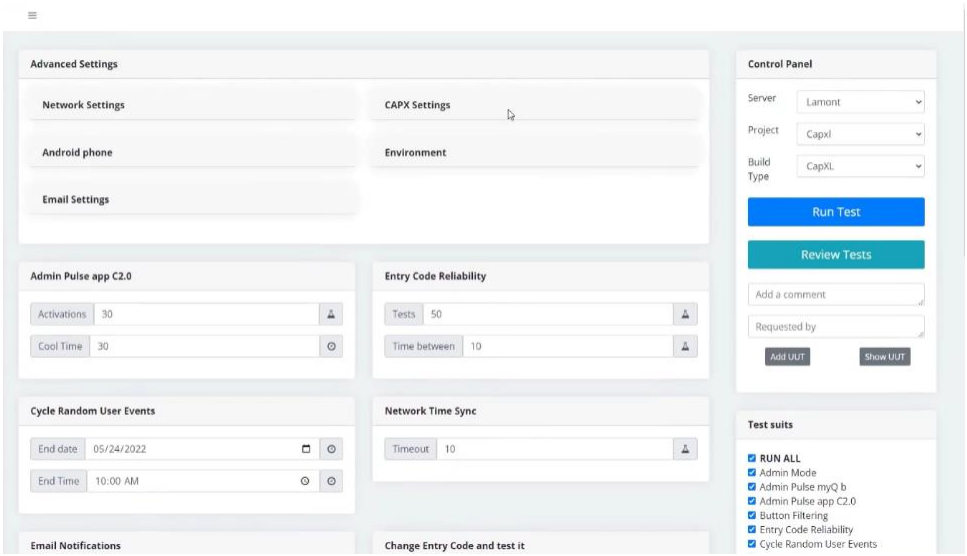

Figura 17. Panel de Configuraciones Avanzadas

Finalmente, antes de ejecutar la prueba es posible agregar un comentario y quién solicito la prueba para mantener un registro en el sistema de reportes (Figura 18).

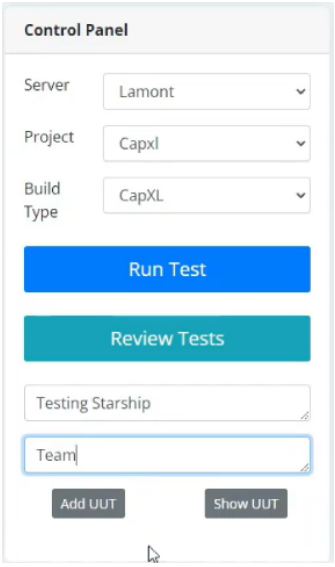

Figura 18. Agregar comentarios y especificar la persona que solicitó la prueba.

Una vez que la prueba está en ejecución es posible ver su progreso al seleccionar un servidor y después dar clic en el botón *Review Tests*. Esta interacción nos llevará a una interfaz secundaria que nos mostrará todas las pruebas que estén en ejecución (Figura 19). La información desplegada muestra el proyecto/producto, repositorio, qué usuario inició la prueba, el estatus actual y el progreso. De manera idéntica a interfaces anteriores, las últimas dos columnas cuentan con botones: el primero, permite acceder a los logs/registros de la terminal del producto; el segundo, es un botón de parado de emergencia.

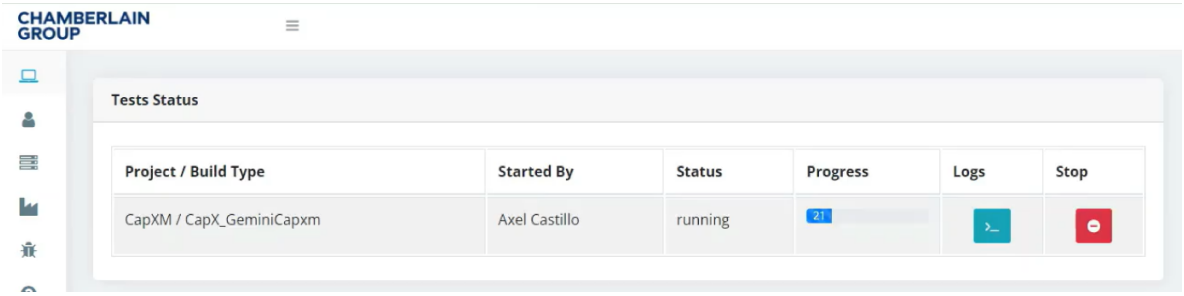

Figura 19. Resumen de la prueba en ejecución.

3.3. Unidades en prueba y reportes de pruebas.

Se pueden agregar, actualizar o eliminar unidades en la página web siempre que se hayan conectado físicamente al servidor a través de comunicación serial. Para agregar una unidad, se requiere información como el servidor al que está conectado el dispositivo, el tipo de producto, su dirección IP, el nombre del dispositivo en *myQ*, el usuario y contraseña, el puerto serial, un tiempo de espera promedio para considerar que el dispositivo no responde, el nombre de las puertas a controlar y un alias para identificarlo.

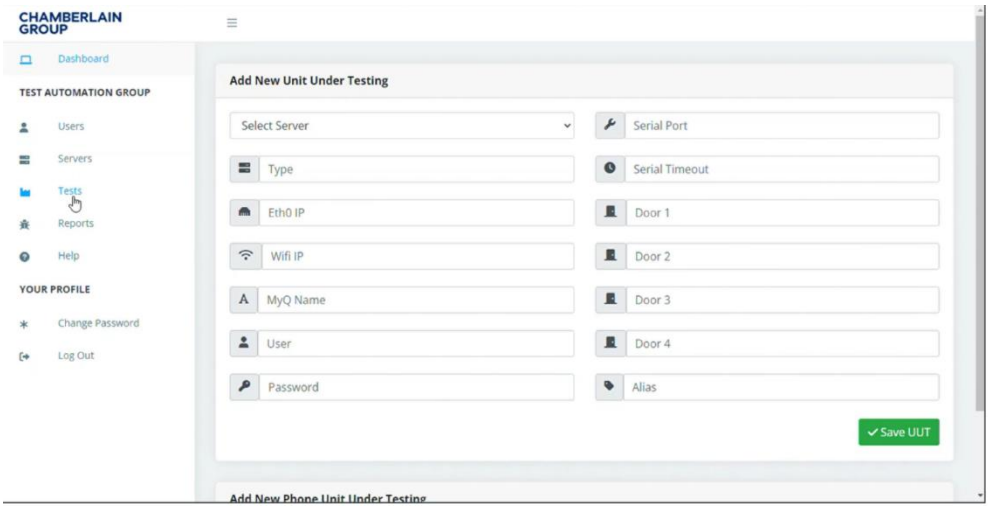

Figura 20. Formulario para la creación de una nueva unidad bajo prueba.

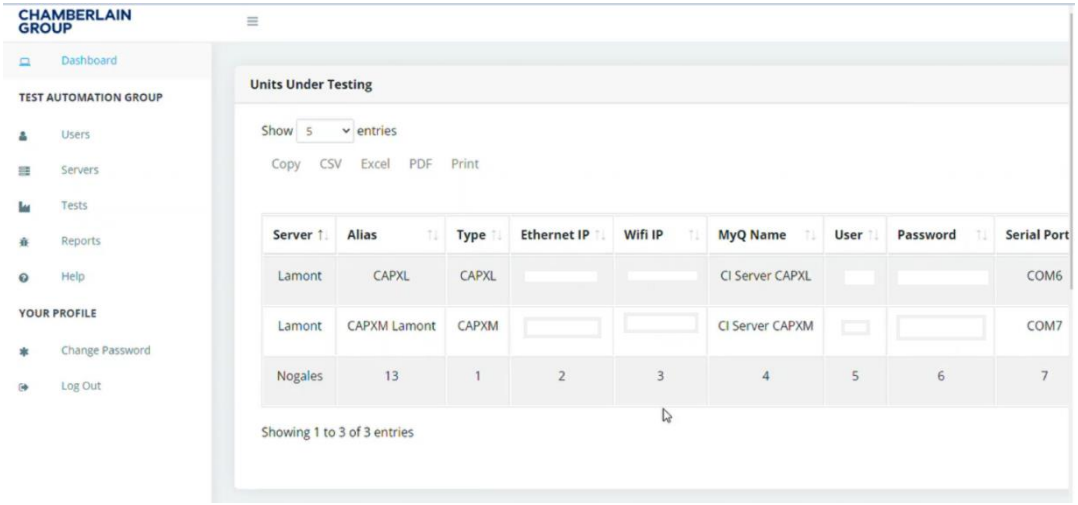

Figura 21. Información de cada unidad bajo prueba agregada a cada servidor.

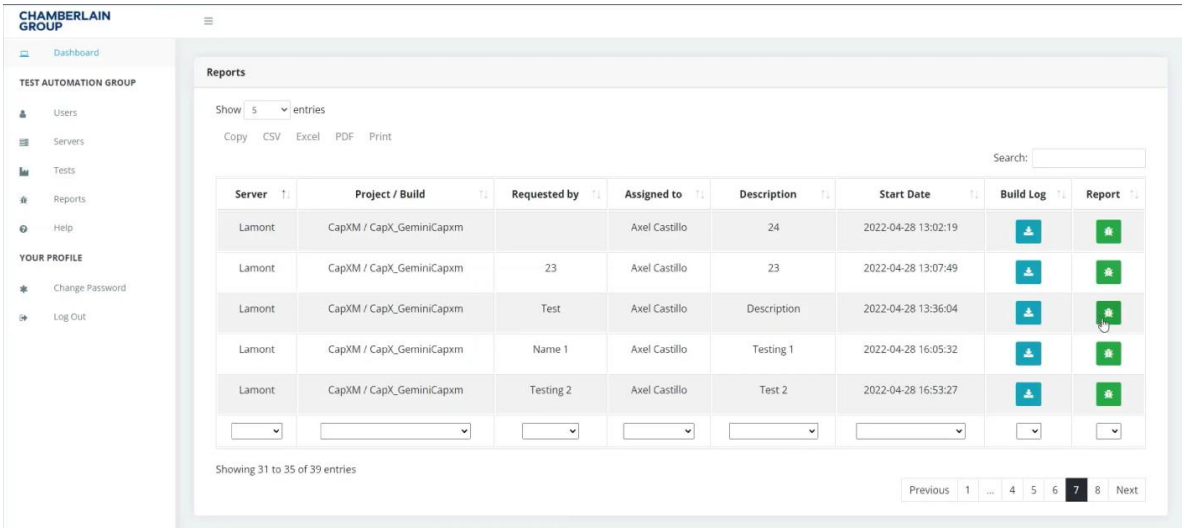

Figura 22. Panel con el resumen de los reportes de todas las pruebas ejecutadas.

Uno de los principales objetivos del proyecto es almacenar cada uno de los reportes de prueba. Después de realizar la prueba, el informe se agregará automáticamente a la tabla de reportes. Para acceder al reporte detallado con imágenes relevantes y archivos generados (como archivos csv y txt), el usuario debe hacer clic en el botón verde en el lado derecho de la pantalla.

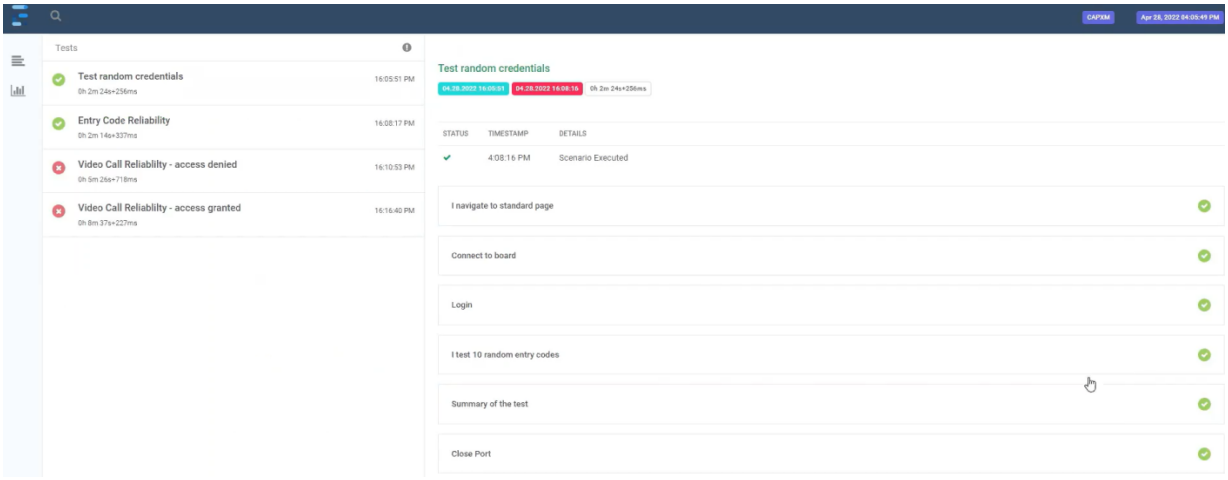

Figura 23. Reporte ejemplo de una prueba ejecutada desde el servidor.

Además, los logs/registros de la terminal del producto se pueden descargar y compartir fácilmente con el equipo interesado al descargarlos después de presionar el botón indicado.

3.4. Diseño e Instalación del servidor de pruebas.

El diseño original consiste en una computadora con sistema operativo Windows que está conectada a dos redes de internet: una externa y una interna de la compañía. La externa se usa para asemejar al ambiente en campo con los usuarios. A su vez, la computadora se conecta a un conmutador USB (*hub*) que conecta con los productos y los celulares.

Por otro lado, se tiene una segunda computadora con Fedora Server (una distribución del sistema operativo Linux) que está diseñada para funcionar como un servidor de red. En esta máquina se lleva a cabo la gestión del servidor.

Sin embargo, por decisiones internas de la empresa, la instalación del servidor tuvo que ser postergada durante un tiempo indefinido.

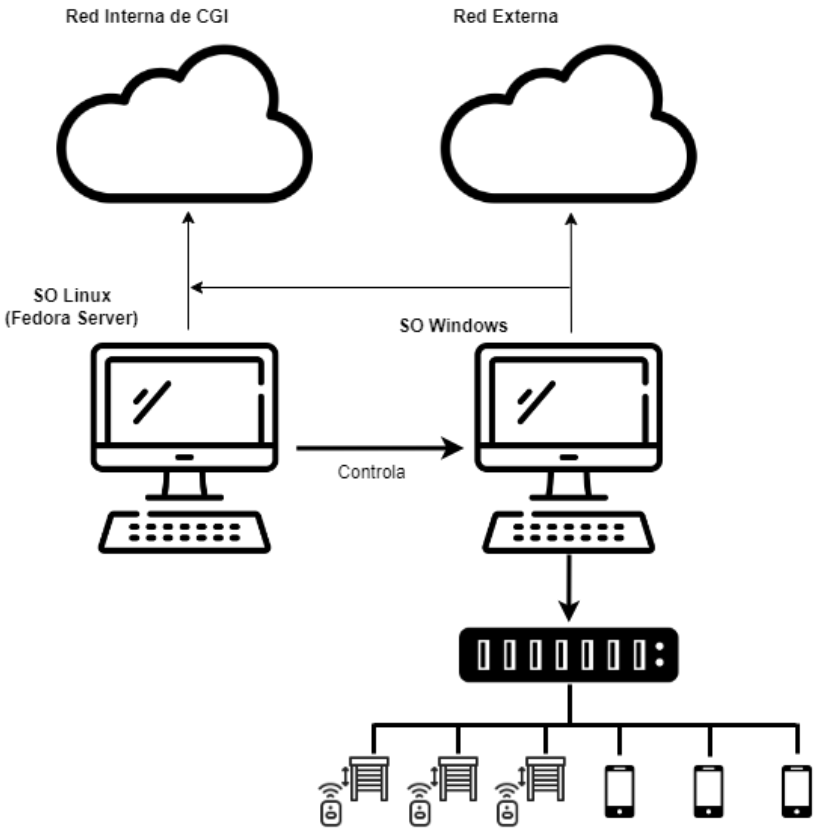

Diagrama 4. Diagrama conceptual de la instalación del servidor en Nogales.

4. Resultados

Al final del programa *DEAP*, presenté ante los presidentes de las distintas áreas de la empresa y a mis compañeros de trabajo el proceso de desarrollo de la plataforma web y sus potenciales alcances a corto y largo plazo. Además, destaqué la gran oportunidad que se le ofrece a recién egresados, como yo, de empezar a desarrollarse en la industria ingenieril y el crecimiento personal que tuve durante los cinco meses, al interactuar con un equipo multidisciplinario e internacional, así como la aventura de vivir solo en otro estado de la República.

Por otro lado, la plataforma web entró a producción en las últimas semanas de mi periodo como *intern* en *CGI*. Dió soporte a tres proyectos distintos (todos fueron de productos que actualmente están en el mercado como el *CAPXLV, CAPXM* y el *Smart Video Keypad*), en donde cuatro ingenieros de TAG y tres usuarios externos al equipo interactuaron con la plataforma durante al menos ocho meses. En este periodo, en total, se ejecutaron aproximadamente 10,000 pruebas, en donde se ejecutaron hasta tres pruebas paralelas en tres productos distintos. Debido a postergación de la instalación del servidor en Nogales, solo direccionaba las pruebas al servidor en Estados Unidos.

A la par con su introducción, se creó una página de reporte de defectos donde usuarios registraban errores inesperados en el comportamiento del sistema. Se registraron un total de 8 defectos que fueron resueltos en un periodo de un mes.

Estos resultados generaron interés en los líderes de otros proyectos y se empezaron pláticas entre ellos y TAG para escalar este proyecto. Se planea expandir la capacidad y cubrir, no solo los nuevos proyectos en desarrollo sino también dar soporte y mantenimiento a los que ya fueron lanzados al mercado.

4.1. Pasos siguientes

A casi un año del rediseño de la primera plataforma, ya se está trabajando en un nuevo rediseño. El servidor ubicado en Estados Unidos aumentará su capacidad de pruebas al poder enlazar más

productos por comunicación serial, además de que iniciará la construcción de una réplica con menor capacidad en la planta de manufactura de Nogales.

Junto con esto, ahora no solo se plantea a los nuevos servidores como una forma de delegar pruebas a máquinas remotas, sino que se pretende usar como el centro de pruebas de todos los productos. Se rediseñarán como servidores de Integración Continua/Entrega Continua, los cuales son una herramienta que ayuda a los equipos de desarrollo a de software a automatizar la compilación, pruebas, integración y entrega de código, con la ayuda de *TeamCity* como el ambiente virtual de pruebas.

El nuevo flujo de pruebas iniciará desde el momento en el que los equipos de Firmware liberan la versión más reciente del código, ya que el servidor CI/CD detectará el cambio en el repositorio, compilará y ejecutará las pruebas ya implementadas, generará los resultados y los enviará por correo a los interesados. Todo esto con el objetivo de reducir el tiempo y los errores en el ciclo de desarrollo de software.

5. Conclusiones

La experiencia laboral ofrecida por CGI fue realmente enriquecedora para mi desarrollo profesional. Estar en contacto directo con las partes interesadas, los desarrolladores de software, ingenieros de pruebas y gerentes, fue muy grato porque pude aprender cómo se lleva un proyecto ingenieril en la industria. Disfruté mucho la multiculturalidad que me ofreció (y me sigue ofreciendo) la empresa no solo internacional, sino también nacional.

Esta experiencia también sigue siendo un reto constante al tener que comunicarme completamente en inglés con mi equipo y mi jefe directo. Actualmente continúo ejerciendo la posición de *Test Automation – Engineer 1*, estoy dirigiendo el segundo rediseño de la plataforma web y el servidor. Con ese objetivo, tuve la oportunidad de hacer un viaje de trabajo a Illinois, Estados Unidos para hacer la instalación del servidor que albergará los productos a probar. Por otra parte, también soy el ingeniero líder que representa a TAG en el desarrollo de un nuevo producto, en donde genero pruebas automatizadas para verificar los requerimientos establecidos y ofrezco apoyo en un segundo proyecto también en desarrollo.

Espero que este reporte de prácticas profesionales demuestre lo efectiva que es la vinculación entre universidades públicas, como lo es la UNAM, y el sector privado. Incentivar este tipo de relaciones solo trae beneficios para ambas partes y, sobre todo, para el egresado o la egresada al adquirir experiencia en el mundo laboral. Así mismo, me parece prudente destacar que la Facultad de Ingeniería y la UNAM deben continuar con los esfuerzos de enseñanza del idioma inglés como segunda lengua.

Mi preparación como Ingeniero Mecatrónico a pesar de su inclinación al área de desarrollo de software, no se ha visto mermada en la aplicación de conceptos básicos de mecánica, pero especialmente de la eléctrica, ya que generalmente interactúo con los microcontroladores de los productos.

Sin lugar a duda, la Facultad de Ingeniería forma ingenieros e ingenieras a la altura de lo que la industria demanda. Durante mi estadía en CGI, he tenido la oportunidad de ver de primera mano

como se desenvuelven mis compañeros y compañeras egresados y recién egresados en sus respectivos ámbitos en la empresa. Manteniendo el rigor académico adquirido en la Facultad, pero también con la facilidad de mirar los problemas desde otras perspectivas para resolverlos. De un momento a otro, pasamos de estudiar los conceptos en los salones de la facultad, a aplicarlos en la industria.

6. Bibliografía

1. Chamberlain Group. (s.f.). *About Us.* https://chamberlaingroup.com/about-us/#our-businesses
2. Merriam-Webster. (22 de marzo de 2023). firmware. https://www.merriam-webster.com/dictionary/firmware
3. Rosencrance, L. (Marzo de 2021). Software. Tech Target. https://www.techtarget.com/searchapparchitecture/definition/software
4. Merriam-Webster. (19 de marzo de 2023). backend. https://www.merriam-webster.com/dictionary/backend
5. Merriam-Webster. (14 de marzo de 2023). Front-end. https://www.merriam-webster.com/dictionary/front-end
6. Herramientas Web. (s.f). El protocol HTTP. https://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html
7. Mdn web docs. (9 de marzo 2023). *¿Qué es JavaScript? https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript*
8. FutureLearn. (9 de marzo de 2022). *What is JavaScript used for? 7 practical JavaScript uses.* https://www.futurelearn.com/info/blog/what-is-javascript-used-for
9. Priyanjalee, M. (30 de Octubre de 2020). *What is JSON? It is a lightweight format for data interchange.* Medium. https://medium.com/analytics-vidhya/what-is-json-it-is-a-lightweight-format-for-data-interchange-7b8ad8732b62
10. Jordana, A. (7 de diciembre de 2022). *What is Bootstrap?* Hostinger Tutorial. https://www.hostinger.com/tutorials/what-is-bootstrap/
11. Fitzgerald, A. (s.f.). The Ultimate Guide to Bootstrap CSS. https://blog.hubspot.com/website/bootstrap-css
12. Astari, S. (16 de febrero de 2023). *What Is HTML? Hypertext Markup Language Basics.* Hostinger Tutorials. https://www.hostinger.com/tutorials/what-is-html
13. Mdn web docs. (23 de febrero de 2023). *What is CSS?* https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS
14. Amazon Web Services. (s.f.). *¿Qué es una API?* https://aws.amazon.com/es/what-is/api/
15. Mdn web docs. (5 de diciembre de 2022). *MVC.* https://developer.mozilla.org/es/docs/Glossary/MVC
16. Wikipedia, The Free Enciclopedia. (S/F). TRIZ. Recuperado el 15 diciembre de 2022 de https://en.wikipedia.org/wiki/TRIZ

A. Apéndice: TRIZ aplicado a software

Poco después de haber concluido el proyecto y mi periodo como *intern* en *CGI,* cursé la materia de *Tema Selectos de Ingeniería de Diseño* donde aprendí a implementar la metodología TRIZ (a continuación, descrita). Me pareció una gran oportunidad realizar como proyecto final el análisis del rediseño de la plataforma de pruebas ahora utilizando esta metodología y comparar los resultados teóricos con los reales mencionados en capítulos anteriores. La principal razón por la que me interesó realizarlo es que TRIZ está diseñada para proyectos ingenieriles tangibles, no para software. Esto me llevó a realizar una investigación acerca de los límites que puede tener esta teoría en otros aspectos de la ingeniería como lo es el software. En el siguiente capítulo me dedico a ofrecer un antecedente a futuros alumnos de cómo implementar la metodología y qué tan efectiva puede ser para software.

Durante la implementación de esta metodología, me tomé la libertad de agregar requerimientos que fueron descartados, por cuestiones de tiempo, del desarrollo del rediseño en *CGI* y parafrasear algunos otros para enriquecer este análisis.

A.1. Introducción a TRIZ

La Teoría para Resolver Problemas de Inventiva (TRIZ, por sus siglas en ruso), surgida a partir de la extensa investigación y recopilación de patrones en patentes de diferentes disciplinas de la ingeniería por Genrich Altshuller y sus colegas durante el siglo XX, ha demostrado ser un algoritmo práctico que al aplicarse genera nuevas ideas y soluciones para la resolución de problemas [1].

Durante sus investigaciones se encontraron tres puntos principales [2]:

1. Los problemas y sus soluciones se repiten en todas las industrias y ciencias.

2. Los patrones de evolución tecnológica también se repiten en todas las industrias y las ciencias.

3. Las innovaciones se basan en el uso de conocimientos científicos fuera del ámbito en que se han desarrollado.

Apelando al primer y segundo hallazgo, me parece que es posible y plausible extender esta metodología a una industria “nueva” como lo es la industria del software. De esta forma, encontré que Daniel Kluender en su artículo *TRIZ for software architecture* [3] utilizó los principios de inventiva y la matriz de contradicciones para reestructurar un sistema de simulación de vuelo; también Mery Helen Pesantes Espinoza, en su reporte técnico de investigación de maestría titulado como: *Aplicando TRIZ en el Proceso de Desarrollo de Arquitecturas de Software,* propone complementar un proceso de desarrollo con las herramientas de TRIZ; e incluso autores como Kevin C. Rea han propuesto y publicado analogías para el desarrollo de software de los 40 principios inventivos [5] y años después Rob van den Tillart los refresca en su publicación *TRIZ and Software – 40 Principle Analogies, a sequel* [6].

Por lo tanto, me pareció adecuado guiarme de estos trabajos (en especial el de Daniel Kluender) para proporcionar un antecedente a futuros alumnas y alumnos de la materia de Temas Selectos de Ingeniería de Diseño para aplicar la esencia de TRIZ en el rediseño de una plataforma de software, el cual categorizo como un sistema no tangible.

Si bien este sistema ha sido funcional hasta el momento, se requiere un rediseño por las siguientes razones:

- Solo se puede ejecutar una prueba a la vez y sobre solo un producto, lo cual alarga los tiempos de espera de los diferentes equipos que requieren con urgencia los resultados.
- Las pruebas solo las pueden ejecutar las personas que tienen acceso a la plataforma web (generalmente el equipo de pruebas), entonces, cuando estos no se encuentran disponibles, los equipos no tienen resultados hasta que lo estén.
- Algunas pruebas requieren ser ejecutados en periodos específicos de tiempo, algunas veces, fuera del horario de trabajo del equipo de pruebas.
- La plataforma no posee con almacenamiento ni con rastreo de los resultados. Si los resultados (archivos del tipo .html) no se guardan y se ejecutan otras pruebas, se pierden.

De igual manera se plantean los siguientes puntos como elementos deseados después del rediseño:

- Facilidad de uso para usuarios externos al equipo de pruebas.
- La plataforma debe iniciar y completar las pruebas sin supervisión de algún miembro del equipo de pruebas.

A.2. Herramientas de TRIZ y adaptaciones

Antes de utilizar las herramientas de TRIZ, es necesario definir el supersistema, sistema y subsistema para analizar y resolver este problema complejo de manera efectiva. Así es posible definir sus interacciones, funciones y límites entre los componentes.

A grandes rasgos, el supersistema es el ambiente en donde el problema existe, el sistema es el problema principal para resolver y el subsistema son los componentes internos dentro del sistema que interactúan entre sí para cumplir con sus funciones y las identificaremos en conjunto como jurarquías físicas. Con base en eso, se propuso estos como:

Supersistema: Equipos externos que se comunican con el equipo de pruebas para que estos interactúen con el ambiente de pruebas (sistema) y así extraer resultados para enviarlos a quién los haya solicitado.

Sistema: El ambiente de pruebas almacena y despliega las pruebas en una página web. Además, las administra por medio de interacciones con el usuario del equipo de prueba, para que estas sean ejecutadas en un servidor externo y este servidor genere y devuelva los archivos con resultados.

Subsistema: La interfaz que despliega la administración de pruebas al usuario (f*rontend,* aquello con lo que interactúa el usuario), el servidor que ejecuta la pruebas, el ambiente que genera los archivos con resultados y la base de datos que almacena las pruebas (*backend*, aquello que ejecuta lo que el usuario solicita en el *frontend*).

A.3. Las nueve ventanas

La herramienta de las nueve ventanas permite establecer el problema en el tiempo y delimitarlo a un espacio, para así enfocarnos en lo que realmente se puede mejorar y no ahondar en áreas que resultan irrelevantes para su innovación. Pata construirla, hay que redactar la situación actual de cada una de las jerarquías físicas en la columna Presente; posteriormente se escribe en la columna de Pasado cómo se hacían antes las cosas que el sistema actual realiza (es aquí donde se entiendo cómo mejoró el sistema en el pasado) y finalmente en la columna Futuro, plasmaremos las ideas de cómo se tendría comportar el sistema para conseguir una mejora dada la situación actual. En la tabla A1 se muestra el resultado de las nueve ventanas, con base en las jerarquías físicas definidas anteriormente.

	Pasado	Presente	Futuro
Supersistema	Equipos externos hacían sus propias pruebas y si estaban saturados, solicitaban apoyo al nuevo equipo de pruebas	Equipos externos solicitan por correo o por medio de reuniones la ejecución de pruebas de productos.	Equipos externos acceden por sí mismos a la página web a ejecutar las pruebas que diseñó el equipo de pruebas.
Sistema	Las pruebas son ejecutadas manualmente en los equipos personales del equipo de pruebas y los resultados son enviados por correo.	Las pruebas son ejecutadas manualmente en una página web por el equipo de pruebas y los resultados son enviados por correo. Solo se ejecutan pruebas de un producto a la vez.	Las pruebas son ejecutadas directamente por el equipo que las solicita y los resultados se envían automáticamente por correo. Se pueden ejecutar pruebas de hasta cuatro productos a la vez.
Subsistema	Las pruebas se ejecutaban en las computadoras personales del equipo de pruebas.	Un solo servidor de ejecución se encarga de todas las pruebas de todos los productos administrados por una página web.	Un servidor principal y dos secundarios se encargan de ejecutar las pruebas administradas a través de una página web.

Tabla A1. Método de las nueve ventanas

Con esta tabla podemos entender en qué aspectos evolucionó la plataforma que se usa actualmente y qué aspectos de interés habría que enforcarse para solucionar y mejorar los problemas actuales.

De esta manera, se determina que se busca un sistema versátil que se pueda adaptar a la demanda de pruebas, así como un subsistema que se administre automáticamente para evitar la saturación de pruebas (distribuyéndolas en equipos de apoyo) para completarlas más rápido. Por último, un

subsistema que envíe los resultados a sus destinatarios para evitar la pérdida de información. La gran parte de estos puntos de interés implican modificaciones en el subsistema. La identificación de estos puntos nos permitirá seleccionar los parámetros del cambio más adecuados para este rediseño durante la siguiente fase.

A.4 Analogías para el área de software de los 39 parámetros del cambio y los 40 principios inventivos.

El método de las 9 ventanas nos ofreció unos puntos de interés con los cuales enfocarnos para implementarlo en la matriz de contradicciones. Para ello utilizaremos los 39 parámetros del cambio y los 40 principios inventivos, descritos a continuación, como marca la teoría TRIZ. Los *39 parámetros del cambio* representan un aspecto específico o característica de un sistema que puede ser modificado o sustituido para resolver contradicciones, mejorar funcionalidad y superar complicaciones en el diseño. Estos cubren un amplio rango de atributos físicos como el tamaño, forma, temperatura, presión, velocidad, energía, entre otros.

Por otro lado, los *40 principios inventivos* proveen una forma estructurada de resolver problemas al sugerir distintas formas de modificar un sistema o resolver conflictos provocados por los 39 parámetros del cambio.

Sin embargo, como menciona Kluender, si bien los principios inventivos no son directamente aplicables a software (debido a que se generaron a través de patentes de objetos tangibles), por lo menos son lo bastantes generales para hacer un mapeo cuidadoso que nos permita hacer analogías a software que aporten soluciones viables.

En un análisis comparativo entre los principios inventivos y patrones de diseño de software, Kluender concluye que se pueden hacer analogías directas cuando el principio está relacionado al tiempo, al espacio y a la estructura, pero no al material. Este punto lo extiende también a los parámetros del cambio.

Entonces, con base en esto y en los puntos de interés establecidos en las 9 ventanas se tiene la siguiente tabla:

Punto de Interés	Parámetros del cambio
Sistema adaptable a la demanda de pruebas	35. Adaptabilidad o versatilidad
Administración automática para evitar saturaciones y terminar pruebas más rápido.	38. Alcance de la automatización. 39. Productividad.
Enviar los resultados a los equipos interesados al terminar la prueba	24. Pérdida de Información

Tabla A2. Parámetros del cambio

Para los parámetros de cambio no se hizo ninguna analogía en particular porque los términos son bastante transparentes, tanto para hardware como para software.

A.5. Matriz de contradicciones

La matriz de contradicciones ofrece enfoques innovadores para superar las contradicciones técnicas en un sistema al cruzar los 39 parámetros de cambio con los 40 principios inventivos, identificados anteriormente. Consiste en una matriz que se construye al poner como columnas y filas los 39 parámetros del cambio, cada una de las intersecciones entre los parámetros puede estar vacía (ya sea que no existe contradicción entre los parámetros o que no puede existir una contradicción entre el mismo parámetro) o puede contener los principios inventivos sugeridos para resolver la contradicción entre parámetros.

Al mejorar una característica del sistema, inherentemente se deteriora otra. Un ejemplo comúnmente mencionado en clases es que si se mejora la velocidad de un vehículo se verá afectado su peso. El aspecto más importante de esta matriz es que las intersecciones de estas contradicciones se encuentran algunos de los 40 principios inventivos. Siguiendo el ejemplo anterior donde mejoramos la velocidad del vehículo mientras afectamos su peso, la intersección de estos parámetros en la matriz indica que para resolverse la contradicción (o aminorar el deterioro) se puede implementar cualquiera de los siguientes principios inventivos: 2. Extracción/Separación, 8. Contrapeso, 15. Dinamicidad o 38. Oxidación Acelerada. Una solución sencilla para concluir con el ejemplo sería utilizar el principio inventivo 2. Extracción/Separación, donde podríamos analizar el vehículo y extraer alguna parte o componente que resulte innecesario. De esta manera se mejora la velocidad y no se afecta gravemente el peso.

Entonces, la matriz de contradicciones resulta ser un balance entre mejoras y afectaciones. Siempre con el objetivo de conseguir las mejoras en los parámetros del cambio de interés y evitar el mayor deterioro posible en otros parámetros del sistema al seguir las recomendaciones de los principios inventivos.

Ahora bien, ya que tenemos definidos los parámetros del cambio o técnicos, los usaremos en la matriz de contradicciones para obtener las sugerencias propuestas. En la tabla 2 es posible ver las soluciones sugeridas por las contradicciones existentes entre los parámetros. Se busca mejorar la 35. Adaptabilidad del sistema, el 38. Alcance de la automatización y la 24. Pérdida de la información, y en los tres casos, mantener la 39. Productividad. En la última columna se tienen las 3 soluciones más apropiadas para el problema: 5. Consolidación, 6. Multifuncionalidad/Universalidad y 23. Retroalimentación. Estas fueron seleccionadas con base en los ejemplos que acompañan a las analogías de los 40 principios inventivos para software proporcionadas por Rea [5] Y Tillaart [6].

Contradicción	Soluciones Sugeridas	Solución Seleccionada
35 vs. 39	35, 28, 6, 37	6: Multifuncionalidad/Universalidad
38 vs 39	5, 12, 35, 26	5: Consolidación
24 vs 39	13, 15, 23	23: Retroalimentación

Tabla A3. Contradicciones, Sugerencias y mejores soluciones seleccionadas.

5: Consolidación

La analogía que propone Tillard para este principio es ejecutar procesos en paralelo (*Make processes run in parallel*). Y esta idea concide con lo que encontramos en el futuro del sistema en el método de las 9 ventanas. Si se instalan servidores de apoyo, es posible redireccionar algunas pruebas hacia ellos y así ejecutarlas en paralelo. Teniendo como resultado una reducción del tiempo de espera para los equipos interesados en los resultados.

6: Multifuncionalidad/Universalidad

Tillard propone hacer que un sistema técnico soporte clasificaciones múltiples y dinámicas (*Make a technical system support multiple and dynamic classifications based on context*). Con base en esto, se plantea hacer mediciones del tiempo que toma una prueba en terminarse y solicitar a los equipos fechas límite de entrega de los reportes. De esta forma, se puede implementar un algoritmo de

priorización que determine qué pruebas requieren ejecutarse primero dependiendo de los criterios antes mencionados. Así, podrá regularse automáticamente.

23: Retroalimentación

Para evitar la pérdida de información y mantener la productividad se propone agregar Retroalimentación al sistema. Este principio resulta bastante transparente al mundo del software. Uno de los ejemplos que propone Tillard es poner una barra de progreso cuando un proceso toma mucho tiempo. Para el proceso de enviar los resultados a los equipos, la retroalimentación no es tan clara porque la solución intuitiva y más sencilla es almacenar los reportes en la memoria del sistema, pero lo que sí se puede considerar es enviar un correo inicial a los equipos con el tiempo estimado que tomará en completarse la prueba y/o agregar una interfaz gráfica al sistema para que los equipos puedan consultar cómo va su prueba sin tener que contactar al equipo de pruebas.

A.6. Conclusiones

TRIZ resultó ser una metodología interesante de aplicar como ejercicio complementario para este reporte ya que permitió aprender del pasado del sistema para entender por qué se tiene la plataforma actual y qué se desearía tener en un estado futuro gracias al método de las 9 ventanas.

Con respecto a los principios inventivos y parámetros del cambio, desafortunadamente, no todos son transparentes a software y muchos ni siquiera tienen alguna analogía directa. Lo cual de momento reduce un poco la generación de soluciones. Aunque aparentemente sería un buen ejercicio plantear principios inventivos enteramente enfocados a software resulta ser una tarea compleja debido a que no todos los desarrollos de softwares son patentados. Por otro lado, sí es posible aportar más ejemplos de implementación que acompañan a dichas analogías.

A pesar de los puntos anteriores, con las analogías existentes propuestas por Tillard y Rea sí es posible generar soluciones. De las tres soluciones que se generaron en este trabajo, dos sí fueron implementadas en el sistema real (Consolidación y Retroalimentación). Cabe destacar que este trabajo se llevó a cabo meses después del rediseño real del sistema y aún así, las soluciones

arrojadas por la metodología coincidieron con la implementación real. La última, resultó ser una idea que actualmente se está discutiendo para una actualización del sistema actual.

A.7. Referencias

1. The TRIZ Journal. (Diciembre, 2011) What is TRIZ? Recuperado el 17 de diciembre de 2022 de https://web.archive.org/web/20111201172928/http://www.triz-journal.com/whatistriz.htm
2. Kluender, Daniel. (2011). TRIZ for software architecture. ELSEVIER. Volumen 9, Páginas 708-713. https://doi.org/10.1016/j.proeng.2011.03.159
3. Pasantes Espinosa, M. H. (2006). Aplicando TRIZ en el Proceso de Desarrollo de Arquitecturas de Software [Reporte Técnico de Investigación]. Centro de Investigación en Matemáticas, A.C. https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/94/2/TE%20195.pdf
4. Rea, K. (S/F) TRIZ and Software – 40 Principle Analogies, Part 1. Novalis. Recuperado de: https://novalis.org/triz-talk/softwarearticle.html
5. 21. Tillaart, R. (Enero, 2006): TRIZ and Software – 40 Principle Analogies, a sequel. The TRIZ Journa. Recuperado de: http://triz-live.kobus.eu/triz-software-40-principle-analogies-sequel/