Instituto Tecnológico de Costa Rica

Carrera de Ingeniería en Computación,
Campus Tecnológico Local San Carlos

“Integración y Migración del sistema Metricapp”

Práctica Profesional para optar por el título de
Ingeniero en Computación con el grado académico de
Bachiller Universitario

Carlos Andrés Villalobos Salas

Costa Rica, 2021

1. Resumen ejecutivo

Las soluciones tecnológicas utilizadas dentro de una empresa buscan siempre simplificar y mejorar ciertos procesos con el fin de brindar mayores beneficios generales a la empresa y sus trabajadores. En el caso del departamento de aseguramiento de la calidad de la empresa SOIN (Soluciones Integrales) existe la necesidad de utilizar las mejores herramientas disponibles en el mercado actual, herramientas que ayudan a la empresa a garantizar la calidad de sus productos.

Dicho departamento utiliza una aplicación de escritorio para controlar diariamente los estados de todos los proyectos de la empresa, una aplicación llamada Metricapp. Esta aplicación fue desarrollada en 2018 en Java y es utilizada desde entonces. Entre las herramientas que utiliza la empresa para la gestión de sus proyectos se encuentran Test Rail y Bugzilla, por lo que Metricapp ha establecido conexiones con ambas plataformas.

En el año 2021 el departamento de calidad decide migrar de Bugzilla a Jira, ya que, con previos estudios realizados internamente, se llega a la conclusión de que se obtendrían más ventajas para el trabajo diario. Luego de unos años utilizando Metricapp, se llega también a la decisión de que tener esta aplicación en un ambiente web podría beneficiar más a las capacidades de la aplicación, con el objetivo de expandirla cada vez más. Debido a esto, se desarrolla durante 19 de semanas, un proyecto de migración del sistema actual de Metricapp a la API de Jira y a un ambiente web.

Como soluciones se obtuvieron un sistema de Metricapp actual desarrollado en Java con la implementación en Jira para que el departamento pueda concluir los proyectos ya iniciados, y un nuevo sistema web de Metricapp, listo para incluir nuevos proyectos creados en Jira y nuevas funcionalidades extras.

Palabras clave: Jira, Test Rail, React, Node, JavaScript, MySQL, Calidad, Gestión.

2. Executive summary

The technological solutions used within a company always seek to simplify and improve certain processes to provide greater general benefits to the company and its workers. In the case of the quality assurance department of the company SOIN (Soluciones Integrales) there is a need to use the best tools available in the current market, tools that help the company to guarantee the quality of its products.

This department uses a desktop application to monitor the status of all company projects daily, an application called Metricapp. This application was developed in 2018 in Java and has been used ever since. Among the tools the company uses to manage its projects are Test Rail and Bugzilla, which is why Metricapp has established connections with both platforms.

In the year 2021, the quality department decides to migrate from Bugzilla to Jira, since, with previous studies carried out internally, it is concluded that more advantages would be obtained for daily work. After a few years using Metricapp, the decision is also made that having this application in a web environment could further benefit the application´s capabilities, with the aim of expanding it more and more. Because of this, a migration project from the current Metricapp system to the Jira API and a web environment is developed for 19 weeks.

As solutions, a current Metricapp system developed in Java with the Jira implementation was obtained, so that the department can conclude the projects already started, also a new Metricapp web system ready to include new projects created in Jira and new extra functionalities.

Keywords: Jira, Test Rail, React, Node, JavaScript, MySQL, Quality, Management.

3. Tabla de contenidos

Capítulo I 1
1 Introducción 1
1.2 Contexto del proyecto 1
1.3 Problema 2
1.4 Objetivos 2
1.4.1 Objetivo general 2
1.4.2 Objetivos específicos 3
1.5 Justificación 3
Capítulo II 4
2 Revisión de literatura 4
2.1 Marco teórico 4
2.2 Trabajos relacionados 6
Capítulo III 8
3 Solución planteada 8
3.1 Propuesta 8
3.2 Involucrados 9
3.3 Procedimiento metodológico 10
3.4 Análisis de los riesgos 13
3.5 Cronograma de trabajo 16
Capítulo IV 17
4 Requerimientos y Diseño 17
4.1 Definición de requerimientos 17
4.2 Diseño de la plataforma de software 41
Capítulo V 49
5 Plataforma de software 49
5.1 Aplicación 49
5.2 Evaluación 52
Capítulo VI 54
6 Conclusiones 54
6.1 Recomendaciones 55
6. Bibliografía 57

Anexos 59

4. Índice de figuras

Figura 1: Gráfico de comparación entre tecnologías más buscadas en Google en el último año. 4
Figura 2: Cronograma de trabajo completo. 16
Figura 3.1: Diagrama Casos de Uso Metricapp Java. 18
Figura 3.2: Diagrama Casos de Uso Metricapp Web. 21
Figura 4: Modelo Conceptual. 38
Figura 5.1: Interfaz Inicio de sesión. 41
Figura 5.2: Interfaz Página inicial. 41
Figura 5.3: Interfaz Generar reporte diario. 42
Figura 5.4: Interfaz Generar reporte de finalización. 42
Figura 5.5: Interfaz Ver métricas. 43
Figura 5.6: Interfaz Métrica ejecutada. 43
Figura 5.7: Interfaz Ver usuarios. 44
Figura 5.8: Interfaz Modificar usuarios. 44
Figura 5.9: Interfaz Ver gestores. 45
Figura 5.10: Interfaz Modificar gestor. 45
Figura 5.11: Interfaz Ejecutar servicio Test Rail – Base de datos. 46
Figura 5.12: Interfaz Ver Ayuda. 46
Figura 6: Diagrama Entidad – Relación de la base de datos. 48
Figura 7: Arquitectura de la solución. 51

5. Índice de tablas

Tabla 1: Involucrados. 9
Tabla 2: Matriz para el procedimiento metodológico. 10
Tabla 3: Documentación de riesgos. 13
Tabla 3.1: Riesgo 1. 13
Tabla 3.2: Riesgo 2. 14
Tabla 3.3: Riesgo 3. 14
Tabla 3.4: Riesgo 4. 15
Tabla 3.5: Riesgo 5. 15
Tabla 4: Requerimientos Funcionales. 17
Tabla 5.1: Casos de Uso, Generar Reporte Diario Java. 19
El usuario debe haber iniciado sesión correctamente 19
Tabla 5.2: Casos de Uso, Generar Reporte de Finalización Java. 20
Tabla 5.3: Casos de Uso, Iniciar Sesión. 22
Tabla 5.4: Casos de Uso, Recuperar Contraseña. 23
Tabla 5.5: Casos de Uso, Ver Ayuda. 24
Tabla 5.6: Casos de Uso, Cerrar Sesión. 25
Tabla 5.7: Casos de Uso, Cambiar Contraseña. 26
Tabla 5.8: Generar Reporte Diario Web 27
Tabla 5.9: Casos de Uso, Generar Reporte de Finalización Web. 28
Tabla 5.10: Casos de Uso, Ejecutar Métricas. 29
Tabla 5.11: Casos de Uso, Ver Gestores. 30
Tabla 5.12: Casos de Uso, Agregar Gestores. 31
Tabla 5.13: Casos de Uso, Modificar Gestores. 32
Tabla 5.14: Casos de Uso, Eliminar Gestor. 33
Tabla 5.15: Casos de Uso, Ver Usuarios. 34
Tabla 5.16: Casos de Uso, Agregar Usuarios. 35
Tabla 5.17: Casos de Uso, Modificar Usuarios. 36
Tabla 5.18: Casos de Uso, Servicio Test Rail – Base de datos. 37
Tabla 6: Requerimientos no funcionales. 40

Capítulo I

1 Introducción

1.1 Descripción de empresa

SOIN (Soluciones Integrales) es una empresa de 36 años de existencia con capital cien por ciento costarricense. Desarrolladora de soluciones informáticas para grandes corporaciones y proyectos de alta complejidad con presencia en algunos países de América Latina, Estados Unidos, Europa y China. Mayormente cuenta con clientes ligados directamente al gobierno, pero, además, gestiona otros tipos de proyectos de alta gama relacionados a la tecnología y todo lo que en ella se incluye.

En SOIN se comprometen con la mejora continua de sus soluciones para promover la satisfacción del cliente, el cumplimiento legal aplicable al sector y la búsqueda de la excelencia que los caracteriza como empresa. Se caracteriza por ser la primera empresa en América Latina en adoptar e invertir seriamente en las tecnologías emergentes, generando productos informáticos de calidad mundial que se han adelantado a los tiempos, siendo la primera empresa costarricense en exportar software desde 1989.

1.2 Contexto del proyecto

El proyecto presenta una propuesta para actualizar Metricapp, una aplicación desarrollada en 2018 y que desde entonces sigue siendo utilizada. Esta aplicación fue desarrollada en Java hace 3 años y desde sus inicios ha ido recibiendo actualizaciones. Fue creada por uno de los miembros del departamento de Sostenibilidad y empezó como un pequeño aplicativo con interfaz gráfica. Consiste en un generador de reportes o informes de finalización de proyecto de pruebas y algunas métricas exclusivas del equipo de QA (quality assurance). También se ejecuta diariamente, ya que al final de cada día hay que realizar el envío de un informe diario y un informe de finalización de proyecto de pruebas.

Además, Metricapp se encuentra conectada con otros sistemas del departamento de QA, estas conexiones consisten en la extracción de información necesario para generar reportes o ejecutar métricas en Metricapp. Debido a estas conexiones con otros sistemas, es necesario que todas se encuentren en un mismo punto para poder funcionar de manera correcta. Esto significa que puede haber ciertos cambios significativos en cualquiera de los sistemas que alcance a afectar a los demás, en este caso, ya sea que Metricapp cambie o que necesite cambiar gracias a un cambio en otro sistema.

1.3 Problema

El sistema Metricapp ha dejado de recibir datos necesarios por parte del gestor de defectos, otro sistema utilizado en el departamento de QA. Estos datos son necesarios para el correcto funcionamiento de algunos módulos de la aplicación Metricapp, como generar reportes diarios o de finalización de proyecto de pruebas. Con este cambio, el departamento de QA busca poder aprovechar nuevas y mejores funciones que brinda la API de JIRA en todos sus sistemas.

Como se menciona anteriormente, Metricapp fue desarrollada en Java como una aplicación de escritorio. Desde este entorno se conecta directamente a la base de datos y realiza peticiones http a TestRail y Bugzilla. Realizar una actualización sobre esta aplicación implica tener que enviar a todos los usuarios un instalador nuevo para que cada uno realice la respectiva actualización. Además de no ser recomendable tener funciones de servidor y vistas en el mismo entorno.

1.4 Objetivos

1.4.1 *Objetivo general*

Desarrollar una plataforma web con la integración de JIRA con el sistema Metricapp para la gestión de los informes de seguimiento de proyectos de prueba del departamento de QA de SOIN.

1.4.2 *Objetivos específicos*

1 Definir los requerimientos requeridos para el proceso de integración y migración.

2 Desarrollar la integración de JIRA en Metricapp y la migración a un sistema web.

3 Diseñar una arquitectura base que permite visualizar la estructura general del aplicativo, scripts y demás elementos.

4 Evaluar la calidad del producto por medio de la aplicación de pruebas en el desarrollo y correcciones solicitadas por el departamento de Aseguramiento de la Calidad.

1.5 Justificación

La migración de la API de Bugzilla a la API de JIRA en Metricapp, le garantizará al sistema una correcta comunicación con los demás aplicativos del departamento de QA de la empresa. Además de corregir la comunicación, Metricapp podría aprovechar en un futuro las nuevas funciones proporcionadas por JIRA que no estaban en Bugzilla. Por lo que, si algún otro sistema que ya está conectado con JIRA decide implementar una función de esta API, Metricapp estará listo para utilizarla también en caso de ser necesario, o viceversa.

Luego de corregir la primera problemática, el equipo ve necesario eliminar las deficiencias de las aplicaciones de escritorio descritas previamente. Por medio de un sistema web nuevo, Metricapp puede quedar mejor estructurado separando las funciones de servidores con las vistas con las que interactúan los usuarios, mejorando el rendimiento del sistema. Además, ya no existiría la necesidad de enviar a todos los usuarios un nuevo instalador para realizar la actualización. Con el link de la página web, los usuarios nada más deberán de actualizar la página para poder ver los cambios, ahorrando tiempo al generador de la actualización y a los usuarios.

Capítulo II

2 Revisión de literatura

2.1 Marco teórico

2.1.1 React

Es una librería de JavaScript inicialmente desarrollada por Facebook (*Qué es React. Por qué usar React*, 2019). Esta tecnología surge gracias a las necesidades producidas durante el desarrollo web de esta red social, como lo es la velocidad en la que se le presenta el contenido al usuario, una de las principales características de React. Esta librería es muy popular en la actualidad debido a estar muy enfocada al desarrollo de interfaces de usuario. Según Google Trends, en términos más buscados entre el 6 de septiembre del 2020 y el 16 de mayo del 2021, React lidera el interés. Aunque estas estadísticas no demuestren que la tecnología ha sido utilizada, si nos indica el interés que existe respecto a otras. El siguiente gráfico muestra la comparación, en el tiempo mencionado anteriormente, entre React JS (media de 18), Angular JS (media de13) y Vue JS (media de 8). Los números indican el interés a lo largo del tiempo, donde 100 sería la popularidad máxima de un término.

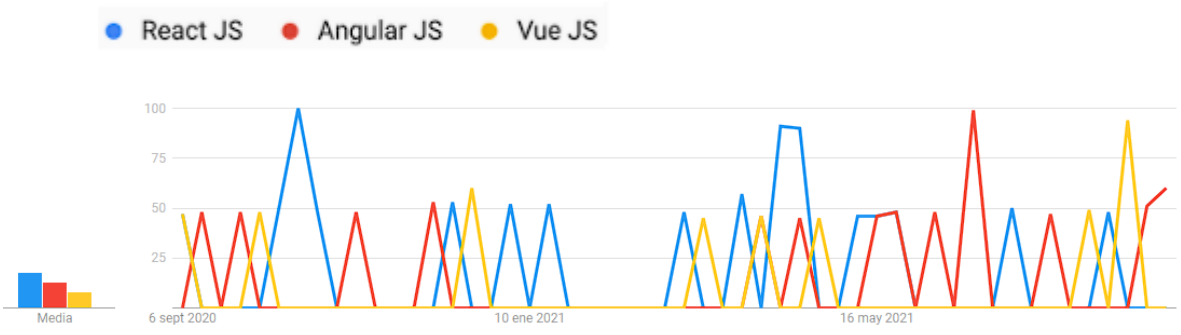

Figura 1: Gráfico de comparación entre tecnologías más buscadas en Google en el último año.
Fuente: Google Trends.

Por su crecimiento, hoy en día se pueden desarrollar todo tipo de aplicaciones web y móviles. Algunos ejemplos de aplicaciones que han utilizado

React JS son: Airbnb, Discord, WhatsApp, Netflix, Instagram, entre otros (Colectiva, 2020). Para este proyecto, se utilizará esta tecnología para desarrolla una aplicación web tipo SPA (Single Page Application), en busca de mejor rendimiento al navegar entre vistas, en lugar de páginas.

2.1.2 Express/Node

Para el desarrollo de la RESTful API en este proyecto se utilizarán NodeJs y ExpressJs. Node es un entorno que permite crear aplicaciones network escalables y aplicaciones en JavaScript (*Introducción a Express/Node – Aprende Sobre Desarrollo Web* | MDN, 2021). Express es el framework web más popular de Node (con 17 243 771 descargas semanales según NPM (2019)), contiene miles de métodos de utilidad HTTP y middlewares a su disposición, gracias a esto, la creación de una API es rápida, eficiente y sencilla (*Express – Infraestructura de aplicaciones web Node.js,* s. f.).

2.1.3 MySQL

Según un estudio de Robledano (2020), MySQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad. Su popularidad recae en ser gratuita, fácil de usar (necesita pocos comandos para empezar, por ejemplo), es veloz, utiliza varias capas de seguridad, asegura la eficiencia y es compatible tanto con Windows como con Linux. Algunos sitios que utilizan MySQL son YouTube, Wikipedia, Facebook, Google, Flickr, Twitter, entre otros (*MySQL: ¿Qué es? Características, Ventajas y Desventajas*, 2019).

2.1.4 Jira

En el estudio de Vega (2021) se describe a Jira como una herramienta que sirve para la gestión del trabajo, incluyendo prácticamente todas las etapas de desarrollo de software. Esta es una herramienta en línea que permite organizar tareas o actividades y además ayuda a equipos de todo tipo a gestionar incidencias y errores. En el contexto del proyecto, se consumirán datos provenientes de la API de Jira relacionada a los bugs de los proyectos.

2.1.5 Test Rail

Al igual que Jira, Metricapp consume datos provenientes de la API de Test Rail. Esta es una herramienta de gestión de casos de prueba basada en la web (*Introduction to TestRail*, 2021). Es una solución para gestionar, organizar y rastrear el proceso de prueba de uno o varios softwares en una empresa, beneficiaria para los equipos de desarrollo y aseguramiento de la calidad (QA)

2.2 Trabajos relacionados

Hace ya varios años que para el desarrollo de aplicaciones web han surgido frameworks o librerías. Estos se han vuelto sumamente populares y han desplazado a prácticamente todas las tecnologías utilizadas en el pasado para este ambiente. Dependiendo del aplicativo web a desarrollar, se pueden involucrar muchas tecnologías y en esta sección se detallan las utilizadas por este proyecto, con base a trabajos relacionados que justifican su uso.

Los frameworks o librerías para la creación de interfaces web han surgido como una necesidad de ayudar al desarrollar y de ofrecer más en rendimiento y funcionalidades a los usuarios consumidores. La popularidad de React se debe, según Aggarwal (2018), a la manera en que interactúa con el DOM (Document Object Model), ya que no es una interacción con el DOM generado por el navegador, como lo realizan otros frameworks. React interactúa con el DOM almacenado en la memoria y esto aumenta la rapidez y el rendimiento la aplicación. La diferencia entre el Virtual DOM (así conocido en React) con el DOM del navegador recae a la hora de modificar gran cantidad de datos y en la manipulación del DOM que dispara eventos en cada página. Esta forma de trabajo es la que deja atrás al DOM en comparación al Virtual DOM de React.

El trabajo de Liang (2017) justamente detalla que NodeJs contiene múltiples módulos de red como HTTP, DNS, NET, UDP, HTTPS, entre otros. Esta característica de Node incita a desarrollar este servidor web, que, si a futuro se desea incluir alguna otra API externa o interna que utilice alguno de estos

protocolos, esta pueda ser implementa sin mayor dificultad. Este trabajo comenta también la ventaja que emplea Node con los eventos conducidos y la programación asíncrona, la cual es utilizar todos los recursos del sistema.

Capítulo III

3 Solución planteada

3.1 Propuesta

Se propone realizar una integración entre la aplicación desarrollada en Java y JIRA. Luego de realizar la integración de la nueva API en Metricapp, se procederá a realizar una migración a un sistema web con el fin de mejorar la aplicación actual. Para este sistema web se desarrollará tanto la parte de Frontend como de Backend. Los componentes por utilizar serían:

- Frontend: realiza la comunicación entre los usuarios y el Backend. Será desarrollado utilizando ReactJS.
- Backend: gestiona los datos extraídos de JIRA y generados por el sistema. Será desarrollado en NodeJs y para la base de datos se reutilizará la que está implementada actualmente, en MySQL.
- API: pasaría de ser la de Bugzilla a utilizar la JIRA.

3.2 Involucrados

Tabla 1: Involucrados.

Nombre	Departamento	Puesto	Labores	Responsabilidades
Gustavo Fernández Jiménez	Aseguramiento de la Calidad	Senior Manager	Encargado de todos los departamentos de QA de la empresa	Será uno de los miembros que evaluará el éxito del proyecto, además evaluará el desempeño del practicante y finalmente realizará ciertas funciones administrativas necesarias
Maykol Picado Aguilar	Aseguramiento de la Calidad	Senior Automation Tester	Encargado del área de automatización	Administrador del proyecto. Brindará indicaciones y correcciones para la ejecución del proyecto, determinará si se cumplieron los requerimientos, dará sello de calidad e indicará el éxito del proyecto
Carlos Andrés Villalobos Salas	-	Software Developer	Encargado de realizar el cambio de API en el sistema Metricap y de desarrollar tanto Frontend como Backend para el sistema web.	Cumplir con los requerimientos establecidos y demás tareas necesarias para satisfacer el objetivo del proyecto.
Josué Brenes Rojas	Aseguramiento de la Calidad	Entry Level QA Tester	Realizar diseños automatizados para las pruebas. Desarrollar o programar pruebas automatizadas, hacer ejecuciones semanales de regresivas automatizadas	Brindará una función de soporte ante cualquier duda técnica respecto al proyecto.

3.3 Procedimiento metodológico

Tabla 2: Matriz para el procedimiento metodológico.

Objetivo Específico	Tarea	Meta	Indicador
Definir los requerimientos para el proceso de integración y migración.	**A**. Determinar los requerimientos funcionales del sistema actual con la nueva API de JIRA. **B.** Desarrollar los requerimientos funcionales del sistema web. **C.** Establecer los requerimientos no funcionales de la aplicación web.	**A.** Reporte con la información obtenida de una revisión del sistema actual en funcionamiento. **B.** Establecer las funcionalidades que desarrollaran en la integración y migración.	**A.** Lista con todos los requerimientos funcionales y no funcionales del proyecto avalada por los encargados del proyecto.

Desarrollar la integración de JIRA en Metricapp y la migración a un sistema web.	**A.** Reemplazar en el sistema Metricapp la API de Bugzilla por la API de JIRA. **B.** Realizar una aplicación web en ReactJS. **C.** Desarrollar el Backend de la aplicación en NodeJs.	**A.** Implementar correcto de la nueva API. **B.** Migrar el primer módulo del sistema a una aplicación web. **C.** Realizar la conexión entre la base de datos y la API con la aplicación web.	**A.** Aprobación del funcionamiento por parte de los involucrados. **B.** El módulo contiene las funcionalidades necesarias y es avalado por los encargados del proyecto. **C.** Aprobación de los involucrados de las conexiones realizadas para el funcionamiento del sistema web.
Diseñar una arquitectura base que permite visualizar la estructura general del aplicativo, scripts y demás elementos	**A.** Descubrir los cambios del sistema actual con la implementación de la nueva API. **B.** Diseñar los módulos del sistema web a desarrollar.	**A.** Reporte con los cambios sufridos por el sistema después de la implementación. **B.** Realizar dos diagramas de clases de programación de Metricapp con	**A.** Lista aprobada por los involucrados con los cambios generados en Metricapp para realizar la integración de la nueva API. **B.** Diagrama de clases que abarca el sistema web

		JIRA y la aplicación web. **C.** Elaborar un prototipo de la aplicación web (Interfaces de usuario).	completo, avalado por los encargados. **C.** Vistas de la aplicación web aprobadas por la empresa.
Garantizar la calidad del producto por medio de la aplicación de pruebas en el desarrollo y correcciones solicitadas por el departamento de Aseguramiento de la Calidad.	**A.** Aplicar pruebas de caja blanca en el proceso de desarrollo. **B.** Aplicar las correcciones indicadas por el departamento de Aseguramiento de la Calidad.	**A.** Al menos una prueba exitosa por cada función del sistema **B.** Eliminar las fallas encontradas por el departamento de Aseguramiento de la Calidad.	**A.** Aprobación de las pruebas realizadas por parte del departamento. **B.** Verificación del departamento de que las correcciones se han realizado correctamente.

3.4 Análisis de los riesgos

La Tabla 2 hace énfasis al formato establecido para documentar los riesgos. En todos los desarrollos de proyectos existen posibles riesgos que hay que tomar en cuenta. Es importante realizar esta identificación antes de iniciar el desarrollo para poder prevenir la disconformidad del resultado.

Tabla 3: Documentación de riesgos.

Código	
Nombre	
Categoría	
Causa	
Impacto	
Estrategia de evasión	
Estrategia mitigación	
Estrategia contingencia	

Tabla 3.1: Riesgo 1.

Código	R-1
Nombre	No cumplir con los requisitos funcionales
Categoría	Humano
Causa	El desarrollador no logra ser capaz de finalizar lo establecido.
Impacto	Muy Alto
Estrategia de evasión	No hay
Estrategia mitigación	Constante monitoreo por parte de los encargados del proyecto. Inclusión de un desarrollador adicional.

Estrategia contingencia	Continuar con el uso de Metricapp en Java en caso de no concluir este proyecto, para poder seguir creando los reportes.

Tabla 3.2: Riesgo 2.

Código	R-2
Nombre	Acceso a internet.
Categoría	Tecnológico
Causa	No poder conectarse a una red o la caída de la misma.
Impacto	Alto
Estrategia de evasión	No hay forma de evadir este riesgo.
Estrategia mitigación	No hay forma de mitigar este riesgo.
Estrategia contingencia	Sin estrategia de contingencia.

Tabla 3.3: Riesgo 3.

Código	R-3
Nombre	Estimación de tiempos incorrecta.
Categoría	Humano
Causa	Planificación mal realizada.
Impacto	Alto
Estrategia de evasión	Contar con expertos sobre el tema a la hora de realizar y revisar la propuesta.
Estrategia mitigación	Uso de metodologías ágiles para la administración del proyecto.
Estrategia contingencia	Priorizar tareas y en caso de ser necesario, involucrar un desarrollador más.

Tabla 3.4: Riesgo 4.

Código	R-4
Nombre	Conexión Base de Datos
Categoría	Tecnológico
Causa	Caída de la conexión.
Impacto	Alto
Estrategia de evasión	Desarrollo perfecto de la base de datos.
Estrategia mitigación	Realizar todas las pruebas necesarias.
Estrategia contingencia	Intentos de conexión para ver si se puede lograr la reconexión.

Tabla 3.5: Riesgo 5.

Código	R-5
Nombre	Conexión Test Rail/Jira
Categoría	Tecnológico
Causa	Caída de las conexiones.
Impacto	Alto
Estrategia de evasión	No hay forma de evadir este riesgo.
Estrategia mitigación	No hay forma de mitigar este riesgo.
Estrategia contingencia	Sin estrategia de contingencia.

3.5 Cronograma de trabajo

Task Name	Estados	% completado	Duración	Comienzo	Fin	Predecesoras	Trabajo
Proyecto: Integración y migración del sistema Metricap		**0%**	**552 hrs**	**6/7/2021 08:00**	**15/11/2021 15:00**		**0 hrs**
Induccion	Sin iniciar	0%	18 hrs	6/7/2021 08:00	8/7/2021 15:00		0 hrs
Fase 1		**0%**	**194 hrs**	**9/7/2021 08:00**	**25/8/2021 10:00**		**0 hrs**
Levantamiento de requerimientos		**0%**	**60 hrs**	**9/7/2021 08:00**	**22/7/2021 15:00**		**0 hrs**
Casos de uso extendidos	Sin iniciar	0%	30 hrs	9/7/2021 08:00	15/7/2021 15:00	187	0 hrs
Modelo conceptual	Sin iniciar	0%	24 hrs	16/7/2021 08:00	21/7/2021 15:00	190	0 hrs
Glosario de términos	Sin iniciar	0%	6 hrs	22/7/2021 08:00	22/7/2021 15:00	191	0 hrs
Diseño		**0%**	**60 hrs**	**23/7/2021 08:00**	**6/8/2021 15:00**		**0 hrs**
Diagrama de clases/Diagrama DB	Sin iniciar	0%	30 hrs	23/7/2021 08:00	29/7/2021 15:00	189	0 hrs
Prototipo interfaz de usuario Web (no funcional)	Sin iniciar	0%	30 hrs	30/7/2021 08:00	6/8/2021 15:00	194	0 hrs
Desarrollo y pruebas		**0%**	**72 hrs**	**9/8/2021 08:00**	**24/8/2021 15:00**		**0 hrs**
Integración Metricap actual + Jira en un nuevo repositorio	Sin iniciar	0%	60 hrs	9/8/2021 08:00	20/8/2021 15:00	193	0 hrs
Pruebas	Sin iniciar	0%	12 hrs	23/8/2021 08:00	24/8/2021 15:00	197	0 hrs
Generar y entregar instalador	Sin iniciar	0%	2 hrs	25/8/2021 08:00	25/8/2021 10:00	198	0 hrs
Fase 2		**0%**	**310 hrs**	**25/8/2021 10:00**	**8/11/2021 15:00**		**0 hrs**
Desarrollar base del sistema Web (Modelo navegable no funcional)	Sin iniciar	0%	30 hrs	25/8/2021 10:00	1/9/2021 10:00	199	0 hrs
Desarrollo de un modulo Web (reporte diario y de cierre) + Unitarias	Sin iniciar	0%	240 hrs	1/9/2021 10:00	29/10/2021 10:00	201	0 hrs
Pruebas funcionales (QA)	Sin iniciar	0%	40 hrs	29/10/2021 10:00	8/11/2021 15:00	202	0 hrs
Documentación de cierre de proyecto	Sin iniciar	0%	30 hrs	9/11/2021 08:00	15/11/2021 15:00	203	0 hrs

Figura 2: Cronograma de trabajo completo. Fuente: elaboración propia

Capítulo IV

4 Requerimientos y Diseño

4.1 Definición de requerimientos

4.1.1 Requerimientos Funcionales

Tabla 4: Requerimientos Funcionales.

Código	Nombre	Alcance Propuesto
MJ-0	Generar Reportes Diarios en Java	Funcional
MJ-1	Generar Reportes de Finalización en Java	Funcional
MW-0	Iniciar Sesión	Funcional
MW-1	Recuperar Contraseña	Funcional
MW-2	Ver Ayuda	Funcional
MW-3	Cerrar Sesión	Funcional
MW-4	Cambiar Contraseña	Funcional
MW-5	Generar Reportes Diarios en Web	Funcional
MW-6	Generar Reportes de Finalización en Web	Funcional
MW-7	Ejecutar Métricas	Visual
MW-8	Ver Gestores	Funcional
MW-9	Agregar Gestores	Funcional
MW-10	Modificar Gestores	Funcional
MW-11	Eliminar Gestores	Funcional
MW-12	Ver Usuarios	Funcional
MW-13	Agregar Usuarios	Funcional
MW-14	Modificar Usuarios	Funcional
MW-15	Cargar datos de TestRail a la base de datos	Visual

4.1.2 Diagrama de casos de uso de la fase 1 (Integración de JIRA en Metricapp Java):

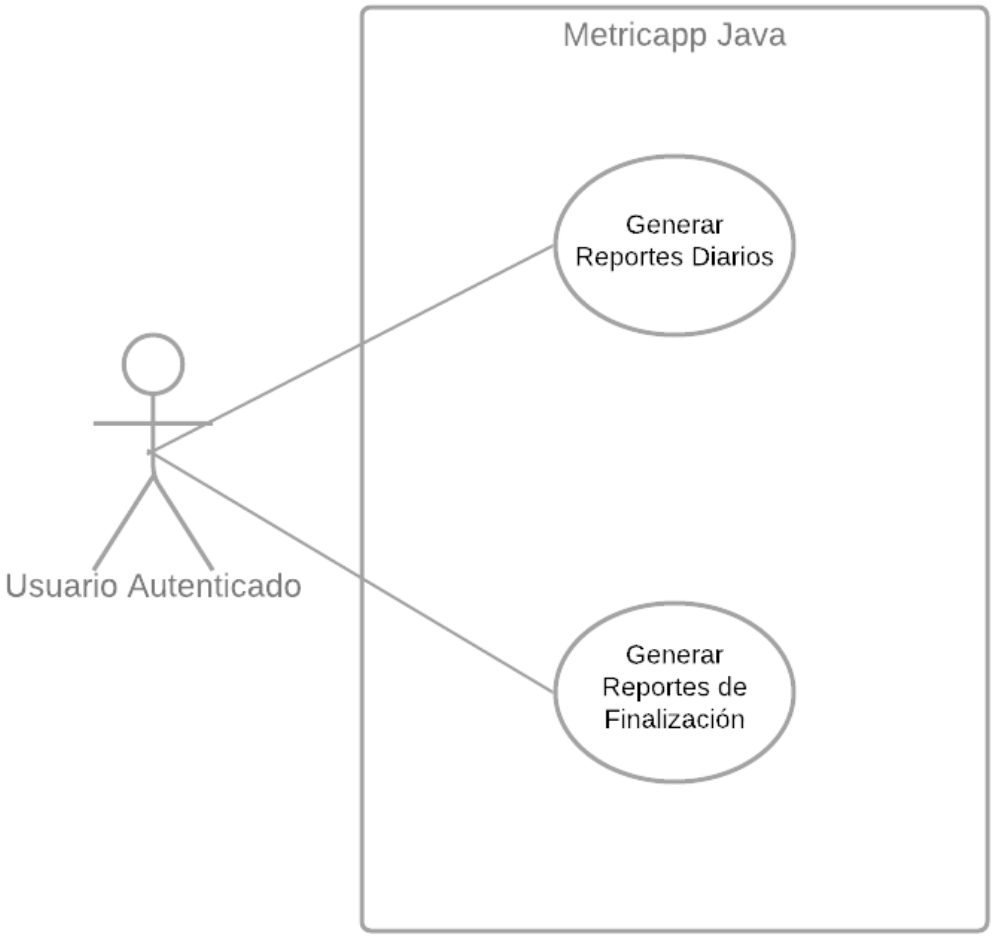

Figura 3.1: Diagrama Casos de Uso Metricapp Java. Fuente: Elaboración propia.

4.1.3 Especificaciones de Casos de Uso

Tabla 5.1: Casos de Uso, Generar Reporte Diario Java.

Nombre:	Generar Reporte Diario Java
Descripción:	Los usuarios generan reportes diarios sobre requerimientos de distintos proyectos. Con el fin de darle seguimiento a las pruebas y defectos de estos.
Actores:	Usuario Tester y Usuario Administrador
Precondición:	**El usuario debe haber iniciado sesión correctamente**
Postcondición:	El usuario habrá generado un reporte diario.
Flujo básico:	1. Se ejecuta el sistema. 2. Se inicia sesión. 3. Clic sobre la opción “Reportes”. 4. Se despliegan las opciones que incluye “Reportes”. 5. Se selecciona la opción “Generar Reporte Diario”. 6. Se actualiza la ventana principal con el respectivo menú. 7. Se selecciona el proyecto. 8. Se selecciona el requerimiento del proyecto. 9. Se coloca un comentario. 10. Se presiona crear reporte. 11. Se guarda en el equipo.
Flujos alternativos:	1. Se marca la opción de ver Test Plans 2. Error si el usuario no digita un comentario. 3. Error si el usuario no selecciona un proyecto. 4. Error si el usuario no selecciona un requerimiento.
Requisitos de interfaz de usuario:	* La aplicación en Java ya existe, se realizará la integración con JIRA *

Tabla 5.2: Casos de Uso, Generar Reporte de Finalización Java.

Nombre:	Generar Reporte de Finalización de proyectos Java
Descripción:	Los usuarios generan reportes finales una vez que las pruebas sobre un proyecto hayan concluido.
Actores:	Usuario Tester y Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente
Postcondición:	El usuario habrá generado un reporte de finalización de proyecto de pruebas.
Flujo básico:	1. Se ejecuta el sistema. 2. Se inicia sesión. 3. Click sobre la opción “Reportes”. 4. Se despliegan las opciones que incluye “Reportes”. 5. Se selecciona la opción “Generar Reporte de Finalización de proyecto”. 6. Se actualiza la ventana principal con el respectivo menú. 7. Se selecciona el proyecto. 8. Se selecciona el requerimiento del proyecto. 9. Se coloca un comentario. 10. Se presiona crear reporte. 11. Se guarda en el equipo.
Flujos alternativos:	1. Se selecciona Ver Test Plans 2. Se selecciona Ver Finalizados 3. Error si el usuario no digita un comentario. 4. Error si el usuario no selecciona un proyecto. 5. Error si el usuario no selecciona un requerimiento.
Requisitos de interfaz de usuario:	* La aplicación en Java ya existe, se realizará la integración con JIRA *

4.1.4 Diagrama de casos de uso de la fase 2 (Migración de Metricapp a Web):

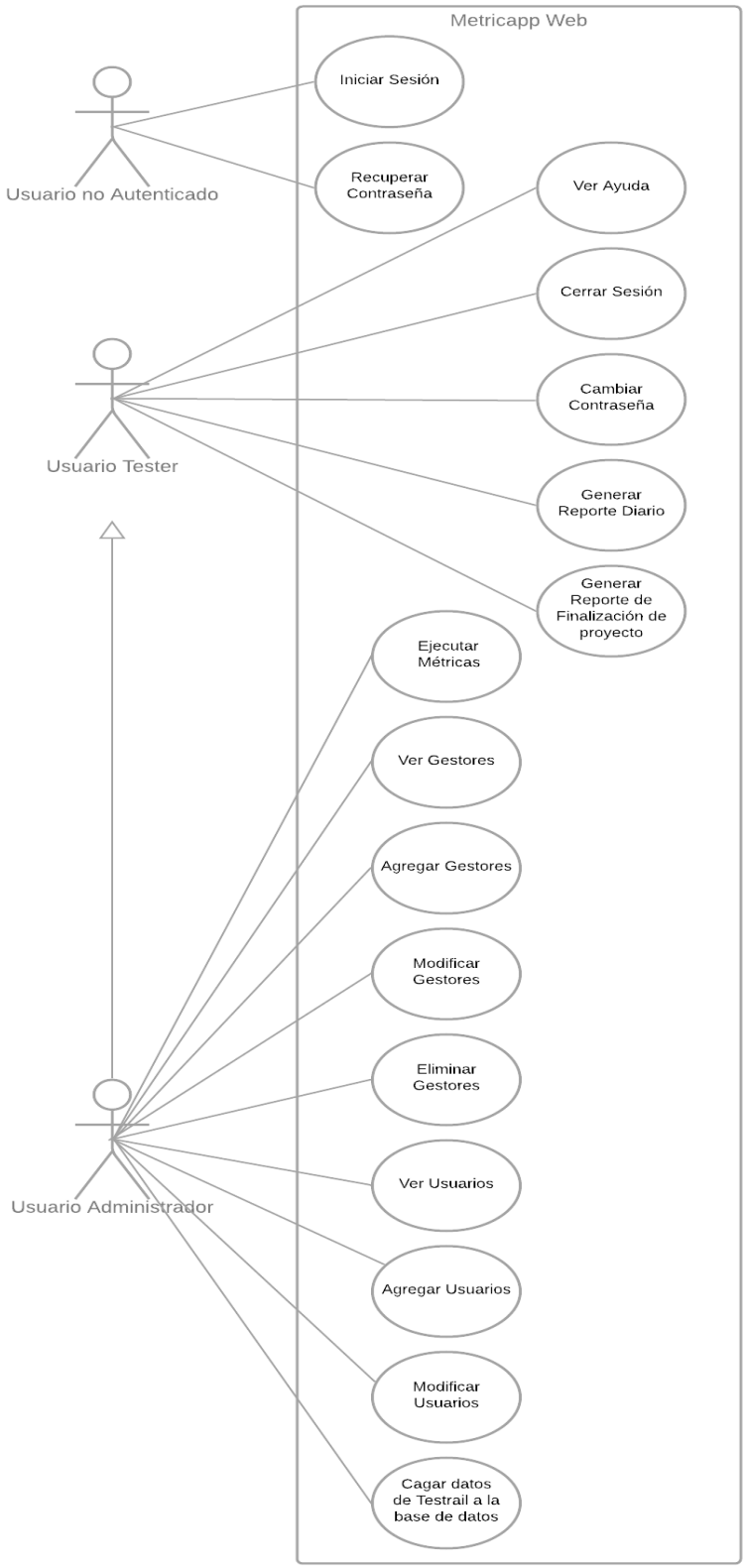

Figura 3.2: Diagrama Casos de Uso Metricapp Web. Fuente: Elaboración propia.

4.1.5 Especificaciones de Casos de Uso.

Tabla 5.3: Casos de Uso, Iniciar Sesión.

Nombre:	Iniciar Sesión
Descripción:	Para poder utilizar el sistema, es necesario iniciar sesión, tanto para el usuario tester como para el administrador.
Actores:	Usuario no autenticado
Precondición:	El usuario debe haber sido creado previamente.
Postcondición:	Acceso las respectivas funciones según el rol asignado al usuario.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se digita el usuario y la contraseña. 3. Se presiona sobre el botón de iniciar sesión. 4. Se realiza la confirmación del acceso.
Flujos alternativos:	1. El usuario ya ha iniciado sesión y abandonó el sistema sin cerrar sesión. Entonces no necesita realizar el inicio de sesión otra vez, en un tiempo estimado de 8 horas. 2. Las credenciales que ha digitado son erróneas. Debe volver a digitarlas o solicitar acceso a un administrador.
Requisitos de interfaz de usuario:	1. Campo para digitar el correo 2. Campo para digitar la contraseña 3. Botón para iniciar sesión. 4. Botón para recuperar la contraseña olvidada.

Tabla 5.4: Casos de Uso, Recuperar Contraseña.

Nombre:	Recuperar Contraseña
Descripción:	Se le permite al usuario generar una nueva contraseña en caso de haberla olvidado.
Actores:	Usuario no autenticado
Precondición:	El usuario debe haber sido creado previamente.
Postcondición:	El usuario tendrá una nueva contraseña.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se presiona sobre el botón ´Recuperar Contraseña´. 3. Se abre un modal para digitar el correo. 4. Se digita y se confirma. 5. Al correo llegará una nueva contraseña para el usuario. 6. Se realizan los pasos que indica el correo para recuperar la contraseña.
Flujos alternativos:	1. El usuario no existe. Se le indica y no se le envía una contraseña. 2. Cierra el modal y vuelve a la página de inicio. 3. Recibe el correo, pero no termina los pasos para la recuperación.
Requisitos de interfaz de usuario:	1. Botón para acceder al modal. 2. Modal. 3. Campo para digitar el correo. 4. Botón para generar nueva contraseña. 5. Botón para cerrar modal. 6. Ventana para terminar la recuperación de la contraseña con los siguientes campos de texto: • Correo, campo no editable. • Contraseña nueva. • Confirmar nueva contraseña.

Tabla 5.5: Casos de Uso, Ver Ayuda.

Nombre:	Ver Ayuda
Descripción:	Se muestra un cuadro con información que podría ser útil para aclarar dudas de algún usuario.
Actores:	Usuario Tester y Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente
Postcondición:	El usuario podría haber aclarado sus dudas.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se presiona sobre la opción “Ayuda” del menú de opciones. 4. Se muestra en la ventana principal la ayuda que brinda el sistema.
Flujos alternativos:	1. El usuario ingresa al sistema, pero no presiona esta opción.
Requisitos de interfaz de usuario:	1. Botón en el menú de opciones 2. Cuadro de información.

Tabla 5.6: Casos de Uso, Cerrar Sesión.

Nombre:	Cerrar Sesión
Descripción:	La opción segura a la hora de dejar de trabajar en un ordenador que no es el propio. O necesaria si el usuario desea iniciar sesión con otra cuenta.
Actores:	Usuario Tester y Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente
Postcondición:	El usuario se encontrará en la página inicial del sistema.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se presiona sobre la opción “Cerrar Sesión” del menú de opciones. 4. Se navega a la página inicial del sistema.
Flujos alternativos:	
Requisitos de interfaz de usuario:	1. Botón en el menú de opciones

Tabla 5.7: Casos de Uso, Cambiar Contraseña.

Nombre:	Cambiar Contraseña
Descripción:	El usuario tiene la opción de cambiar su contraseña si así lo desea.
Actores:	Usuario Tester y Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente
Postcondición:	El usuario obtendrá una nueva contraseña
Flujo básico:	1. Se inicia sesión. 2. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 3. Se despliegan las opciones que incluye “Administración”. 4. Se selecciona la opción “Cambiar Contraseña”. 5. Se abre un modal. 6. Se digita la nueva contraseña. 5. Se presiona confirmar cambios.
Flujos alternativos:	1. El usuario ingresa al sistema, pero es administrador. 2. Se coloca el cursor sobre “Administración” en el menú de opciones. 3. Se despliegan las opciones de “Administración” 4. Se selecciona “Gestionar Usuarios” 5. Se busca y selecciona el usuario. 6. Se presiona en modificar. 7. Se abre un modal. 8. Se digita la nueva contraseña. 9. Se presiona confirmar cambios.
Requisitos de interfaz de usuario:	1. Modal. 2. Campos para la digitar y con. 3. Botón para confirmar cambios. 2. Botón para devolverse.

Tabla 5.8: Generar Reporte Diario Web

Nombre:	Generar Reporte Diario Web
Descripción:	Los usuarios generan reportes diarios sobre requerimientos de distintos proyectos, con el fin de darle seguimiento a las pruebas y defectos de estos.
Actores:	Usuario Tester y Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente
Postcondición:	El usuario habrá generado un reporte diario.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se coloca el cursor sobre la opción “Reportes” del menú de opciones. 4. Se selecciona la opción “Generar Reporte Diario”. 5. Se selecciona el proyecto. 6. Se selecciona el requerimiento del proyecto. 7. Se coloca un comentario. 8. Se presiona crear reporte. 6. Se guarda en el equipo.
Flujos alternativos:	1. Se marca la opción de ver Test Plans 2. Error si el usuario no digita un comentario. 3. Error si el usuario no selecciona un proyecto. Error si el usuario no selecciona un requerimiento.
Requisitos de interfaz de usuario:	1. Picker para seleccionar un proyecto. 2. Tabla para seleccionar un requerimiento. 3. Checkbox para ver Test Plans. 4. Botones para pasar de página. 5. Campo para escribir el tester que genera el reporte. 6. Campo para escribir comentarios. 7. Botón para crear reporte. 3. Interfaz para ver y guardar reporte en el equipo.

Tabla 5.9: Casos de Uso, Generar Reporte de Finalización Web.

Nombre:	Generar Reporte de Finalización de proyectos Web
Descripción:	Los usuarios generan reportes finales una vez que las pruebas sobre un proyecto hayan concluido.
Actores:	Usuario Tester y Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente
Postcondición:	El usuario habrá generado un reporte de finalización de proyecto
Flujo básico:	1. Se inicia sesión. 2. Se colocar el cursor sobre la opción “Reportes” del menú de opciones. 3. Se selecciona la opción “Generar Reporte de Finalización”. 4. Se selecciona el proyecto. 5. Se selecciona el requerimiento del proyecto. 6. Se coloca un comentario. 7. Se presiona crear reporte. 8. Se guarda en el equipo.
Flujos alternativos:	1. Se selecciona Ver Test Plans o Ver Finalizados 2. Error si el usuario no digita un comentario. 3. Error si el usuario no selecciona un proyecto o un requerimiento. 4. Error si el usuario no selecciona un requerimiento.
Requisitos de interfaz de usuario:	1. Picker para seleccionar un proyecto. 2. Tabla para seleccionar un requerimiento. 3. Checkbox para ver Test Plans. 4. Checkbox para ver Finalizados 5. Campo para escribir el tester que genera el reporte. 6. Campo para escribir comentarios. 7. Botón para crear reporte. 8. Interfaz para ver y guardar reporte en el equipo.

Tabla 5.10: Casos de Uso, Ejecutar Métricas.

Nombre:	Ejecutar Métricas
Descripción:	Los usuarios administradores pueden graficar algunas métricas.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá graficado una métrica.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se coloca el cursor sobre la opción “Métricas” del menú de opciones. 4. Se despliegan las opciones que incluye “Métricas”. 5. Se selecciona la opción “Ejecutar Métricas”. 6. Se selecciona una métrica de la tabla.
Flujos alternativos:	1. No modifica el filtro de año. 2. No modifica el filtro de meses. 3. No modifica el filtro de complejidades.
Requisitos de interfaz de usuario:	1. Tabla con las métricas disponibles. 2. Campo para filtrar los datos de la métrica por años. 3. Picker para filtrar la métrica por complejidades. 4. Picker para filtrar la métrica por meses. 5. Gráfico que muestra los resultados de la métrica.

Tabla 5.11: Casos de Uso, Ver Gestores.

Nombre:	Ver Gestores
Descripción:	Los usuarios administradores pueden ver los gestores de la empresa.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá visto los gestores.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se coloca el cursor sobre la opción “Administración”. 4. Se despliegan las opciones que incluye “Métricas”. 5. Se selecciona la opción “Gestionar Gestores”. 6. Se actualiza la ventana principal con el respectivo menú y la tabla que contiene los gestores.
Flujos alternativos:	
Requisitos de interfaz de usuario:	1. Botón dentro del submenú de “Administración”. 2. Tabla con los gestores. 3. Botón para agregar un gestor. 4. Botón para modificar un gestor. 5. Botón para eliminar un gestor.

Tabla 5.12: Casos de Uso, Agregar Gestores.

Nombre:	Agregar Gestores
Descripción:	Los usuarios administradores pueden agregar nuevos gestores.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá agregado un nuevo gestor.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 4. Se despliegan las opciones que incluye “Administración”. 5. Se selecciona la opción “Gestionar Gestores”. 6. Se actualiza la ventana principal con el respectivo menú y tabla de gestores. 7. Se da clic sobre el botón Agregar. 8. Se abre el modal con los datos a solicitar. 9. Se rellenan todos los campos. 10. Se presiona Aceptar y se guarda el nuevo gestor.
Flujos alternativos:	1. El usuario abre el modal de Agregar gestor, pero lo vuelve a cerrar con Cancelar.
Requisitos de interfaz de usuario:	1. Botón dentro del submenú de “Administración”. 2. Modal con los siguientes campos para digitar: • Correo • Nombre • Primer Apellido • Segundo Apellido • Área 3. Botón de Aceptar para guardar el gestor. 4. Botón de Cancelar para devolverse.

Tabla 5.13: Casos de Uso, Modificar Gestores.

Nombre:	Modificar Gestores
Descripción:	Los usuarios administradores pueden modificar gestores.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá modificado los datos de un gestor.
Flujo básico:	1. Se inicia sesión. 2. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 3. Se despliegan las opciones que incluye “Administración”. 4. Se selecciona la opción “Gestionar Gestores”. 5. Se selecciona un gestor de la tabla. 6. Se presiona el botón de Modificar. 7. Se reemplazan los datos que muestra el modal. 11. Se presiona Aceptar para guardar los cambios realizados.
Flujos alternativos:	1. Se presiona el botón de Modificar sin haber seleccionado un gestor antes, se muestra una alerta. 2. Se selecciona un gestor y se presiona el botón de Modificar, se realizan cambios, pero se presiona el botón de Cancelar.
Requisitos de interfaz de usuario:	1. Tabla de gestores. 2. Botón para Modificar. 3. Modal con los siguientes campos con información incluida: • Correo • Nombre • Primer Apellido • Segundo Apellido • Área 4. Botón para Aceptar. 5. Botón para Cancelar.

Tabla 5.14: Casos de Uso, Eliminar Gestor.

Nombre:	Eliminar Gestor
Descripción:	Los usuarios administradores pueden eliminar gestores.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá eliminado a un gestor.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 4. Se despliegan las opciones que incluye “Administración”. 5. Se selecciona la opción “Gestionar Gestores”. 6. Se actualiza la ventana principal con el respectivo menú y tabla de gestores. 7. Se selecciona un gestor de la tabla. 8. Se presiona el botón de Eliminar. 12. Se presiona el botón de Confirmar.
Flujos alternativos:	1. Se presiona el botón de Eliminar sin haber seleccionado un gestor antes, se muestra una alerta. 3. Se selecciona un gestor y se presiona el botón de Eliminar, pero se presiona el botón de Cancelar.
Requisitos de interfaz de usuario:	1. Tabla de gestores. 2. Botón para Eliminar. 6. Modal para confirmar o eliminar.

Tabla 5.15: Casos de Uso, Ver Usuarios.

Nombre:	Ver Usuarios
Descripción:	Los usuarios administradores pueden ver una tabla que muestra a todos los usuarios registrados en el sistema.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá visto los usuarios.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 4. Se despliegan las opciones que incluye “Administración”. 5. Se selecciona la opción “Gestionar Usuarios”. 6. Se actualiza la ventana principal con la respectiva tabla de usuarios.
Flujos alternativos:	
Requisitos de interfaz de usuario:	1. Botón dentro del submenú de “Administración”. 2. Tabla con los usuarios existentes en el sistema. 3. Botón para modificar usuarios 4. Botón para agregar usuarios.

Tabla 5.16: Casos de Uso, Agregar Usuarios.

Nombre:	Agregar Usuarios
Descripción:	Los usuarios administradores pueden agregar nuevos usuarios.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá agregado un nuevo usuario al sistema.
Flujo básico:	1. Se inicia sesión. 2. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 3. Se despliegan las opciones que incluye “Administración”. 4. Se selecciona la opción “Gestionar Usuarios”. 5. Se presiona el botón “Agregar”. 6. Aparece un modal con el formulario para añadir un nuevo usuario. 7. Se completa el formulario. 7. Se presiona “Aceptar” para agregar el nuevo usuario.
Flujos alternativos:	1. El usuario cancela la opción de agregar. 2. Error si el usuario deja los campos vacíos. 3. Error si el usuario no confirma la contraseña. Error si el usuario ya existe.
Requisitos de interfaz de usuario:	1. Lista con los usuarios existentes. 2. Botón para agregar. 3. Modal con el formulario para añadir los datos del usuario. 4. Campos para ingresar el correo, nombre, nuevo rol, contraseña y confirmar contraseña. 5. Picker para seleccionar un rol ya existente o la opción de agregar uno nuevo. 6. Checkbox para marcar la opción de activar usuario. 7. Botón para aceptar y guardar al usuario. 5. Botón para cancelar.

Tabla 5.17: Casos de Uso, Modificar Usuarios.

Nombre:	Modificar Usuarios
Descripción:	Los usuarios administradores pueden modificar usuarios.
Actores:	Usuario Administrador
Precondición:	El usuario administrador debe haber iniciado sesión correctamente.
Postcondición:	El usuario habrá modificado los datos de un usuario.
Flujo básico:	1. Se inicia sesión. 2. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 3. Se despliegan las opciones que incluye “Administración”. 4. Se selecciona la opción “Gestionar Usuarios”. 5. Se selecciona un usuario de la lista con usuarios existentes. 6. Se presiona el botón “Modificar”. 7. Aparece un modal para modificar los datos de un usuario. 8. Se realizan los cambios deseados. 8. Se presiona “Aceptar” para guardar los cambios del usuario.
Flujos alternativos:	1. El usuario cancela la opción de modificar. 2. Error si el usuario no selecciona un usuario antes de presionar el botón de modificar.
Requisitos de interfaz de usuario:	1. Lista con los usuarios existentes. 2. Botón para modificar. 3. Modal con el formulario para modificar los datos del usuario. 4. Campos para cambiar el correo, nombre, nuevo rol, contraseña y confirmar contraseña. 5. Picker para cambiar el rol ya existente o la opción de agregar uno nuevo. 6. Checkbox para marcar o desmarcar la opción de usuario activo. 7. Botón para aceptar y guardar los cambios. 6. Botón para cancelar.

Tabla 5.18: Casos de Uso, Servicio Test Rail – Base de datos.

Nombre:	Cargar datos de TestRail a la base de datos
Descripción:	Los usuarios administradores pueden actualizar la base de datos con información proveniente de TestRail.
Actores:	Usuario Administrador
Precondición:	El usuario debe haber iniciado sesión correctamente y debe de ser administrador.
Postcondición:	El usuario habrá actualizado la base de datos.
Flujo básico:	1. Se ingresa a la página inicial del sistema. 2. Se inicia sesión. 3. Se coloca el cursor sobre la opción “Administración” del menú de opciones. 4. Se despliegan las opciones que incluye “Administración”. 5. Se selecciona la opción “Cargar datos de TestRail a Base de Datos”. 6. Se muestra un mensaje de confirmación. 9. Se selecciona la opción “Ok” para ejecutar el servicio.
Flujos alternativos:	El usuario selecciona cancelar en el mensaje de confirmación.
Requisitos de interfaz de usuario:	1. Botón dentro del submenú de “Administración”. 2. Mensaje de confirmación preguntando si se desea ejecutar el servicio. 7. Botones para ejecutar o cancelar la ejecución del servicio.

4.1.6 Modelo Conceptual

El siguiente modelo representa los componentes y las respectivas interacciones de la aplicación Web Metricapp. Este modelo presenta las funciones y la comunicación del sistema por medio de la aclaración de las ejecuciones que se pueden realizar, además de la proveniencia de los datos a utilizar.

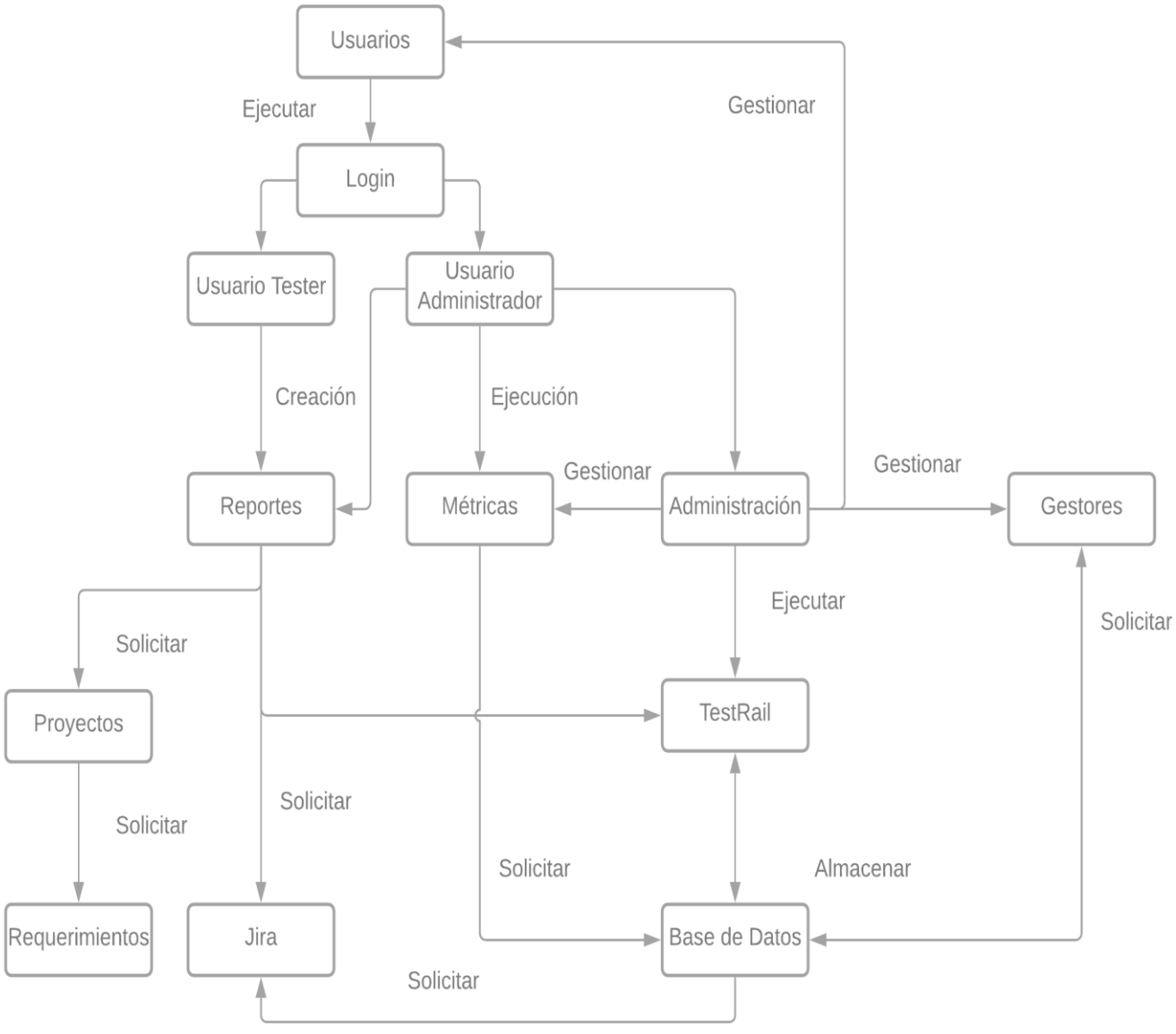

Figura 4: Modelo Conceptual. Fuente: Elaboración propia.

4.1.7 Glosario de términos

- Metricapp: Sistema del departamento de QA para realizar reportes y ejecutar métricas sobre los requerimientos de los proyectos.
- Proyecto: Es la planificación de una idea que se quiere llevar a cabo.
- Requerimientos: Son las tareas que hay que realizar para poder concluir un proyecto.
- Usuario Tester: Este usuario solo tiene permisos para generar reportes diarios, generar reportes de finalización de proyectos de pruebas y de realizar cambio de contraseña.
- Usuario Administrador: Este usuario tiene todos los permisos del sistema, puede generar todo tipo de reporte, ejecutar métricas y administrar cada apartado del sistema.
- Reporte Diario: Reporte que marcan las pruebas ejecutadas día a día sobre un requerimiento.
- Reporte de Finalización de proyecto de pruebas: Reporte final de un proyecto luego de haber hecho todas las pruebas correspondientes.
- Métricas: Proporcionan orientación y seguimiento de los requerimientos sobre temas como tiempos, estimaciones, defectos, otros.
- Estimaciones: Son cálculos de horas para algunas tareas sobre requerimientos.
- Histórico de Resultados: Consiste en la visualización detallada de las métricas que se han ejecutado desde el inicio del uso de Metricapp.
- Test Rail: Es un gestor de casos de prueba utilizado en el Aseguramiento de la Calidad.

4.1.8 Requerimientos no funcionales

Tabla 6: Requerimientos no funcionales.

Código	Descripción	Prioridad
RNF-01	La RESTful API se conectará con una base de datos ya creada en MySQL	Alta
RNF-02	La RESTful API podrá ser consultada desde cualquier medio que sea capaz de realizar peticiones http.	Alta
RNF-03	El formato de las response de la RESTful API será JSON.	Alta
RNF-04	Para poder obtener respuestas exitosas de la RESTful API, se deberá ingresar un JSON Web Token. Generado por el inicio de sesión de un usuario.	Alta
RNF-05	Las comunicaciones que involucren a la base de datos deberán de ser lo óptimas posibles, para obtener tiempos de respuesta aceptables.	Media
RNF-06	Se podrá utilizar el sistema solo si se está conectado a la red de la empresa.	Media
RNF-07	El sistema web podrá ser accedido por medio de los navegadores Chrome, Edge y Firefox como mínimo.	Media
RNF-08	La RESTful API deberá soportar múltiples peticiones provenientes de distintas conexiones	Media
RNF-09	El tiempo de aprendizaje del sistema será no más de una hora.	Baja
RNF-10	La RESTful API establecerá conexión con la API de JIRA, los tiempos de respuesta deben ser lo óptimos posibles.	Alta
RNF-11	El acceso al sistema está garantizado por medio de un usuario y una contraseña registrados en la base de datos.	Alta
RNF-12	El sistema proporciona mensajes de error claros para orientar a los usuarios.	Media
RNF-13	La RESTful API extraerá información de TestRail y en tiempos aceptables.	Alta

4.2 Diseño de la plataforma de software

4.2.1 Interfaces de Usuario

- Inicio de Sesión

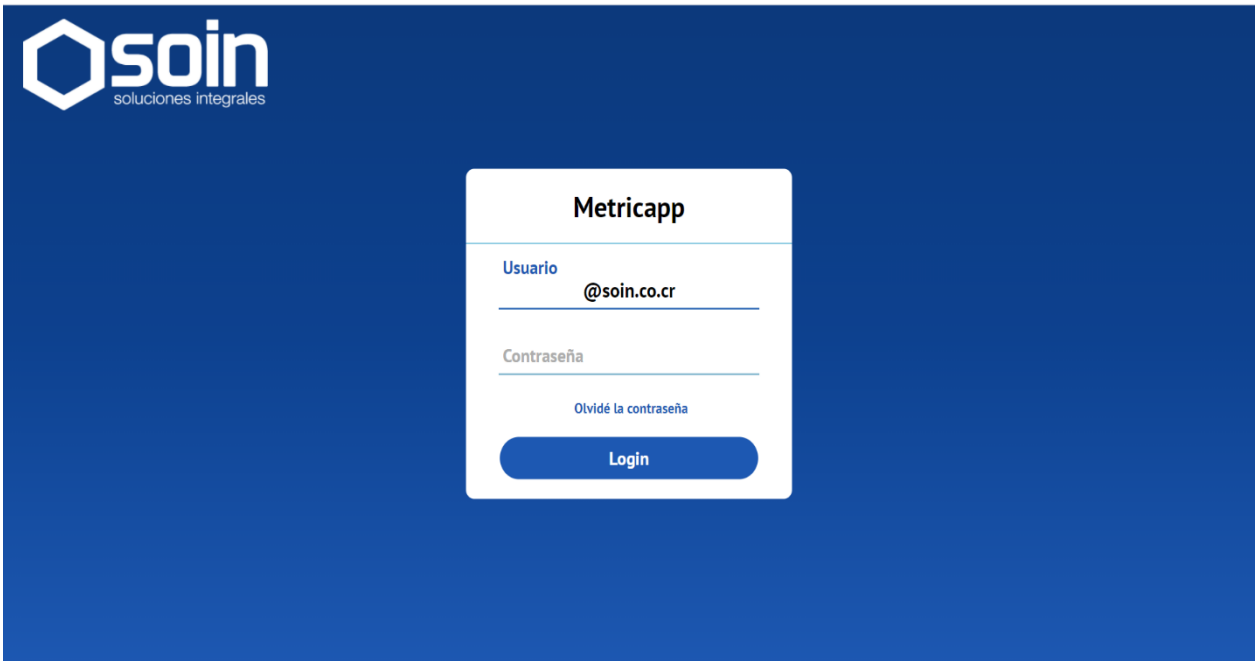

Figura 5.1: Interfaz Inicio de sesión. Fuente: Elaboración propia.

- Página inicial

Figura 5.2: Interfaz Página inicial. Fuente: Elaboración propia.

- Generar Reporte Diario

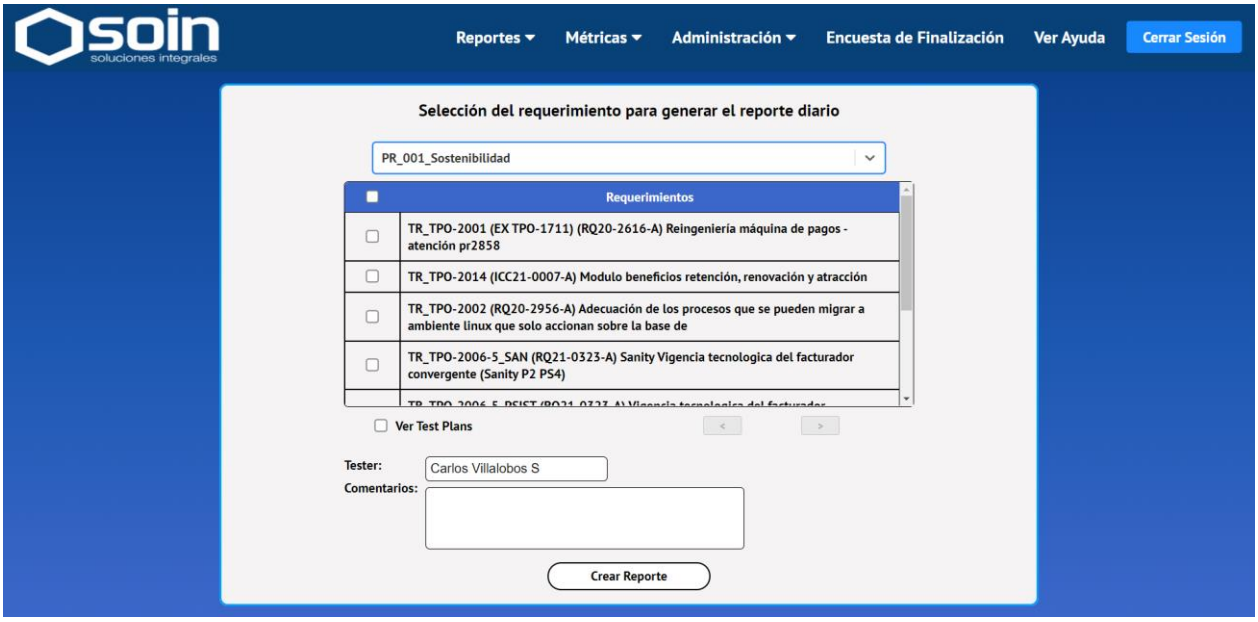

Figura 5.3: Interfaz Generar reporte diario. Fuente: Elaboración propia.

- Generar Reporte de Finalización

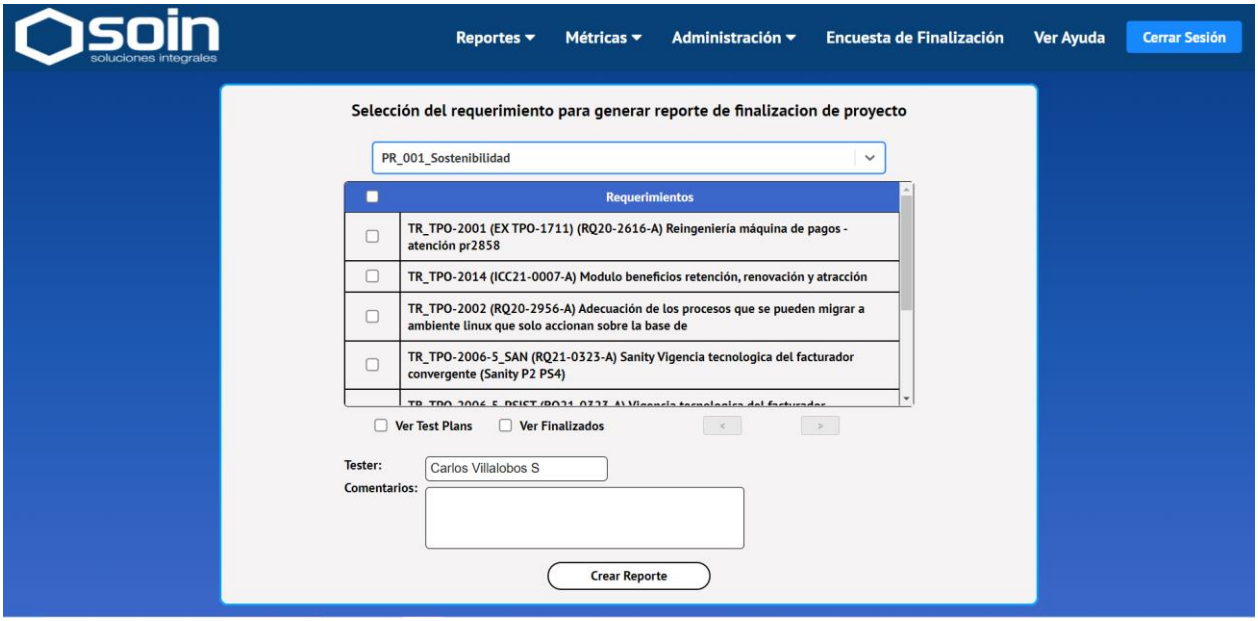

Figura 5.4: Interfaz Generar reporte de finalización. Fuente: Elaboración propia.

- Ejecutar Métricas

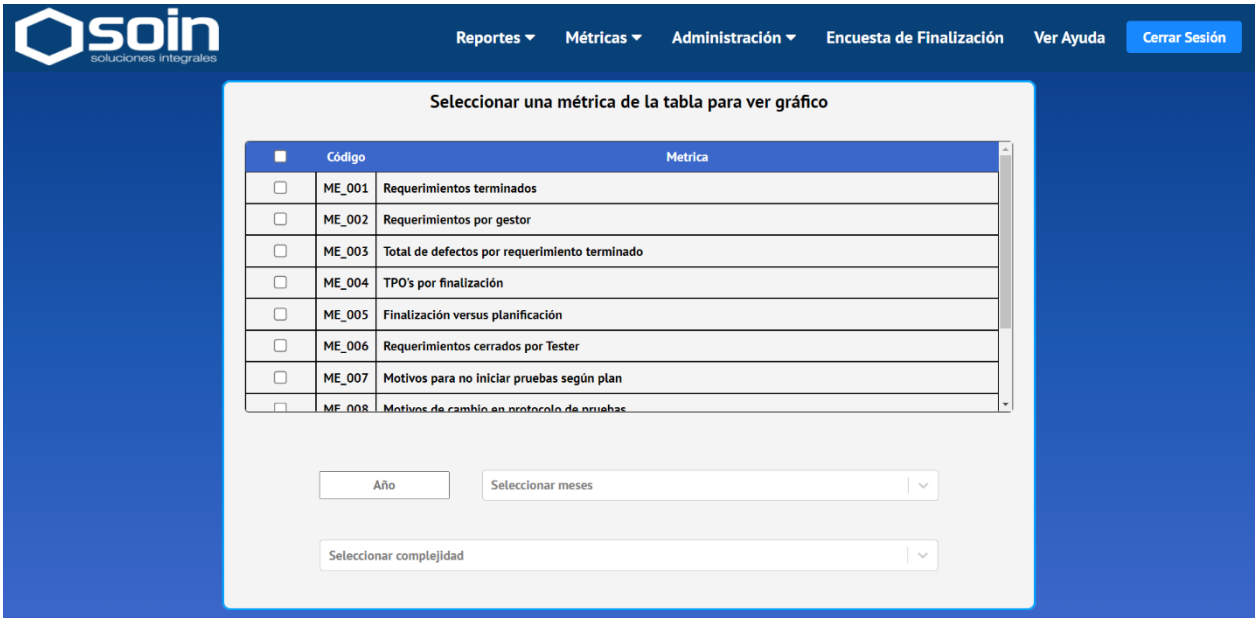

Figura 5.5: Interfaz Ver métricas. Fuente: Elaboración propia.

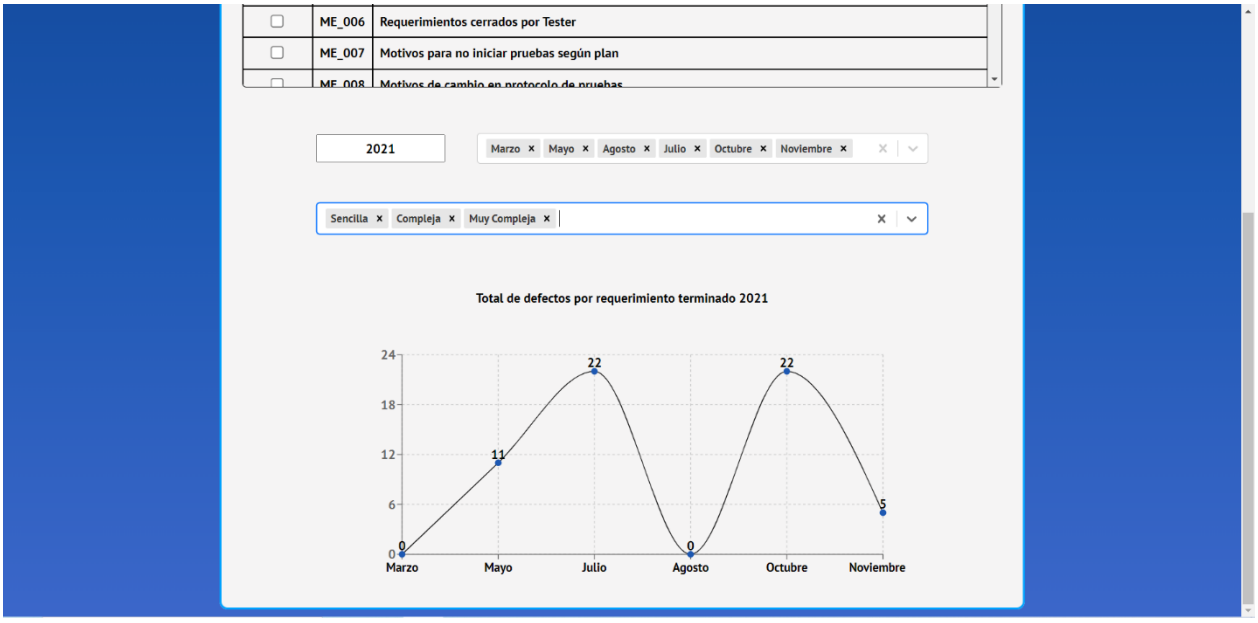

Figura 5.6: Interfaz Métrica ejecutada. Fuente: Elaboración propia.

- Gestionar Usuarios

Figura 5.7: Interfaz Ver usuarios. Fuente: Elaboración propia.

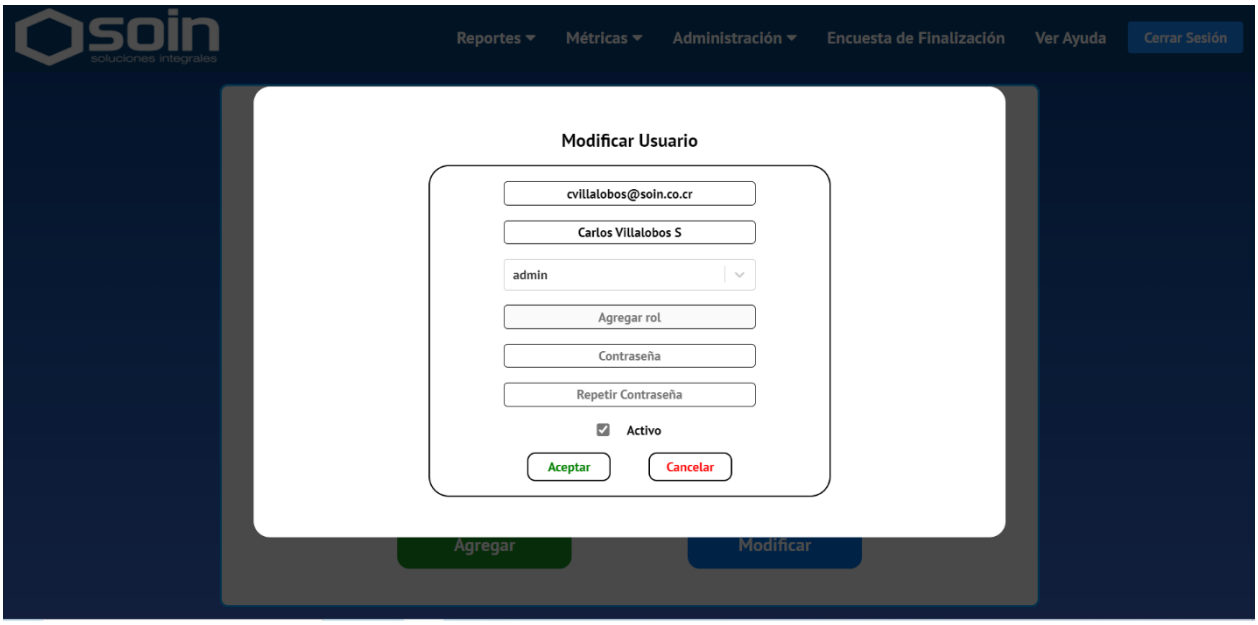

Figura 5.8: Interfaz Modificar usuarios. Fuente: Elaboración propia.

- Gestionar Gestores

Figura 5.9: Interfaz Ver gestores. Fuente: Elaboración propia.

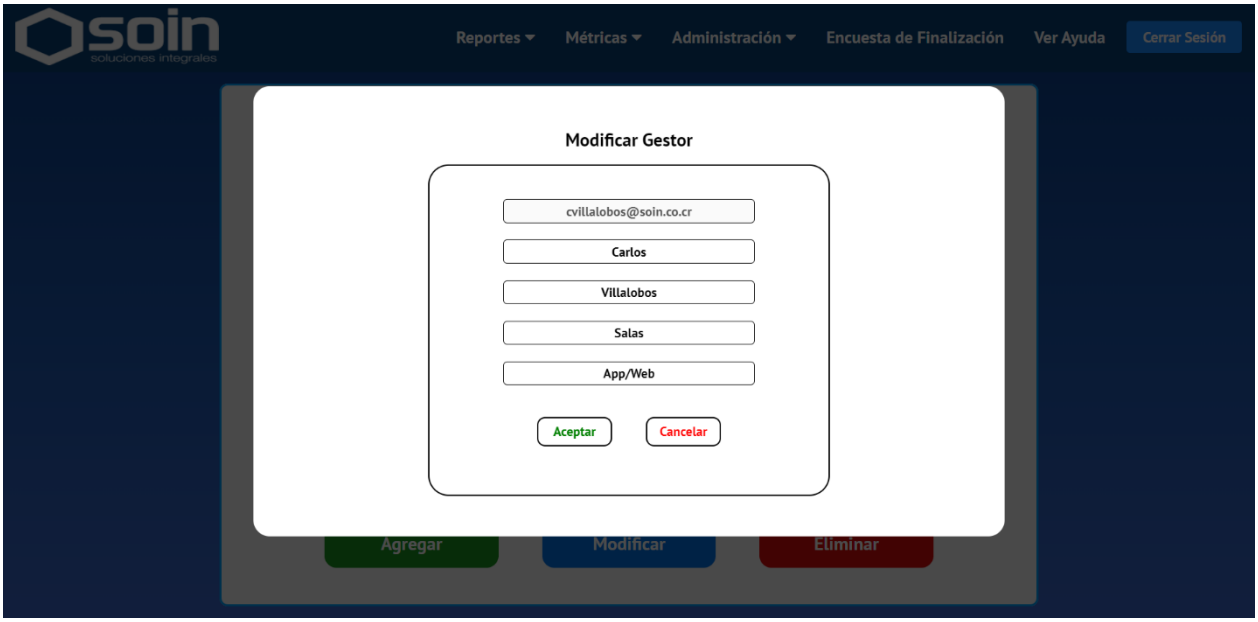

Figura 5.10: Interfaz Modificar gestor. Fuente: Elaboración propia.

- Cargar información de Test Rail a Base de Datos

Figura 5.11: Interfaz Ejecutar servicio Test Rail – Base de datos. Fuente: Elaboración propia.

- Ver Ayuda

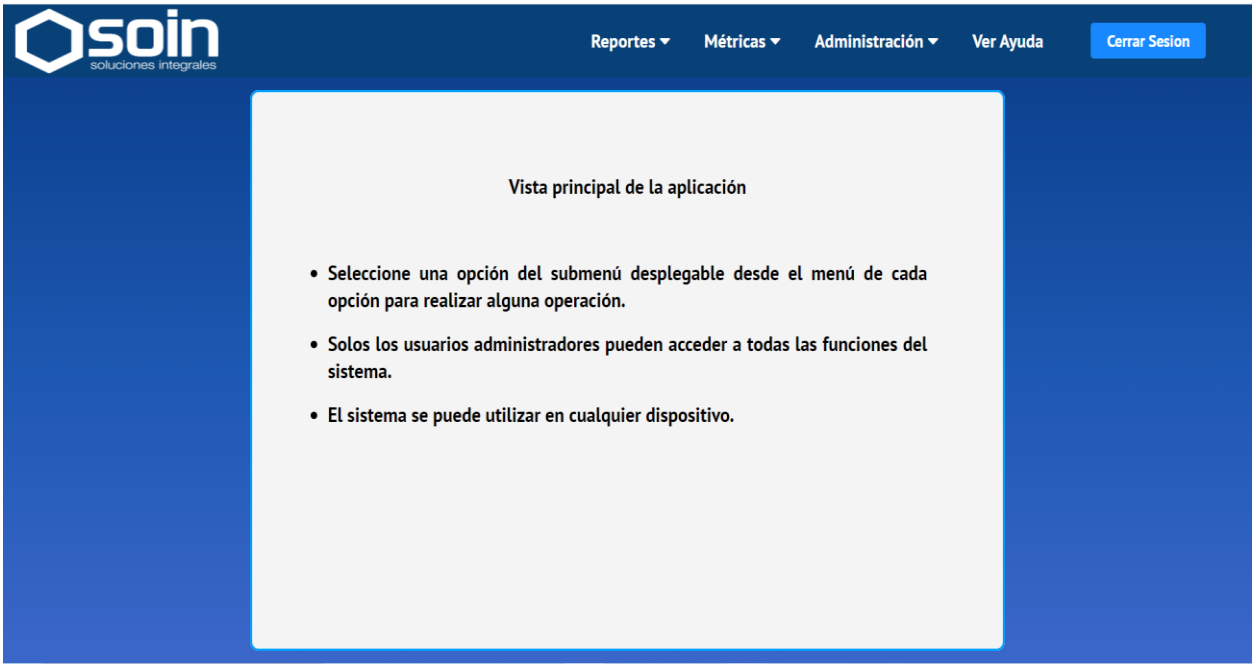

Figura 5.12: Interfaz Ver Ayuda. Fuente: Elaboración propia.

4.2.2 Diagrama Entidad – Relación de la base de datos

Como Metricapp es un sistema ya existente antes del inicio de este proyecto, cuenta ya con una base de datos. Para efectos del proyecto, esta base no sufre de ningún cambio, los detalles de las tablas fueron definidos anteriormente.

Detalles:

- **T_PROYECTO**: Esta tabla almacena la información de un proyecto de pruebas.
- **T_TAREA**: Esta tabla asocia el proyecto con la tarea usada para reportar las horas.
- **T_TAREA_X_TAREA**: De esta tabla no se tiene detalle.
- **T_PARAM_SIGES_TIEMPO**: Esta tabla guarda el registro del tiempo trabajado por cada tarea realizada.
- **T_USUARIO_SIGES**: Tabla que almacena usuario del sistema SIGES que es donde se cargan las horas laburadas.
- **T_PARAM_ESTIMACION_TIEMPO**: Tabla que almacena el tiempo estimado para cada proyecto.
- **T_USUARIO_APP**: Tabla para almacenar los usuarios que usan la aplicación Metricap.
- **T_PROYECTO_TESTRAIL**: Tabla donde se almacena el detalle de cada proyecto que esta creado en TestRail.
- **T_REQUERIMIENTO**: Tabla que almacena cada requerimiento creado en TestRail.
- **T_METRICA**: Tabla que almacena detalle de las métricas implementada en la versión JAVA de Metricapp.
- **T_HISTORICO_RESULTADO**: Tabla pensada para almacenar los resultados de pruebas para cada requerimiento.

Diagrama:

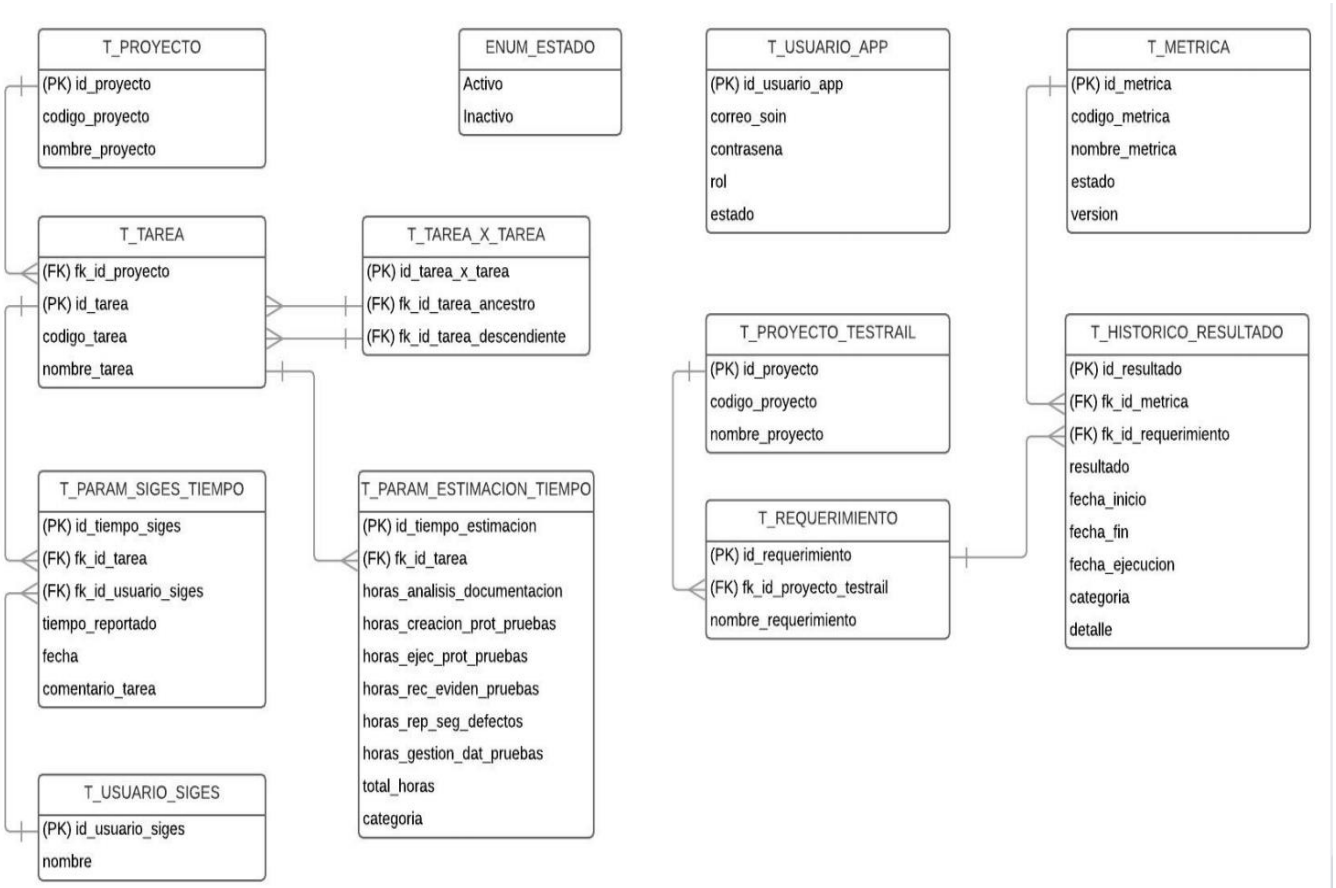

Figura 6: Diagrama Entidad – Relación de la base de datos. Fuente: Elaboración propia.

Capítulo V

5 Plataforma de software

5.1 Aplicación

5.1.1 Introducción al desarrollo de la plataforma de software

El desarrollo de este proyecto consiste en dos fases, primero una mejora en Metricapp actual y luego una migración de este a un sistema web completamente nuevo. Siguiendo el cronograma mostrado en secciones anteriores, se establecieron alcances y tiempos específicos para ambas fases. Aunque en la tabla de requerimientos funcionales se hayan detallado los visuales y funcionales, no se descarta que, con disponibilidad, requerimientos que se definieron como visuales puedan concluir como funcionales. Para la empresa, el módulo que se encarga de generar reportes es el más importante y el más necesario, por lo que a este se le da prioridad por encima de los demás, una vez finalizado este, se pueden pasar requerimientos visuales a funcionales.

5.1.2 Tareas realizadas para desarrollar la plataforma de software

Como se ha mencionado anteriormente, el proyecto consta de dos fases, las cuales se han marcado en el cronograma. A continuación, se detallarán más sus partes de desarrollo.

5.1.2.1 Fase 1

Esta fase consiste en una actualización del sistema Metricapp actualmente utilizado por la empresa. Dicha actualización consiste en integrar la API de Jira a los reportes que generan los usuarios. Antes de iniciar a tocar el código en Java de Metricapp, se estudió y probó la nueva API de Jira. Se recolectaron algunos endpoints necesarios a implementar en Metricapp, se probaron y guardaron en

Postman, se comparó la estructura de algunas respuestas de los endpoints con las respuestas de la API ya utilizada en Metricapp (API de Bugzilla) y se verificaron las respuestas con la plataforma web de Jira que utiliza la empresa para administrar los proyectos.

También se ha estudiado Metricapp en general, principalmente las herramientas y los detalles que se involucran con la generación de reportes diarios y reportes de finalización de pruebas. Además, como se mencionó anteriormente, se realizaron comparaciones con los endpoints de cada API y la manera en la que los utiliza Metricapp, esto con el fin de obtener antes de trabajar, una idea clara sobre los cambios que se deben realizar. Luego de haber ampliado el conocimiento sobre esta API y el sistema Metricapp, se inició con la integración en Metricapp. Esta integración de la API de Jira no implica que la API anterior dejará de existir en el sistema, ya que son proyectos que la empresa se encuentra utilizando y lo seguirán haciendo hasta finalizar esos proyectos. Los proyectos nuevos si serán iniciados en algún momento desde Jira y así poco a poco se dejará de utilizar Bugzilla. Debido a que ambas APIs se mantendrán en funcionamiento, la integración consiste en el uso de la nueva API sin alterar los proyectos existentes, por lo que se agregan condiciones para ser utilizada, de momento, en proyectos específicos. Estos proyectos son utilizados como plan Piloto para ir adaptando a la empresa a la nueva API y a la nueva forma de administrar los proyectos con Jira.

5.1.2.2 Fase 2

Luego de haber actualizado Metricapp, se inicia con la migración a una plataforma web. El desarrollo de esta etapa incluye tanto Backend como Frontend. Debido a que las interfaces de usuario se habían desarrollado ya en la etapa del prototipo, tanto Backend como Frontend se fueron trabajando de la mano, al mismo tiempo. El desarrollo de esta fase inicio con la autenticación en el sistema, luego se pasó a la generación de ambos tipos de reportes, terminando los requerimientos funcionales con la administración de los usuarios. Al ver que se contaba con más tiempo de lo previsto, se ha procedido a avanzar varios requerimientos marcados

anteriormente como visuales, como ver histórico de resultados o cargar datos de Test Rail a la base de datos.

5.1.3 Resultados obtenidos en el desarrollo de la plataforma de software

El siguiente diagrama de la arquitectura conceptual de Metricapp refleja las interacciones que se han logrado durante el desarrollo de la plataforma. El usuario al ingresar al sistema provoca que Metricapp se empiece a relacionar con el servidor, quien es el que interactúa con los tres elementos mostrados en el diagrama dependiendo de las acciones que realice dicho usuario.

En cuanto al sistema web, este comparte todas las funcionalidades que tiene el sistema utilizado actualmente, pero se le han podido añadir algunas funciones nuevas. Por ejemplo, ahora los reportes generados en PDF no se descargan luego de su generación, si no que ahora los usuarios tienen una vista previa para su revisión. Otro ejemplo sería la adición de un módulo para manejar encuestas a la hora de finalizar pruebas sobre los requerimientos de los proyectos, encuesta que antes de este sistema, se utiliza por medio de Google y el correo de los usuarios.

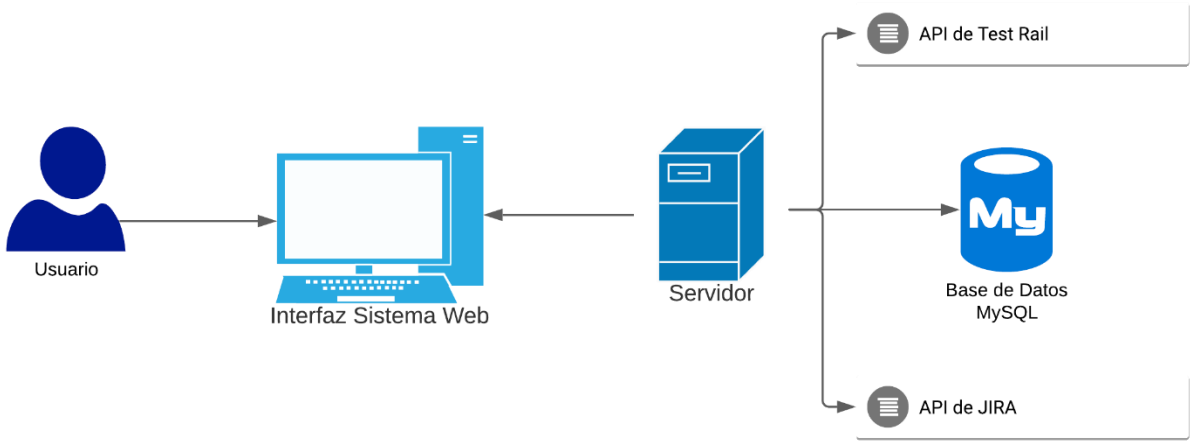

Figura 7: Arquitectura de la solución. Fuente: Elaboración propia.

5.2 Evaluación

5.2.1 Introducción a la evaluación de la plataforma de software

La aplicación Metricapp actual que utiliza la empresa no se dejará de utilizar durante el desarrollo de la integración de Jira, por lo que el sistema modificado se irá evaluando diariamente, principalmente con los reportes diarios que generan los integrantes del departamento de QA. Debido a esto, pueden surgir cambios, mejoras o peticiones adicionales sobre el trabajo a realizar. Luego de realizar estos cambios, se comunican al encargado de la empresa para que los evalúe y verifique que la API de Jira genera los mismos resultados que la API de Bugzilla, por medio de la comparación de dos reportes generados, uno con Jira y uno con Bugzilla. Gracias a la plataforma web de Jira, los reportes generados con la integración también se pueden evaluar revisando la información de un requerimiento en dicha plataforma.

Gracias a contar ya con una versión de Metricapp, el desarrollo web se irá probando durante el desarrollo de manera que trabaje de igual manera con el sistema mencionado anteriormente. Sobre los componentes de React se realizaron algunas pruebas unitarias para verificar que estos estén funcionando y las vistas se estén mostrando correctamente. Respecto al Backend, todos los endpoints han sido probados por medio de Postman luego de la creación de cada uno. Una vez probados los endpoints, se realizó a la implementación de estos en React para mostrar datos reales y realizar las funciones de Metricapp. Aunque las pruebas reales sobre el sistema web serán cuando el departamento de QA deje de utilizar la aplicación de escritorio de Metricapp, durante el desarrollo se ha probado el sistema por medio de la revisión de los reportes generados en esta plataforma, en conjunto con el encargado del proyecto.

5.2.2 Tareas realizadas para la evaluación la plataforma de software

Como ya se ha mencionado, Metricapp es un sistema que se utiliza diariamente por parte de todo el departamento de QA y por medio de la generación de reportes diarios. De esta manera es que se van a realizar pruebas, ya que,

realizando reportes de pruebas o finales, se verificará que el sistema satisfaga con las necesidades de los usuarios. Dichas necesidades quedarán cubiertas gracias a las pruebas realizadas por el desarrollador y el encargado del proyecto por parte de la empresa, que como se mencionó anteriormente, es quien da el visto bueno para dar una tarea por concluida satisfactoriamente. Además de probar el sistema por medio de su utilización diaria, existe la posibilidad de comparar tanto la aplicación de escritorio, con años de uso ya, y la aplicación web desarrollada en este proyecto, con el fin de verificar que el sistema funciona correctamente.

5.2.3 Resultados obtenidos en la evaluación de la plataforma de software

El poder recibir retroalimentación diaria facilitó obtener resultados exitosos en ambas fases del proyecto. En el momento que se presentaba la necesidad de corregir o realizar un cambio, se priorizaba esa tarea antes de continuar o iniciar una nueva. Esta forma de trabajo o de comunicación benefició mucho al producto final, que cumple con lo pedido por la empresa.

En cuanto a la integración de Jira en Metricapp Java (sistema que actualmente se utiliza), se registra un mejor tiempo de respuesta a la hora de generar reportes diarios y de finalización de pruebas, en comparación con la API de Bugzilla. Además, sobre estos reportes se tuvieron que hacer algunos cambios visuales, ya que Jira provee nombres distintos a Bugzilla, por ejemplo, para los estados de un bug, estos cambios se ven reflejados en la generación de ambos reportes.

El resultado del nuevo sistema Web para Metricapp se da por exitoso por la aprobación del encargado del proyecto, Maikol Picado. Los resultados obtenidos, en comparación a la aplicación de Metricapp en Java, indican que se desarrollaron exitosamente los requerimientos necesarios para el uso de la aplicación.

Capítulo VI

6 Conclusiones

La importancia que tiene Metricapp dentro del departamento de calidad, sumado a la ya existencia y uso de este sistema, resalta la importancia del desarrollo de este proyecto. Con el fin de dar soluciones a un sistema tan importante dentro de un departamento que se encarga de garantizar la calidad de todos los productos de la empresa es importante incluir en un futuro proyecto, posibles mejoras de rendimiento o nuevas funcionalidades, por ejemplo, agregar nuevas maneras de ejecutar y graficar las métricas o de evaluar los defectos y mejoras de los requerimientos en Jira.

Metricapp es un sistema que facilita y aporta mucho al trabajo del departamento, que ahora esté desarrollado en un ambiente web y se haya reemplazado la API para el control de los defectos, ha mejorado el sistema en cuestión de rendimiento. Debido a que el aplicativo de Metricapp desarrollado en Java no se ha dejado de utilizar, se pueden realizar comparaciones con el sistema desarrollado en este proyecto y ver las mejoras que el nuevo ya ofrece. Además, encontrarse en una ambiente Web permite poder innovar sobre el sistema y agregar más funcionalidades a futuro en caso de ser necesario, también, la actualización de API ya muestra una manera más eficiente de manejar y obtener los datos de los proyectos a administrar.

Como ya se ha mencionado en diferentes secciones de este documento, son muchas las ventajas que tiene un sistema web sobre un sistema de escritorio, en este caso utilizado por todo el departamento de calidad. Durante el desarrollo ya se ha podido notar las diferencias que existen entre tecnologías en términos de velocidad de ejecución para la generación de reportes, por ejemplo. Las demás funciones del sistema también han mostrado mejoras, por lo que se puede garantizar que migrar el ambiente del proyecto ha sido una buena decisión.

Respecto a las etapas de desarrollo del proyecto, se han podido concluir de manera satisfactoria todos los requerimientos establecidos como funcionales. Se ha mencionado la posibilidad de agregar mejoras sobre otros módulos y la posibilidad de incluir nuevas funcionalidades que Metricapp no ha tenido en ningún momento. Funcionalidades que, el equipo de trabajo de la empresa, creen que estando estas en Metricapp puede traer mayores beneficios a sus trabajos, en lugar de estar estas esparcidas en otros sistemas. Por lo que, el desarrollo del proyecto ha garantizado mejora del rendimiento en Metricapp y ha dejado como mayor hallazgo, la escalabilidad o la posibilidad y motivación de incorporar nuevas funcionalidades al sistema.

En cuanto a las conclusiones personales respecto al desarrollo del proyecto y el trabajo cotidiano realizado dentro de la empresa, se puede concluir un proceso realizado satisfactoriamente, en el que se han podido desarrollar más requerimientos de lo previsto, y ha sido avalado por los involucrados en el proyecto. La experiencia ha sido muy grata y en ningún momento se ha presentado algún inconveniente, tanto en el ámbito personal como laboral.

6.1 Recomendaciones

Para la generación de los reportes, tanto diarios como de finalización, Jira necesita contar con el código del proyecto y el nombre del requerimiento, esto para obtener los defectos u otros datos. Con esto en consideración, cada que se agrega un proyecto a Jira, se necesita agregar también en Metricapp Web, manualmente, por medio de un archivo Excel que contiene el nombre del archivo y su código de Jira. Como recomendación, sería bueno agregar estos datos en los nombres de los proyectos y requerimientos para evitar tener que modificar el código en estos casos, para evitar la pérdida de tiempo y la posibilidad de provocar fallos por errores al digitar. Sería dar un formato estándar para beneficiar el uso de Metricapp.

En cuanto a las funcionalidades que tiene Metricapp Web, las cuales fueron replicadas, no existe recomendación alguna. La recomendación recaería más en aprovechar tanto las librerías y ventajas que ofrecen React y Node para añadir más

funciones. A lo largo del desarrollo se mencionó la posibilidad o el deseo de agregar módulos para visualizar algunos gráficos, añadir encuestas que los usuarios puedan contestar o poder administrar cualquier otro dato que utilice el departamento en su trabajo. Para el departamento sería más sencillo tener todas las actividades a realizar en un mismo aplicativo, a tener que visitar distintos sitios, como Google Forms en caso de querer realizar una encuesta. Además, ya Metricapp posee muchos datos que se utilizan en los reportes, por ejemplo, si se quisiera implementar una encuesta sobre los proyectos finalizados a los usuarios, se podrían evitar hacer algunas preguntas y obtener las respuestas de las conexiones a Test Rail o Jira, así los usuarios no tienen que revisarlos y anotarlos, también estos se ahorrarían tiempo y la encuesta quedaría abierta a poder agregar más respuestas provenientes de las API, en caso de ser necesario.

Tratándose del departamento de calidad, este no conoce o no suele estar familiarizado con las tecnologías utilizadas en la actualidad para el desarrollo de aplicaciones web. Como recomendación, se podría mantener la acción de preguntar o dejar proponer al desarrollador dichas tecnologías a utilizar, siempre verificadas por las que son utilizadas en la actualidad, como ha sido mi caso, en el que se me ha permitido dar la propuesta de las tecnologías, las cuales fueron aprobadas y utilizadas.

Y en cuanto a las recomendaciones personales hacia la carrera de computación, recomendaría incluir trabajos en conjunto con empresas, no muy grandes como el curso de Proyecto de Ingeniería, pero tal vez si pequeñas tareas para ir viviendo un poco el trabajo real antes de salir de la universidad.

6. Bibliografía

Qué es React. Por qué usar React. (2019, 25 febrero). Desarrollo Web. https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html

Introducción a Express/Node - Aprende sobre desarrollo web | MDN. (2021, 23 agosto). MDN Web Docs. https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

Express - Infraestructura de aplicaciones web Node.js. (s. f.). Express. Recuperado 25 de agosto de 2021, de https://expressjs.com/es/

Vega, R. (2021, 8 junio). *Qué es JIRA y cómo lo utiliza un Equipo de desarrollo y un Product Owner*. Netmind. https://netmind.net/es/que-es-jira-y-como-lo-utiliza-un-equipo-de-desarrollo-y-un-product-owner/

Introduction to TestRail. (2021, 17 agosto). TestRail. https://www.gurock.com/testrail/docs/user-guide/getting-started/walkthrough

Aggarwal, S. (2018, marzo). *Modern Web-Development using ReactJS*. http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf

Liang, L. (2017). *Express Supervision System Based on NodeJs and MongoDB*. IEEE. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7960064

Colectiva, N. (2020, 5 agosto). *5 Populares Aplicaciones que usan tecnología React (React Native, React JS)*. Blog Nube Colectiva. https://blog.nubecolectiva.com/5-populares-aplicaciones-que-usan-tecnologia-react/

npm: express. (2019, 26 mayo). Npm. https://www.npmjs.com/package/express

Por qué elegir el gestor de base de datos MySQL. (2018, 4 octubre). FP Online. https://fp.uoc.fje.edu/blog/por-que-elegir-el-gestor-de-base-de-datos-mysql/

MySQL: ¿Qué es? Características, Ventajas y Desventajas. (2019). MySQL. https://hostingpedia.net/mysql.html

Anexos

Los anexos deben estar referenciados. Por ejemplo, si empleó un cuestionario en la sección de procedimiento metodológico, se anota la descripción del mismo y se indica “ver anexo X” para que el lector conozca el instrumento completo.

Los anexos son documentos y recursos utilizados, consultados o de referencia para el proyecto. Se consideran formularios de encuestas, procedimientos/instrumentos para efectuar cálculos (hojas de MS Excel), Informes de evaluación o especificaciones de diseño que, por su volumen, se presentan en esta sección.